Sparse Gaussian Process Classification with Variational Inference

by

Kianoosh Ashouritaklimi

University of Glasgow, March 2021

Abstract

Gaussian processes are a particular type of stochastic processes with applications in various machine learning tasks such as regression, dimension reduction and time series modelling. One popular application is in classification, where approximate methods are required for inference and hyperparameter selection. An assortment of approximate methods have been proposed, but most of them are plagued by their computational complexity which scales as $\mathcal{O}(n^3)$, for n the number of training inputs. In this dissertation, we will look at a sparse approach to Gaussian process classification with variational inference, which tackles both the issue of computational complexity and approximate inference.

Table of Contents

List of Figures

List of Tables

List of Symbols

Miscellaneous

$\#$	Denotes "number of"
$\mathcal{O}$	Big O notation used to denote computational complexity
∇	Vector of partial derivatives
$\nabla\nabla$	Hessian matrix of second derivatives
$\sim$	Denotes the distribution of a random variable, e.g. $X \sim \mathscr{B}(p)$

Number Sets

$\mathbb{N}$	The natural numbers
$\mathbb{R}$	The real numbers
$\mathbb{R}^n$	The (vector) space of all n-dimensional real vectors, $n \geq 1$
$\mathbb{R}_{>0}$	Real numbers greater than zero

Random Variables

$\mathbb{E}(\cdot)$	Expectation of a random variable
$\mathbb{E}(\cdot)_{q(\cdot)}$	Expectation of a random variable under the probability distribution $q(\cdot)$
$\perp$	Denotes independence of random variables, e.g. $\mathbf{u} \perp \mathbf{z}$

Statistical Distributions

$\mathscr{B}(p)$	Bernoulli distribution with probability parameter p

$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ or $\mathcal{N}(\mu, \sigma^2)$ Multivariate normal distribution with mean vector $\boldsymbol{\mu}$ (μ for the 1-D case) and covariance matrix $\boldsymbol{\Sigma}$ (σ^2 for the 1-D case)

Vectors and Matrices

$|\cdot|$ Denotes the determinant of a matrix

$\mathbf{0}_n$ n-dimensional vector of zeroes

$\mathbf{A}_{i,j}$ (i, j)th element of a matrix $\mathbf{A}$

$\mathbf{I}_n$ Identity matrix of dimensions $n \times n$

$\mathbf{M}_{mn}$ Matrix of dimensions $m \times n$

$\mathrm{Tr}(\cdot)$ Denotes the trace of a square matrix, i.e. the sum of its diagonal elements

$||\cdot||$ Denotes the magnitude of a vector

Chapter 1

Introduction

In the past decade, machine learning has received wide popularity within both academia and industry. It has grown to such an extent where you can now find applications in everything from suggestions made by Apple's Siri to brewing beer[1]. Gaussian processes (GPs) are a particular type of stochastic processes that have received much interest in the machine learning community. Broadly speaking, they are infinite-dimensional generalizations of the Gaussian distribution, with applications in a wide array of machine learning tasks such as regression, classification and dimension reduction.

In this dissertation, we will focus on the application to classification problems, which is referred to as *Gaussian process classification* (GPC). GPs provide a Bayesian framework for classification, where approximate methods are required due to the intractability of the exact posterior and marginal likelihood. A number of approximate methods have been proposed (Nickisch and Rasmussen, 2008), but they all have $\mathcal{O}(n^3)$ computational complexity, where n is the number of training inputs. As a result, they are not scalable to large datasets, making them limited for practical applications.

To reduce this computational complexity, and thus achieve scalability, several *sparse* approaches to GPC have been proposed. The idea behind these sparse approaches is to use $m < n$ *inducing inputs* and *inducing variables*, rather than the entire training set, to approximate the marginal likelihood and posterior (Snelson and Ghahramani, 2005). These inducing inputs and variables are input-output pairs, much like covariates and responses, that can be either a subset of our original training set or *auxiliary pseudo-points* (Snelson

[1] https://www.forbes.com/sites/bernardmarr/2019/02/01/how-artificial-intelligence-is-used-to-make-beer/

and Ghahramani, 2005). As m is normally taken to be much smaller than n, this allows the computational complexity of GPC to be reduced.

The aim of this dissertation is to show a sparse approach to GPC with variational inference (Hensman et al., 2015), which tackles both the issue of computational complexity and approximate inference. Through tackling these issues, we will show that it provides a framework for GPC to be scaled to large datasets and achieve competitive model performance. To demonstrate these points, we will compare this approach with a standard parametric approach to classification (logistic regression), and two other GP approaches (an approach with Laplace approximations and one with variational inference only). The other GP approaches are approximate approaches without any sparse approximations.

We will first develop the theoretical foundations of GPC in chapter 2, where we will look at the aforementioned models as well as some background material. In chapter 3, we will apply the models to two simulated datasets and one real dataset consisting of handwritten digits from the U.S. Postal Service. There, we see the scalability and competitive model performance of the sparse approach with variational inference. We conclude with discussions in chapter 4.

Chapter 2

Materials and Methods

In this chapter, we will look at a sparse approach to GPC with variational inference as well as two other approximate GPC methods (one with Laplace approximations and one with variational inference only). To build up to that, we will first look at some background material. The material in section 2.1.3 is particularly important for introducing the sparse approach to GPC with variational inference.

2.1 Background Material

2.1.1 Gaussian Processes

Gaussian processes (GPs) are a natural generalization of Gaussian random variables. Formally, a Gaussian process is a collection of random variables $\{f(\mathbf{x})|\mathbf{x} \in X\}$, where f is a random variable and X is an input set, such that any finite subset of them follow a Gaussian distribution (Rasmussen and Williams, 2006, sec. 2.2). X is normally taken to be a subset of $\mathbb{R}^n$, making the inputs $\mathbf{x}$ real vectors. GPs can be fully determined by a *mean function* and a *covariance function* (also known as a *kernel*) defined as follows:

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})], \\ k(\mathbf{x}', \mathbf{x}) &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))], \end{aligned} \tag{2.1}$$

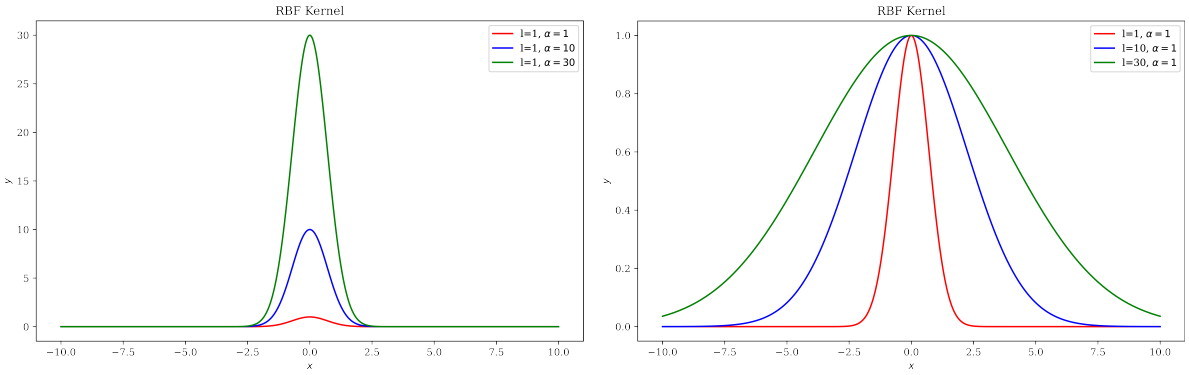
where $\mathbf{x}', \mathbf{x} \in X$, $m(\cdot)$ is the mean function and $k(\cdot, \cdot)$ is the kernel, depending on a vector of hyperparameters $\boldsymbol{\theta}$. Notation-wise, this would be written as:

$$f(\mathbf{x}) \sim \mathrm{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')). \tag{2.2}$$

Throughout this dissertation, we will refer to the *Radial Basis Function* (RBF) kernel, which is defined as follows for $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^m, m \geq 1$:

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

where $l \in \mathbb{R}_{>0}$ is called the *lengthscale* and $\alpha \in \mathbb{R}_{>0}$ the *variance* of the kernel. Here, l controls how quickly the kernel decays, whereas α controls the amplitude of the kernel. A plot of the RBF kernel centred at zero for $m = 1$ and various values of l and α is shown below:



(a) Varying α (b) Varying l

Figure 2.1: 1-D RBF kernel centred at zero for various values of l and α.

2.1.2 Gaussian Process Regression

One common application of GPs is in regression (GPR) where they allow for priors over functions. Consider the 1-D regression problem below, where a response $y \in \mathbb{R}$ is viewed as noisy realisations of a latent function f for covariates $x_i \in \mathbb{R}$:

$$y_i = f(x_i) + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \sigma^2). \tag{2.4}$$

Here, ϵ_i is some i.i.d. Gaussian noise, $\sigma \in \mathbb{R}_{>0}$ and $i = 1, \ldots, n$, $n \in \mathbb{N}$.

Now, a natural way of applying GPs in this case is to provide the normal Bayesian treatment. More specifically, we place a GP prior on the function f with mean function $m(\cdot)$ and kernel $k(\cdot, \cdot)$ s.t.:

$$f(x) \sim \mathrm{GP}(m(x), k(x, x')). \tag{2.5}$$

We can then obtain posterior and likelihood distributions in the normal Bayesian way. For illustration, we will derive the marginal log-likelihood below.

Firstly, w.l.o.g., we let $m(x) = 0$. Noting then that f is Gaussian over the *finite* set $\{x_1, \ldots, x_n\}$, we have

$$\mathbf{f} \sim \mathcal{N}(\mathbf{0}_n, \mathbf{K}_{nn}), \tag{2.6}$$

where $\mathbf{f} = (f(x_1), \ldots, f(x_n))^\top$, and $\mathbf{K}_{nn}$ is the covariance matrix obtained by evaluating the kernel at the covariates:

$$\begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \vdots & \ddots & \vdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{bmatrix} \tag{2.7}$$

Thus y is a sum of two Gaussian variables - one induced from the GP prior and the other from the Gaussian noise. As the sum of Gaussian variables is necessarily Gaussian, we then obtain the following marginal likelihood (ML):

$$\mathbf{y} \sim \mathcal{N}(\mathbf{y}|\mathbf{0}_n, \sigma^2\mathbf{I}_n + \mathbf{K}_{nn}), \tag{2.8}$$

where $\mathbf{y} = (y_1, \ldots, y_n)^\top$. From (2.8) it its then possible to derive the marginal log-likelihood as:

$$\log p(\mathbf{y}) = -\frac{1}{2}\mathbf{y}^\top(\mathbf{K}_{nn} + \sigma^2\mathbf{I}_n)^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K}_{nn} + \sigma^2\mathbf{I}_n| - \frac{n}{2}\log 2\pi. \tag{2.9}$$

With the above marginal log-likelihood, it is possible to select the kernel hyperparameters $\boldsymbol{\theta}$, as well as estimate σ^2, via the standard maximum-likelihood approach. While this approach is certainly elegant and allows for an exact solution, it is not scalable to large datasets as the inversion of $\mathbf{K}_{nn} + \sigma^2\mathbf{I}_n$ has $\mathcal{O}(n^3)$ computational complexity[1]. A similar issue arises for the computation of the posterior which is used for prediction, thus creating the necessity for approximate methods.

2.1.3 Sparse GPR with Variational Inference

One method of reducing the computational cost of prediction and hyperparameter selection in GPR is to seek approximate solutions rather than exact solutions. Sparse approaches are one common way of doing this. A number of sparse methods for GPR have been proposed

[1]Indeed, the standard algorithm for matrix inversion is Gaussian elimination and this has $\mathcal{O}(n^3)$ computational complexity for an n-dimensional matrix (Danziger, 2005).

(see Quiñonero-Candela and Rasmussen, 2005), with most of them following an inducing point method (Snelson and Ghahramani, 2005). Here, we will focus on a sparse approach with variational inference (Titsias, 2009) that also takes an inducing point approach. This approach parallels our main approach to GPC presented in section 2.2.3.

Sparse inducing point methods require using a set of *inducing variables* and *inducing inputs*, rather than all the training inputs, for approximating the ML and posterior. More specifically, inducing variables, u, are defined to be realisations of a latent function f for m inducing inputs z, where $m < n$. As m is normally taken to be much smaller than n, this allows for solutions with reduced computational complexity. Typically, these inducing inputs come from the original training inputs, although they can also be *auxiliary pseudo-points* (Snelson and Ghahramani, 2005). The selection of the inducing inputs, as well as the model hyperparameters, is the key issue with sparse methods. Loosely speaking, we would like to choose z such that the inducing variables u are as informative about the latent function f as possible.

Titsias (2009) presents an approach with variational inference to select the inducing inputs and hyperparameters for GPR. Here, we will present this approach for the 1-D regression problem presented in section 2.1.2, although the same method can easily be extended for higher dimensions.

Let the setup be as in section 2.1.2. Let $\mathbf{u} = (u_1, \ldots, u_m)^\top$ be a vector of inducing variables obtained as latent realisations of inducing inputs $\mathbf{Z}_m = (z_1, \ldots, z_m)^\top$, i.e. $\mathbf{u} = f(\mathbf{Z}_m)$, $m < n$. Consider a GP prior on f like before, where w.l.o.g we again assume a zero mean function. This results in the following distributions:

$$\begin{aligned} p(\mathbf{f}) &= \mathcal{N}(\mathbf{0}_n, \mathbf{K}_{nn}), \\ p(\mathbf{u}) &= \mathcal{N}(\mathbf{0}_m, \mathbf{K}_{mm}), \\ p(\mathbf{y}|\mathbf{f}) &= \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma^2\mathbf{I}_n), \\ p(\mathbf{f}|\mathbf{u}) &= \mathcal{N}(\mathbf{f}|\mathbf{K}_{nm}\mathbf{K}_{mm}^{-1}\mathbf{u}, \tilde{\mathbf{K}}), \end{aligned} \tag{2.10}$$

where $\mathbf{K}_{mm}$ is the covariance matrix obtained by evaluating the kernel at the inducing inputs, $\mathbf{K}_{nm}$ is the matrix obtained by evaluating the kernel at the pairs (x_i, z_j) for $i = 1, \ldots, n$, $j = 1, \ldots, m$, and $\tilde{\mathbf{K}} = \mathbf{K}_{nn} - \mathbf{K}_{nm}\mathbf{K}_{mm}^{-1}\mathbf{K}_{mn}$ for $\mathbf{K}_{mn} = \mathbf{K}_{nm}^\top$. Now, without sparse approximations, we can obtain the posterior distribution of $\mathbf{f}$ as follows:

$$p(\mathbf{f}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f})}{\int p(\mathbf{y}|\mathbf{f})p(\mathbf{f})d\mathbf{f}}, \tag{2.11}$$

where the integral is computationally intractable[2] due to the inversion of $\mathbf{K}_{nn} + \sigma^2\mathbf{I}_n$. With

[2] $\int p(\mathbf{y}|\mathbf{f})p(\mathbf{f})d\mathbf{f} = p(\mathbf{y}) \sim \mathcal{N}(\mathbf{y}|\mathbf{0}_n, \mathbf{K}_{nn} + \sigma^2\mathbf{I}_n)$ and computing this costs $\mathcal{O}(n^3)$.

a sparse approach, we have the following approximate posterior:

$$p(\mathbf{u}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{u})p(\mathbf{u})}{\int p(\mathbf{y}|\mathbf{u})p(\mathbf{u})d\mathbf{u}}, \tag{2.12}$$

where $p(\mathbf{y}|\mathbf{u}) = \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{u})d\mathbf{f}$[3]. Again, this integral is computationally intractable due to the inversion of the $n \times n$ matrix $\tilde{\mathbf{K}}$ in $p(\mathbf{f}|\mathbf{u})$. However, this is where we can use *variational inference* (Blei et al., 2017). To obtain our *variational approximation* to $p(\mathbf{y}|\mathbf{u})$ first note that:

$$\begin{aligned} p(\mathbf{y}|\mathbf{u}) &= \frac{p(\mathbf{y}, \mathbf{f}|\mathbf{u})}{p(\mathbf{f}|\mathbf{y}, \mathbf{u})} \\ &= \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{u})}{p(\mathbf{f}|\mathbf{y}, \mathbf{u})}, \end{aligned} \tag{2.13}$$

where we have rearranged $p(\mathbf{f}|\mathbf{y}, \mathbf{u})$ for $p(\mathbf{y}|\mathbf{u})$. Taking logs and expectations with respect to $\mathbf{f}|\mathbf{u}$ on both sides then gives:

$$\begin{aligned} &\log p(\mathbf{y}|\mathbf{u}) = \log p(\mathbf{y}|\mathbf{f}) + \log \frac{p(\mathbf{f}|\mathbf{u})}{p(\mathbf{f}|\mathbf{y}, \mathbf{u})} \\ \implies &\log p(\mathbf{y}|\mathbf{u}) = \mathbb{E}_{p(\mathbf{f}|\mathbf{u})}\left[\log p(\mathbf{y}|\mathbf{f})\right] + \mathbb{E}_{p(\mathbf{f}|\mathbf{u})}\left[\log \frac{p(\mathbf{f}|\mathbf{u})}{p(\mathbf{f}|\mathbf{y}, \mathbf{u})}\right] \\ &\qquad = \log \widetilde{p}(\mathbf{y}|\mathbf{u}) + \mathrm{KL}\left[p(\mathbf{f}|\mathbf{u})||p(\mathbf{f}|\mathbf{y}, \mathbf{u})\right], \end{aligned} \tag{2.14}$$

where $\log \widetilde{p}(\mathbf{y}|\mathbf{u}) \coloneqq \mathbb{E}_{p(\mathbf{f}|\mathbf{u})}[\log p(\mathbf{y}|\mathbf{f})]$ and KL denotes the Kullback-Leibler divergence. As the KL divergence is non-negative, we obtain $\log \tilde{p}(\mathbf{y}|\mathbf{u})$ as a lower bound to the log-likelihood of $\mathbf{y}|\mathbf{u}$. Exponentiating then gives $\widetilde{p}(\mathbf{y}|\mathbf{u})$ as a lower bound to $p(\mathbf{y}|\mathbf{u})$ and this gives us our desired variational approximation. Moreover, this lower bound has a closed-form expression (see Appendix A for derivation) given by:

$$\tilde{p}(\mathbf{y}|\mathbf{u}) = \prod_{i=1}^{n} \mathcal{N}(y_i|\mathbf{k}_{nm,i}\mathbf{K}_{mm}^{-1}\mathbf{u}, \sigma^2) \left[\exp(-\frac{1}{2\sigma^2}\mathrm{Tr}(\mathbf{K}_{nn} - \mathbf{K}_{nm}\mathbf{K}_{mm}^{-1}\mathbf{K}_{mn}))\right], \tag{2.15}$$

where $\mathbf{k}_{nm,i}$ is the ith row of $\mathbf{K}_{nm}$ and the exponential term is referred to as the *trace term* or the *penalty term*.

Note that the lower the KL divergence, the tighter this lower bound becomes and thus the better an approximation we will obtain. In particular, when the divergence is 0, e.g. when $\mathbf{f} = \mathbf{u}$, the penalty vanishes.

[3]Indeed, $p(\mathbf{y}|\mathbf{u}) = \int p(\mathbf{y}, \mathbf{f}|\mathbf{u})d\mathbf{f}$ and $p(\mathbf{y}, \mathbf{f}|\mathbf{u}) = p(\mathbf{y}|\mathbf{f}, \mathbf{u})p(\mathbf{f}|\mathbf{u}) = p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{u})$. Here, we have used $\mathbf{y} \perp \mathbf{u}|\mathbf{f}$ which follows from $y_i = f(x_i) + \epsilon_i$, i.e. if we know $f(x_i)$ then y_i is independent of u_i.

With this approximation, it is now possible to obtain the approximate posterior distribution in a computationally tractable manner. Indeed, computing $\tilde{p}(\mathbf{y}|\mathbf{u})$ requires at most m^2n computations (Hensman et al., 2015) and this allows for the approximate posterior (2.16) to be computed with $\mathcal{O}(m^2n)$ computational complexity. We can also obtain the approximate posterior predictive distribution from the posterior, which lets us make prediction also with $\mathcal{O}(m^2n)$ computational complexity. Similarly, the approximate ML can also be obtained from $\tilde{p}(\mathbf{y}|\mathbf{u})$, allowing for the selection of the inducing variables and kernel hyperparameters with the same computational complexity. It is clear therefore that this approach resolves the issues of computational complexity for standard GPR discussed in section 2.1.2, allowing GPR to be scaled to large datasets.

For completeness, we state the expressions for the approximate posterior, posterior predictive and ML distributions below:

$$\begin{aligned} \tilde{p}(\mathbf{u}|\mathbf{y}) &= \frac{\tilde{p}(\mathbf{y}|\mathbf{u})p(\mathbf{u})}{\int \tilde{p}(\mathbf{y}|\mathbf{u})p(\mathbf{u})d\mathbf{u}}, \\ \tilde{p}(\mathbf{y}) &= \int \tilde{p}(\mathbf{y}|\mathbf{u})p(\mathbf{u})d\mathbf{u}, \\ \tilde{p}(f_*|\mathbf{y}) &= \int p(f_*|\mathbf{u})\tilde{p}(\mathbf{u}|\mathbf{y})d\mathbf{u} \end{aligned} \tag{2.16}$$

where f_* represents the latent function value for a test case (x_*, y_*), and $\tilde{p}(f_*|\mathbf{y})$ is used to compute the approximate posterior predictive distribution of y_*.

Remark 1. Note that our penalty term is independent of $\mathbf{u}$ and thus cancels out in the calculation of the approximate posterior.

2.2 Gaussian Process Classification

In this section we will explore the application of GPs to classification, in particular to binary classification. We will first look at logistic regression as our basic building block, then introduce three different approaches to Gaussian process classification (GPC): a sparse approach with variational inference, an approach with only variational inference, and an approach with Laplace approximations. Our main focus will be on the sparse approach with variational inference which allows GPC to be scaled to large datasets - datasets with potentially millions of data points - and achieve competitive model performance.

2.2.1 Classification and Logistic Regression

Classification

Besides regression, another large class of problems is classification, where the response variable, y, is discrete. The aim of classification is to assign classes $\mathscr{C}_1, \ldots, \mathscr{C}_N$, $N \geq 2$, to each covariate pattern $\mathbf{x}$. In this dissertation, we will focus on binary classification problems, where $N = 2$, although it is possible to extend GPC to multi-class problems ($N > 2$) also (Rasmussen and Williams, 2006, sec. 3.5).

We will be providing a *probabilistic* treatment to classification problems, where predictions takes the form of class probabilities. This differs from methods which provide only a guess at the class label, such as nearest-neighbours methods, though it is possible to also make guesses for probabilistic methods.

There are two main approaches to probabilistic classification: *discriminative* and *generative*. Both of these rely on the decomposition of the joint distribution $p(y, \mathbf{x})$; the latter, using Bayes' theorem, writes the joint distribution as $p(y)p(\mathbf{x}|y)$ and the former as $p(\mathbf{x})p(y|\mathbf{x})$. It is necessary to model the distributions $p(\mathbf{x}|y)$ or $p(y|\mathbf{x})$, depending on which approach we take. For the generative approach, we first model $p(\mathbf{x}|y)$ and $p(y)$ for $y = \mathscr{C}_1, \ldots, \mathscr{C}_N$, then calculate $p(y|\mathbf{x})$ for each class with the rule

$$p(y|\mathbf{x}) = \frac{p(y)p(\mathbf{x}|y)}{\sum_{i=1}^{N} p(y = \mathscr{C}_i)p(\mathbf{x}|y = \mathscr{C}_i)}. \tag{2.17}$$

Throughout this dissertation we will focus on the discriminative approach, where we model $p(y|\mathbf{x})$ directly through either parametric (logistic regression) or non-parametric approaches (GPC).

Prediction and Evaluation of Classification Models

With the probabilistic framework, prediction for the class probabilities of a test point $\mathbf{x}_*$ can be carried out routinely by substituting $\mathbf{x}_*$ into $p(y|\mathbf{x})$ after model training. For binary problems, prediction of the corresponding class y_* then requires selecting a cutoff threshold on which to assign classes. For example, a threshold of 0.5 would result in an assignment of class $\mathscr{C}_2$ if $p(y_* = \mathscr{C}_2|\mathbf{x}_*) \geq 0.5$ and class $\mathscr{C}_1$ otherwise.

There are a variety of approaches for selecting the optimal threshold. A default threshold of 0.5 is normally used and is sufficient for most classification problems. Throughout this dissertation, we will use a default threshold of 0.5 unless specified otherwise.

There are also various ways to quantify and evaluate the performance of classification models. In this dissertation, we will mostly use the *Correct Classification Rate* (CCR). The CCR for a classification model on a given set of observations is defined as:

$$CCR \coloneqq \frac{\#\text{ correct classifications}}{\#\text{ correct classifications} + \#\text{ incorrect classifications}}, \tag{2.18}$$

where *correct classifications* are the class predictions that match the actual classes, and *incorrect classifications* are class predictions that do not match the actual classes.

We will also refer to the *Receiver Operator Characteristic* (ROC) curve and the *Area-Under-the-Curve* (AUC) score for model evaluation. A full treatment of these concepts can be found in Majnik and Bosnić (2011).

Logistic Regression

Consider the standard binary classification problem of Bernoulli distributed responses y_i taking labels in $\{0, 1\}$, and covariate patterns $\mathbf{x}_i = (x_{1i}, \ldots, x_{ki})^\top$, $i = 1, \ldots, n$, $k, n \in \mathbb{N}$. This will be the setting for the remaining sections of this chapter. For *logistic regression* (also referred to as *linear logistic regression*), we model the likelihood as a linear function of weights $\mathbf{w} = (w_1, \ldots, w_k)^\top$ "squashed" into $[0, 1]$ through the *logistic function* σ:

$$p(y_i = 1|\mathbf{w}, \mathbf{x}_i) = \sigma(\mathbf{x}_i^\top \mathbf{w}), \tag{2.19}$$

where σ is

$$\sigma(u) = \frac{1}{1 + \exp(-u)}. \tag{2.20}$$

Let $\pi_i = p(y_i = 1|\mathbf{w}, \mathbf{x}_i)$. Noting that the y_is are Bernoulli distributed, we obtain the following log-likelihood (Czepiel, 2002):

$$\ell(\mathbf{y}) = \sum_{i=1}^{n} (y_i \log \pi_i + (1 - y_i) \log (1 - \pi_i)), \tag{2.21}$$

where $\mathbf{X}$ is the design matrix $\mathbf{X} = \begin{bmatrix}\mathbf{x}_1 \cdots \mathbf{x}_n\end{bmatrix}$ and $\mathbf{y} = (y_1, \ldots, y_n)^\top$. Due to the non-Gaussian likelihood, the maxima of ℓ has no closed-form solution, creating the necessity for approximation methods. One popular approximation for solving the above likelihood is *Iteratively Reweighted Least Squares* (IRLS) - an application of the Newton-Rhapson method. A full treatment of IRLS can be found in Green (1984).

Although logistic regression provides a simple framework for classification problems, it also has some drawbacks. One significant drawback occurs when our dataset is *linearly separable*; that is, when there exists a hyperplane which perfectly discriminates between the two classes. In this scenario, maximizing the log-likelihood can be unstable and result in very large values of $||\mathbf{w}||$. One common remedy for this problem is to add a penalty term to the log-likelihood which results in *penalized logistic regression*. In this dissertation, we will focus on standard logistic regression.

2.2.2 GPC as a Generalization of Logistic Regression

In the previous section, we saw the frequentist approach to logistic regression and this is the approach we have took for chapter 3. However, it is also possible to provide a Bayesian treatment by placing a prior on $\mathbf{w}$. Indeed, one can place a Gaussian prior on $\mathbf{w}$, where the object of interest is the posterior $p(\mathbf{w}|\mathbf{X}, \mathbf{y})$ which is used for prediction.

A natural generalization of logistic regression in the Bayesian setting is to relax the parametric form of $\sigma^{-1}(p(y_i = 1|\mathbf{w}, \mathbf{x}_i))$, where σ^{-1} inverse of the logistic function[4]. More specifically, one can replace $\mathbf{x}_i^\top \mathbf{w}$ with a latent function $f(\mathbf{x}_i)$, and instead of giving a Gaussian prior to $\mathbf{w}$, place a GP prior on f. This gives the Gaussian process treatment of classification:

$$p(y_i = 1|\mathbf{x}_i) = \sigma(f(\mathbf{x}_i)). \tag{2.22}$$

Like in logistic regression, a Bernoulli likelihood is placed on the transformed values $\sigma(f(\mathbf{x}_i))$ and we additionally assume that the y_i are independent given $f(\mathbf{x}_i)$. This gives

[4]This is also called the *logit function* or *logit link*.

the following joint distribution:

$$\begin{aligned} p(\mathbf{y}, \mathbf{f}) &= p(\mathbf{y}|\mathbf{f})p(\mathbf{f}) \\ &= \left[\prod_{i=1}^{n} p(y_i|f_i)\right] p(\mathbf{f}) \\ &= \left[\prod_{i=1}^{n} \mathscr{B}(y_i|\sigma(f_i))\right] \mathscr{N}(\mathbf{f}|\mathbf{0}_n, \mathbf{K}_{nn}) \\ &= \left[\prod_{i=1}^{n} \sigma(f_i)^{y_i}(1-\sigma(f_i))^{1-y_i}\right] \mathscr{N}(\mathbf{f}|\mathbf{0}_n, \mathbf{K}_{nn}), \end{aligned} \tag{2.23}$$

where $f_i = f(\mathbf{x}_i)$, $\mathbf{f} = (f_1, \ldots, f_n)^\top$, and $\mathscr{B}$ denotes the Bernoulli distribution. Here, we have again, w.l.o.g., assumed a mean of zero for the GP prior.

Moreover, in order to select the kernel hyperparameters, we must optimize the ML, which can be obtained by marginalizing the joint distribution:

$$\begin{aligned} p(\mathbf{y}) &= \int p(\mathbf{y}, \mathbf{f}) d\mathbf{f} \\ &= \int \left[\prod_{i=1}^{n} \sigma(f_i)^{y_i}(1-\sigma(f_i))^{1-y_i}\right] \mathscr{N}(\mathbf{f}|\mathbf{0}_n, \mathbf{K}_{nn}) d\mathbf{f} \end{aligned} \tag{2.24}$$

Due to the non-Gaussian likelihood, we cannot obtain a closed-form expression for (2.24), and so computation of the ML becomes analytically intractable. Similarly, the posterior is also analytically intractable because of the non-conjugate Gaussian prior induced by the GP. As a result, hyperparameter selection and inference (in particular prediction) become analytically intractable, creating the necessity for approximate methods.

A number of methods have been proposed to deal with the intractability of inference and hyperparameter selection in GPC. Nickisch and Rasmussen (2008) provide an excellent comparison of some methods, and Rasmussen and Williams (2006) offer an in-depth treatment of some approximation schemes. Unfortunately, the majority of these methods have $\mathscr{O}(n^3)$ in computational complexity and so are not scalable to large datasets. There have, however, been some advancements in addressing the scalability of approximate GPC through sparse methods, such as with the IVM (Lawrence et al., 2002) and Generalized FITC (Naish-Guzman and Holden, 2007). In this dissertation, we will focus on a sparse approach with variational inference (Hensman et al., 2015) that is both scalable and achieves competitive model performance. We will compare this approach with logistic regression as well as two other approximate GPC methods that do *not* make any sparse approximations.

2.2.3 Sparse GPC with Variational Inference

Here, we will present a sparse approach with variational inference for approximate GPC (Hensman et al., 2015). Let $\mathbf{u} = (u_1, \ldots, u_m)^\top$ be a vector of inducing variables obtained as latent realisations of inducing inputs $\mathbf{Z}_m = \begin{bmatrix}\mathbf{z}_1 & \ldots & \mathbf{z}_m\end{bmatrix}$, where each $\mathbf{z}_i$ is a k-vector and $m < n$. In order to derive sparse *and* variational approximations for both the ML and posterior $p(\mathbf{f}|\mathbf{y})$ we return to the expressions derived in section 2.1.3. In particular, recall from (2.14) that $\mathbb{E}_{p(\mathbf{f}|\mathbf{u})}[\log p(\mathbf{y}|\mathbf{f})]$ bounds $\log p(\mathbf{y}|\mathbf{u})$, i.e.

$$\log p(\mathbf{y}|\mathbf{u}) \geq \mathbb{E}_{p(\mathbf{f}|\mathbf{u})}[\log p(\mathbf{y}|\mathbf{f})]. \tag{2.25}$$

Additionally, we have the variational equation (see Appendix A for derivation):

$$\log p(\mathbf{y}) \geq \mathbb{E}_{\mathbf{q}(\mathbf{u})}[\log p(\mathbf{y}|\mathbf{u})] - \mathrm{KL}[q(\mathbf{u})||p(\mathbf{u})], \tag{2.26}$$

where $q(\mathbf{u})$ is some well-defined variational distribution. Then substituting (2.25) into (2.26) gives:

$$\begin{aligned}\log p(\mathbf{y}) &\geq \mathbb{E}_{\mathbf{q}(\mathbf{u})}[\log p(\mathbf{y}|\mathbf{u})] - \mathrm{KL}[q(\mathbf{u})||p(\mathbf{u})] \\ &\geq \mathbb{E}_{\mathbf{q}(\mathbf{u})}[\mathbb{E}_{p(\mathbf{f}|\mathbf{u})}[\log p(\mathbf{y}|\mathbf{f})]] - \mathrm{KL}[q(\mathbf{u})||p(\mathbf{u})] \\ &\geq \mathbb{E}_{\mathbf{q}(\mathbf{f})}[\log p(\mathbf{y}|\mathbf{f})] - \mathrm{KL}[q(\mathbf{u})||p(\mathbf{u})],\end{aligned} \tag{2.27}$$

where we have defined $q(\mathbf{f}) \coloneqq \int p(\mathbf{f}|\mathbf{u})q(\mathbf{u})d\mathbf{u}$.

With variational inference we want to choose $q(\mathbf{u})$ such that the expectations in (2.27) are tractable. Choosing $q(\mathbf{u})$ to be $\mathcal{N}(\mathbf{u}|\mathbf{m}, \mathbf{S})$, where $\mathbf{m}$ and $\mathbf{S}$ are free parameters of dimensions $m \times 1$ and $m \times m$ respectively, gives a closed-form expression for $q(\mathbf{f})$:

$$q(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{A}\mathbf{m}, \mathbf{K}_{nn} + \mathbf{A}(\mathbf{S} - \mathbf{K}_{mm})\mathbf{A}^\top), \tag{2.28}$$

where $\mathbf{A} = \mathbf{K}_{nm}\mathbf{K}_{mm}^{-1}$. Moreover, as our likelihood can be factored into the product of MLs, i.e $p(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^n p(y_i|f_i)$, then only the marginals $q(f_i)$ are required for the expectations in (2.27). Thus (2.27) can be simplified to

$$\log p(\mathbf{y}) \geq \sum_{i=1}^{n} \mathbb{E}_{q(f_i)}[\log p(y_i|f_i)] - \mathrm{KL}[q(\mathbf{u})||p(\mathbf{u})], \tag{2.29}$$

These marginal expectations have the form $\int_{-\infty}^{\infty} f(x)e^{-x^2}dx$ and can be tractably computed via numerical methods such as Gaussian-Hermite quadrature (Liu and Pierce, 1994).

This gives us the desired approximation to the ML, allowing for selection of the kernel hyperparameters, $\mathbf{Z}_m$, $\mathbf{m}$ and $\mathbf{S}$ via gradient-based optimization methods. This selection has $\mathcal{O}(m^3)$ computational complexity.

After having selected our parameters, it is also possible to carry out prediction in the normal Bayesian manner. Indeed, our approximate posterior is given by $p(\mathbf{f}|\mathbf{y}) \approx q(\mathbf{u}, \mathbf{f}) = p(\mathbf{f}|\mathbf{u})q(\mathbf{u})$ and can be used to make class predictions with $\mathcal{O}(m^2)$ computational complexity.

The computational complexities for prediction and parameter selection here are an improvement on the $\mathcal{O}(n^3)$ computational complexity of previous approximate GPC methods. This results in this approach being scalable to large datasets. We show in chapter 3 that this approach also achieves competitive model performance.

2.2.4 GPC with Variational Inference

It is also possible to carry out approximate GPC with variational inference only, i.e. without sparse approximations. Unsurprisingly, this approach achieves worse computational complexity than the sparse approach, requiring at most n^3 computations for parameter selection. We derive this approach from the previous section.

Simply letting $\mathbf{Z}_m = \mathbf{X}$ in the previous approach arrives at approximate GPC with variational inference only. Here, we have set the inducing inputs equal to the design matrix for the entire dataset and thus have $m = n$. Consequently, the computational complexity for parameter selection and prediction becomes $\mathcal{O}(n^3)$ and $\mathcal{O}(n^2)$ respectively. As we would expect, this approach is not scalable to large datasets, making the sparse approach more favourable.

Remark 2. This approach is the same as the "KL" method described in Nickisch and Rasmussen (2008).

2.2.5 Laplace Approximations for GPC

The variational formulation of GPC is a relatively recent development of approximate GPC. An older formulation of approximate GPC is the treatment given by Laplace approximations (also called the *Laplace method*) in Williams and Barber (1998). Here, we are not seeking for sparse approximations, but rather for approximate classification only.

This is done through approximating the ML and posterior by Taylor series approximations, consequently allowing for hyperparameter selection and prediction.

In the Laplace method, we first make a second-order Taylor series approximation to $\log p(\mathbf{f}|\mathbf{X}, \mathbf{y})$ around the maximum of the posterior:

$$\begin{aligned} \log p(\mathbf{f}|\mathbf{X}, \mathbf{y}) &\approx \log p(\hat{\mathbf{f}}|\mathbf{X}, \mathbf{y}) + \mathbf{g}^\top(\mathbf{f} - \hat{\mathbf{f}}) + \frac{1}{2}(\mathbf{f} - \hat{\mathbf{f}})^\top \mathbf{H}(\mathbf{f} - \hat{\mathbf{f}}) \\ &= \log p(\hat{\mathbf{f}}|\mathbf{X}, \mathbf{y}) + \frac{1}{2}(\mathbf{f} - \hat{\mathbf{f}})^\top \mathbf{H}(\mathbf{f} - \hat{\mathbf{f}}), \end{aligned} \tag{2.30}$$

where $\hat{\mathbf{f}}$ is the maximum of the posterior, $\mathbf{g} = \nabla(\log p(\mathbf{f}|\mathbf{X}, \mathbf{y}))|_{\mathbf{f}=\hat{\mathbf{f}}} = \mathbf{0}_n$ is the gradient of the log posterior at $\hat{\mathbf{f}}$, and $\mathbf{H} = \nabla\nabla(\log p(\mathbf{f}|\mathbf{X}, \mathbf{y}))|_{\mathbf{f}=\hat{\mathbf{f}}}$ is the Hessian of the log posterior at $\hat{\mathbf{f}}$. Noting that (2.30) has the form for the (un-normalized) log of a multivariate normal density, we then obtain the following approximation to the posterior:

$$q(\mathbf{f}|\mathbf{X}, \mathbf{y}) = \mathcal{N}(\mathbf{f}|\hat{\mathbf{f}}, \mathbf{A}^{-1}), \tag{2.31}$$

where $\mathbf{A} = -\mathbf{H}$.

It is possible to show that $\hat{\mathbf{f}} = \mathbf{K}_{nn}(\nabla \log p(\mathbf{y}|\hat{\mathbf{f}}))$ and $\mathbf{A} = \mathbf{K}_{nn}^{-1} + \mathbf{W}$, where $\mathbf{W} = -\nabla\nabla(\log p(\mathbf{y}|\mathbf{f}))|_{\mathbf{f}=\hat{\mathbf{f}}}$ is a diagonal matrix due to the factorization in the likelihood. However, as $\nabla \log p(\mathbf{y}|\hat{\mathbf{f}})$ is a non-linear function of $\hat{\mathbf{f}}$ it is not possible to find an explicit expression for $\hat{\mathbf{f}}$. Instead, approximate methods are used to find $\hat{\mathbf{f}}$ and thus $\mathbf{W}$, with Newton's method being the most common. Full details of these derivations and the application of Newton's method can be found in Rasmussen and Williams (2006, sec. 3.4.1).

Subsequently, with the approximate posterior it is then possible to make class predictions. However, due to the inversion of the $n \times n$ matrix $\mathbf{A}$ prediction costs $\mathcal{O}(n^3)$, although it can be quicker in practise via algebraic simplifications and fast algorithmic solvers.

For hyperparameter selection, it is necessary to approximate the ML. This can be similarly achieved with the Laplace method demonstrated above. Indeed, we can write the ML as

$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f})d\mathbf{f} = \int \exp \phi(\mathbf{f})d\mathbf{f}, \tag{2.32}$$

where $\phi(\mathbf{f}) = \log[p(\mathbf{y}|\mathbf{f})p(\mathbf{f})]$. Through a second-order Taylor series expansion of $\phi(\mathbf{f})$ around $\hat{\mathbf{f}}$ it is possible to analytically evaluate the above integral and arrive at the following approximation, q', to the marginal log-likelihood:

$$\log p(\mathbf{y}) \approx \log q'(\mathbf{y}) = -\frac{1}{2}\hat{\mathbf{f}}^\top \mathbf{K}_{nn}^{-1}\hat{\mathbf{f}} + \log p(\mathbf{y}|\hat{\mathbf{f}}) - \frac{1}{2}\log|\mathbf{B}|, \tag{2.33}$$

where $\mathbf{B} = \mathbf{I}_n + \mathbf{W}^{\frac{1}{2}}\mathbf{K}_{nn}\mathbf{W}^{\frac{1}{2}}$, and $\mathbf{W}^{\frac{1}{2}}$ is the unique positive definite matrix $\mathbf{M}$ satisfying $\mathbf{M}^2 = \mathbf{W}$[5]. Note that this marginal log-likelihood is dependent on the vector of hyperparameters $\boldsymbol{\theta}$ through $\mathbf{K}_{nn}$ and $\mathbf{B}$. The hyperparameters can then be selected by maximizing (2.33) w.r.t the vector of hyperparameters. Again, as a result of having to invert $\mathbf{K}_{nn}$, hyperparameter selection costs $\mathcal{O}(n^3)$.

Laplace approximations allow us to carry out approximate GPC in a relatively simple manner. However, hyperparameter selection and prediction have $\mathcal{O}(n^3)$ computational complexity, prohibiting it from being scaled to large datasets. This again makes the sparse approach with variational inference preferable, although it should be noted that the Laplace approach can perform quicker in practise (Nickisch and Rasmussen, 2008, p. 2057) through the use of algebraic techniques (Rasmussen and Williams, 2006, sec. 3.4.3) and fast algorithmic solvers.

[5]Existence and uniqueness is guaranteed by $\mathbf{W}$ being a (diagonal) positive definite matrix. That $\mathbf{W}$ is positive definite can be easily seen from its diagonal elements which are $-\frac{\partial^2}{\partial f_i^2}\log p(\mathbf{y}|\mathbf{f}) = \frac{\exp(f_i)}{(1+\exp(f_i))^2} > 0, \forall f_i \in \mathbb{R}$.

Chapter 3

Experiments

In this chapter we will demonstrate the scalability and competitive model performance of the sparse GPC approach with variational inference (section 2.2.3). We will do this through applying and comparing the classification models discussed in chapter 2 on two simulated datasets and one real dataset. More specifically, we will apply the models: Logistic regression (LR), sparse GPC with variational inference (SVGP), GPC with variational inference (FVGP), GPC with Laplace approximations (LGP). All the Gaussian process models were implemented via the package GPy (2012) developed by the Sheffield Machine Learning group and the logistic regression models were implemented via the *statsmodel* package (Seabold and Perktold, 2010) in Python. The experiments were ran on an Intel i7 core with 12 GB RAM and the associated Jupyter notebooks can be viewed here.

3.1 Simulations

3.1.1 Simulation I

Data Simulation

In this simulation, we will demonstrate the competitive performance and scalability of SVGP for the case where we have a 1-D dataset and varying data sizes. For the simulation, n data points x_i, where $n = 150, 500, 1000, 1500$, were randomly drawn from the uniform distribution:

$$x_i \sim U(0, 1), \quad i = 1, \ldots, n \tag{3.1}$$

These were then scaled by a constant $c = 11$ and applied the latent function $f(x) = 3(\sin^4 x + \cos^3 x)$:

$$\begin{aligned} x_i &\longmapsto cx_i, \\ f(cx_i) &\longmapsto 3(\sin^4(cx_i) + \cos^3(cx_i)). \end{aligned} \tag{3.2}$$

The latent function values $f_i = f(cx_i)$ in this case allow us to simulate the latent probabilities for the x_is. In order to transform the f_is into latent probabilities the logistic function, σ, was used:

$$p_i = \frac{1}{1 + \exp(-f_i)} \in [0, 1]\,. \tag{3.3}$$

Finally, to assign Bernoulli draws to the x_is, each p_i was compared to a random draw from $U(0, 1)$, and where p_i was greater than the random draw, the class 1 was assigned, and 0 otherwise.

A train-test split of 75-25% was applied to all the datasets. A plot of the simulated latent training probabilities and training Bernoulli draws for the largest dataset ($n = 1500$) is displayed below:

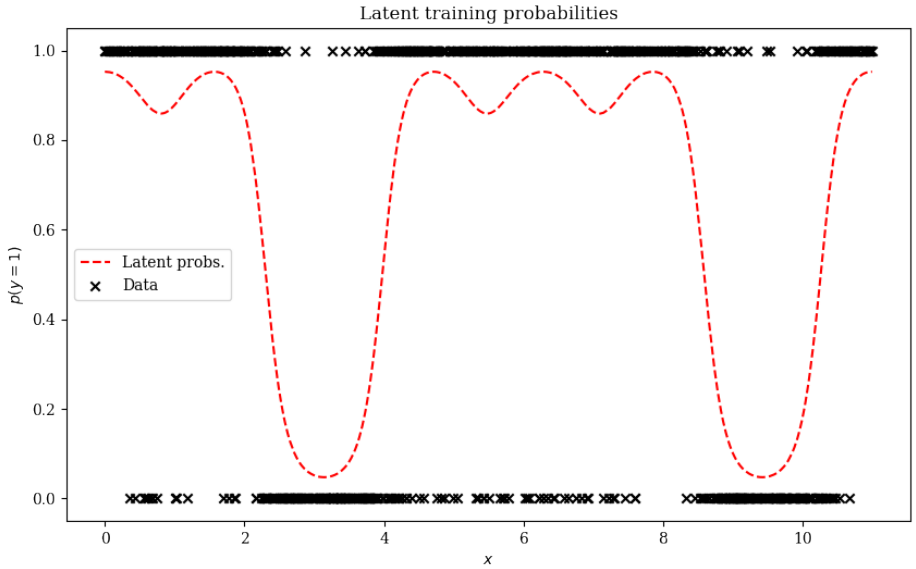


Figure 3.1: Simulated latent training probabilities and training Bernoulli draws for $n = 1500$.

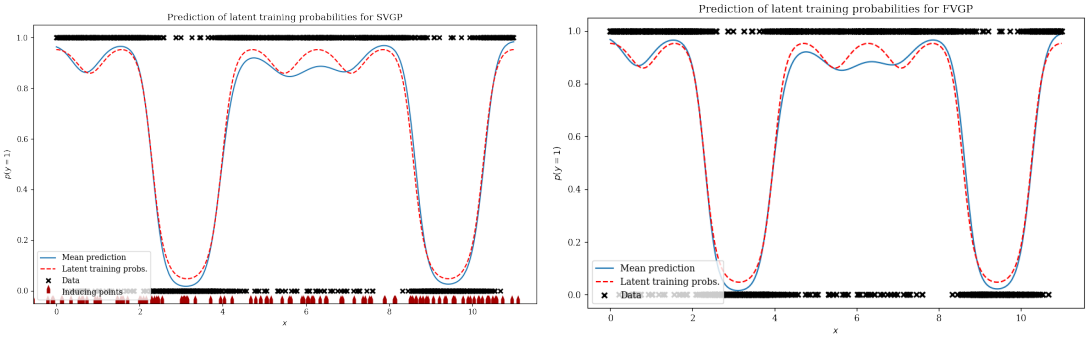
Models and Results

The models LR, SVGP, FVGP, and LGP were all fit on the training set. For SVGP, the inducing inputs were randomly initialized to be a subset of the training set, with the number of inducing points set to 10% the training set. The kernel used for the GP models was a sum of the RBF and white noise (WN) kernels:

$$k(x, x') = \alpha \exp\left(-\frac{(x - x')^2}{2l^2}\right) + \beta\delta(x - x'), \tag{3.4}$$

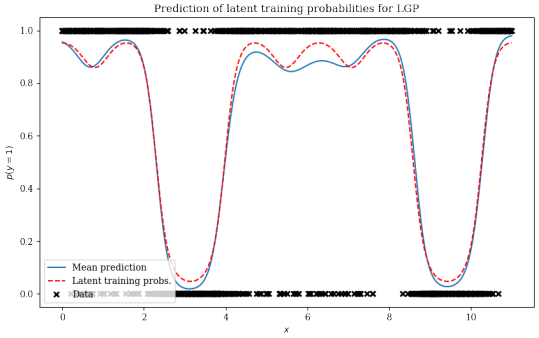
where $x, x' \in \mathbb{R}$, δ is the Dirac-delta function and $\beta \in \mathbb{R}$ is a kernel hyperparameter.

A plot of the mean predicted probabilities for the latent training probabilities is displayed below. For brevity, we have included the predictions for the largest dataset only; we provide a comparison of model performance for different data sizes in the rest of this section.

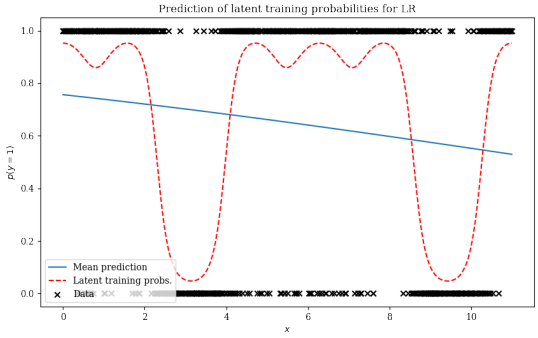


(a) SVGP

(b) FVGP



(c) LGP

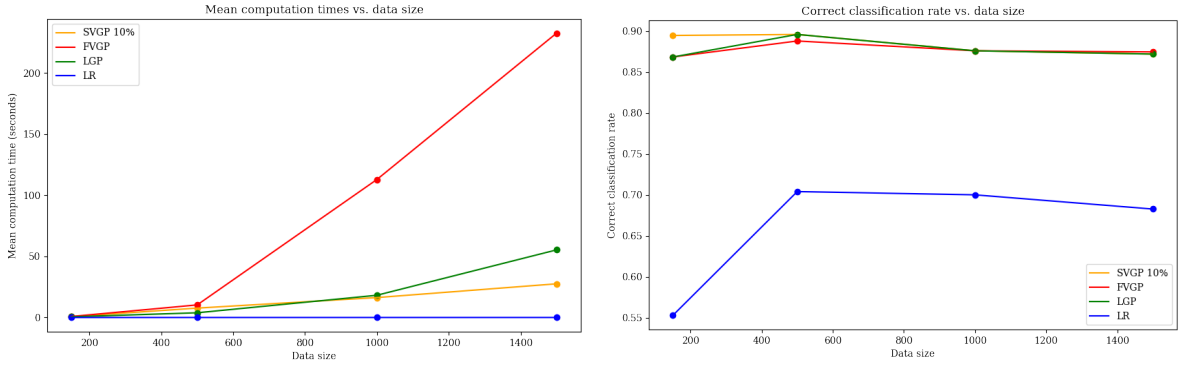


(d) LR

Figure 3.2: Mean predictions for the latent training probabilities ($n = 1500$).

The optimized location of the inducing points for SVGP can be seen from Figure 3.2a. Moreover, we can see that LR provided the worst fit for the latent training probabilities, missing out on the curvature of the dataset. This can be explained by LR fitting a linear function for the latent function values. The GP models, on the other hand, all gave a very good fit for the latent training probabilities, capturing the curvature and peaks and troughs of the dataset very well.

To evaluate the performance of the models, we look at plots of the mean computation time and CRR against data size:



(a) Computation time vs. data size

(b) CRR vs. data size

Figure 3.3: Plots of the mean computation time (seconds) for optimizing the ML vs. data size (a) and CRRs for the test set vs. data size (b). The computation times were averaged over 5 runs.

It is clear that computation wise LR performed the best, with the computation time remaining approximately 0 seconds for all the data sizes. In contrast, FVGP had the worst computation times, with the computation time for the largest dataset being approximately 5 minutes. This is explained by the computational complexity of FVGP which is $\mathcal{O}(n^3)$. Moreover, for the GP models, we see that SVGP achieved the best computation times for the largest datasets, with its computation times increasing slowly with data size. This suggests, as we would expect from section 2.2.3, that SVGP is indeed scalable to large datasets.

Furthermore, Figure 3.3b shows that LR achieved the worst CRRs for all the data sizes, as expected from its fit on the training set. The GP models, however, all achieved very high CCRs between 0.86 and 0.90 for all the data sizes. In particular, this suggests that SVGP also achieves competitive model performance, both in comparison to GP models as well as standard parametric approaches like LR.

Remark 3. Note that LGP's computation times were only marginally different from that of SVGP's, despite the $\mathcal{O}(n^3)$ computational complexity. This is explained by the comment at the end of page 15.

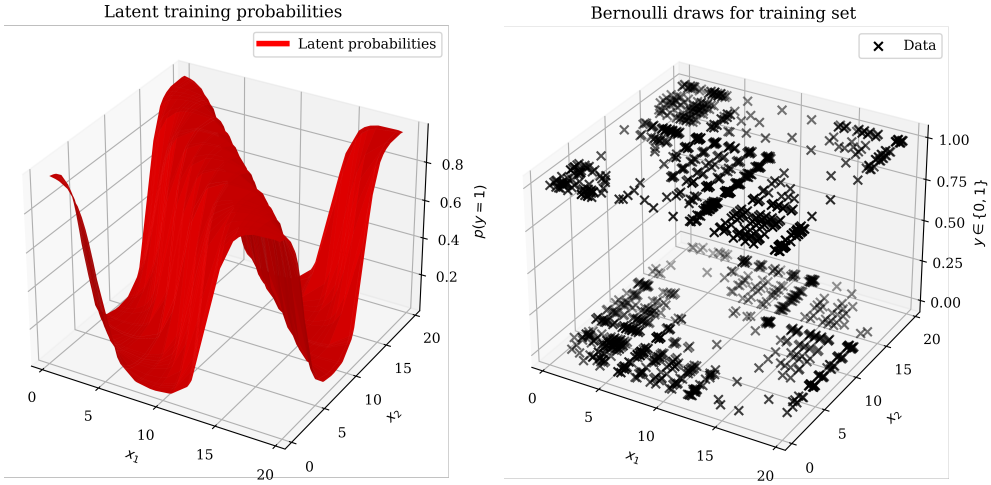
3.1.2 Simulation II

Data Simulation

In this simulation, we will further highlight the scalability and competitive performance of SVGP for a 2-D dataset. The setup is the same as in simulation I, where we fix the data size to be $n = 1500$ and set $c = 21$. As we have a 2-D dataset, two covariates x_{i1}, x_{i2} were simulated like in (3.1). The latent function chosen for this simulation was:

$$f(x_1, x_2) = \frac{4\sin(x_1 + x_2)}{3} \tag{3.5}$$

A train-test split of 75-25% was again taken. A plot of the simulated latent training probabilities and training Bernoulli draws is displayed below:

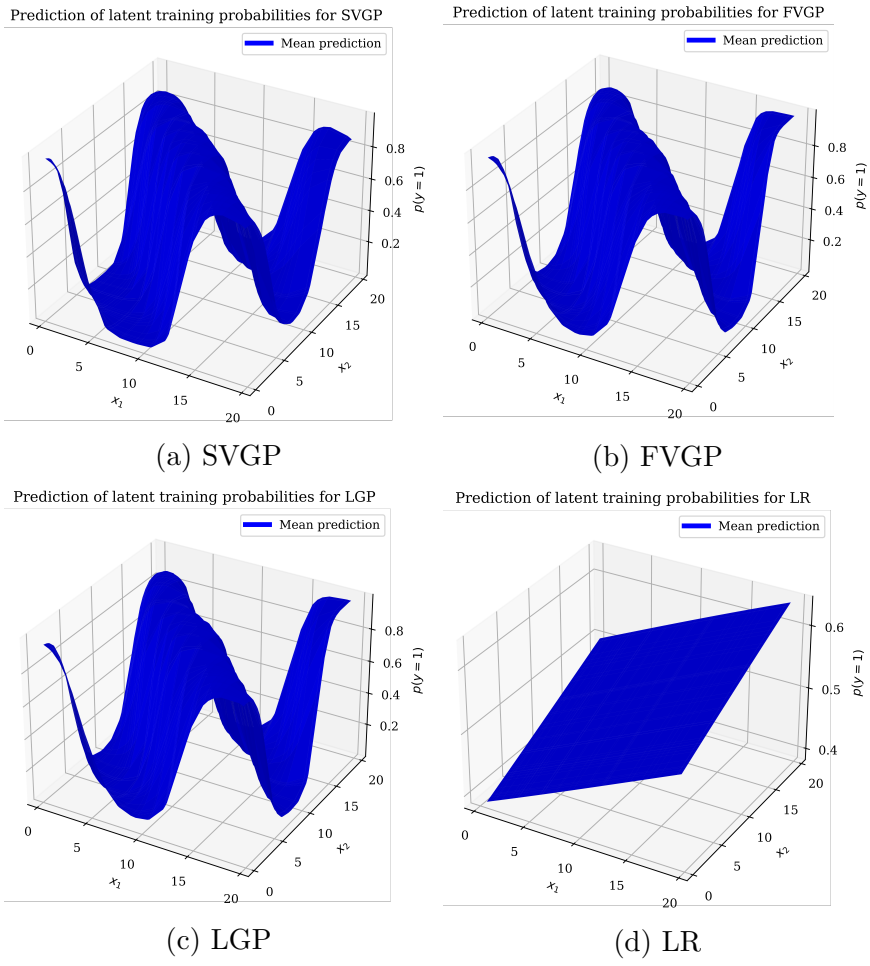


(a) Latent training probabilities (b) Training Bernoulli draws

Figure 3.4: Plots of the latent training probabilities (a) and training Bernoulli draws (b) for the latent function $f(x_1, x_2)$.

Models and Results

The same models as in the previous simulation were used. For the GP models, the same kernel (in 2-D) was used. The inducing points were initialized in the same manner as before and chosen to be 10% the training size. A plot of the mean predicted probabilities for the simulated latent training probabilities is displayed below:



(a) SVGP (b) FVGP

(c) LGP (d) LR

Figure 3.5: Mean predictions for the latent training probabilities.

We can see that LR again provides the worst fit for the latent training probabilities, missing out on the curvature of the dataset. This can be explained like in simulation I. The

GP models, however, all again gave a very good fit for the latent training probabilities, capturing the curvature and peaks and troughs of the dataset very well.

To assess the performance of the models, we look at the table of the mean computation times and CRRs:

	Mean computation time (seconds)	CCRs
SVGP	122.36	0.85
FVGP	998.40	0.89
LGP	140.54	0.89
LR	≈ 0.00	0.52

Table 3.1: CCRs for the test set and mean computation times (seconds) for optimization of the ML. The computation times were averaged over 5 runs.

From Table 3.1, we see that computation wise LR again performed the best, with the computation time being approximately 0 seconds. Similar to before, FVGP had the longest computation time ($\approx$ 17 minutes), as expected from its computational complexity. We also see that SVGP achieved the best computation time amongst the GP models, further indicating that SVGP is indeed a scalable approach, even for higher dimensions. (LGP's similar computation time here is explained by remark 3.)

Like before, LR achieved the worst CRR, as expected from its fit on the training set. The GP models, however, all achieved high CCRs between 0.85 and 0.89. This further suggests that SVGP also achieves competitive model performance.

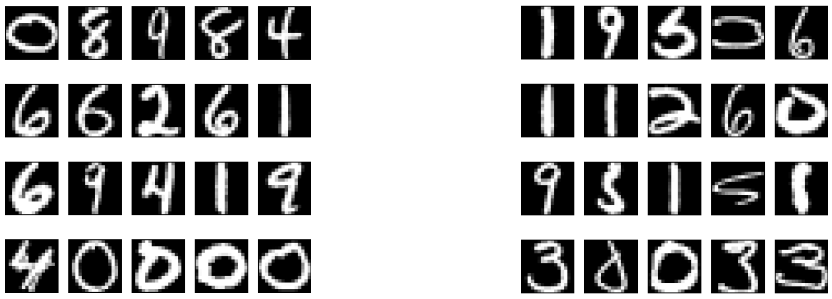
3.2 Handwritten Digits Classification

Handwritten digits recognition is a common machine learning task and is widely used for benchmarking classifiers. Here, we consider the binary classification problem of discriminating between digits that contains loops (e.g. 0 or 8) or not. This is motivated by the problem of feature extraction in handwritten recognition, where the aim is to extract features such as loops or straight lines in a digit. This type of classification problem is normally approached via neural networks (Notley and Magdon-Ismail, 2018; Jogin et al., 2018), and can also be approached directly with feature analysis techniques such as the *Hough transform* (Giuliodori et al., 2011; Ballard, 1981). Here, we approach the problem with logistic regression and the GPC methods discussed in the previous sections.

3.2.1 USPS Dataset

The dataset we consider is the U.S. Postal Service (USPS) handwritten digits[1]. The original dataset consisted of 9298 handwritten digits (7291 training, 2007 test) automatically scanned from envelopes by USPS, where the digits ranged from 0-9. As the original images were of different sizes and orientations, they were pre-processed to create 16×16 greyscale images where the intensity of each pixel lies in $[-1, 1]$ (LeCun et al., 1990).

Some exemplary images from the original test and training sets can be seen below:



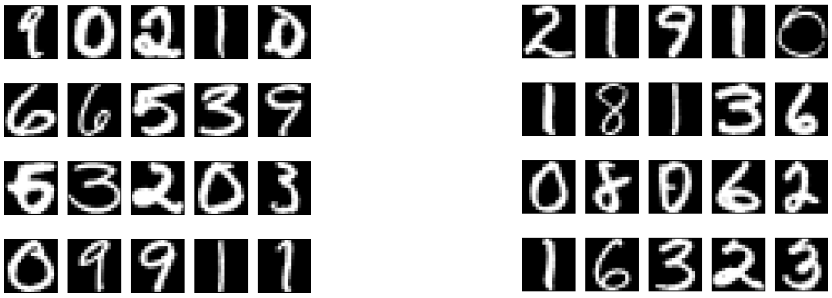
(a) Training (b) Test

Figure 3.6: Exemplary training and test images from the original USPS dataset.

[1]The dataset can be obtained from: https://web.stanford.edu/~hastie/StatLearnSparsity_files/DATA/zipcode.html

3.2.2 Data Preparation

The original test and training sets did not stem from the same distribution - the original test set contained more difficult cases than the training set, as can be observed from some of the images in Figure 3.6. This was a result of different methods of preparation for the different sets Rasmussen and Williams (2006, sec. 3.7.3). As most machine learning algorithms assume the same training and test distributions, we pooled the dataset before re-partitioning it into a train-test split of 75-25%. Some exemplary images from the re-partitioned test and training sets can be seen below:



(a) Training (b) Test

Figure 3.7: Exemplary training and test images from the re-partitioned USPS dataset.

In order to form a balanced problem, the digits 0,6,8,9 were chosen as the digits with loops, and the digits 1,2,3,5 were chosen as the digits without loops. The digit 4 was omitted as it can be written in two different ways. Moreover, the features for our dataset were taken to be the pixels for each image, resulting in 256 features. The labels for the images with and without loops (0 and 1 respectively) were taken to be our responses.

3.2.3 Models and Results

We applied the same models as for our simulations. For the GP models we specified the kernel to be the RBF kernel and took the number of inducing points for SVGP to be 10% our training size. The inducing points were initialized like for the simulations.

Table 3.2 shows the CCRs and mean computation times for the different models. Optimization of the objective function for LR did not achieve convergence, likely as a result of collinearity between our high-dimensional features (see Figure B.1). As a result, the parameters of LR could also not be estimated, leading to a lack of prediction. For these

	Mean computation time (hours)	CCRs
SVGP	2.16	0.98
FVGP	8.83	0.97
LGP	2.23	0.98
LR	–	–

Table 3.2: CCRs on the test set and mean computation times (hours) for optimization of the ML. The computation times were averaged over 2 runs.

reasons, the computation time and CCR field for LR were left blank. Already, this shows the advantage that SVGP (as well as the other GP models) has over standard parametric approaches like LR; it avoids the issues of model estimation by working with a covariance matrix $\mathbf{K}_{nn}$ rather than the design matrix which can lead to unstable computations.

Moreover, we see that SVGP achieves both the best computation time and CRR amongst all the GP models, thus giving optimal performance. This further reinforces the points that SVGP is both scalable to large datasets and achieves competitive model performance. The computation time for FVGP, as we would expect, is drastically slower (nearly 4 times slower). LGP has only slightly slower computation time (approximately 5 minutes slower) and this is again explained by remark 3.

Although we have seen that SVGP offers the optimal performance in terms of computation time and CCR, it must be noted that that all our models achieved very high CCRs. This could be caused by the issue of linear separation in our dataset and is further discussed in chapter 4. The similar performance of our models can also be seen from their ROC curves and AUC scores (see Figure B.2), where LR is omitted for the same reasons as earlier.

Chapter 4

Discussions

It is clear the SVGP offers an elegant approach to GPC that is both scalable to large datasets and also achieves competitive model performance. Through these points, we see that SVGP solves the issues of computational complexity and approximate inference in GPC.

We showed these points first through two simulated datasets and a real dataset. For our real dataset, we had 256 features, creating a high-dimensional dataset. As a result, it is very possible that our results were affected by the "curse of dimensionality" (Bellman, 1957). This could be one reason for the non-convergence of LR, as collinearity between the high-dimensional features could have rendered the optimization algorithms unstable. For the same reason it is also possible that our dataset was linearly separable, which would explain the very high accuracy of our GP models.

One way of dealing with this issue would be to reduce the dimension of our dataset before modelling. This can be done through dimension reduction techniques such as *Principal Component Analysis* (Pearson, 1901) or even *Gaussian Process Latent Variable Models* (Damianou et al., 2016). Not only could this resolve the issue of collinearity, allowing LR to be optimized, but it could also resolve the possible issue of linear separation in our dataset by projecting it to a lower dimensional space. This would then make our results and comparisons more reliable. Another way to deal with this issue, in particular for LR, would be to use penalized logistic regression, which avoids the issue of collinearity and linear separation by adding a penalty term to our likelihood.

Furthermore, as we saw through both our simulations and real dataset, LGP often has computational performance that is only marginally slower than that of SVGP. In order to make this comparison more robust one can apply our models on other, possibly larger,

datasets such as the MNIST handwritten digits datasets (LeCun et al., 1998). Applying other models, such as neural networks or other approximate schemes for GPC, could also further reinforce the competitive model performance of SVGP.

References

Ballard, D. (1981), 'Generalizing the Hough transform to detect arbitrary shapes', *Pattern Recognition* **13**(2), 111–122.

Bellman, R. (1957), *Dynamic Programming*, Princeton University Press, Princeton, New Jersey, USA.

Blei, D. M., Kucukelbir, A. and McAuliffe, J. D. (2017), 'Variational inference: A review for statisticians', *Journal of the American Statistical Association* **112**(518), 859–877.

Czepiel, S. A. (2002), 'Maximum likelihood estimation of logistic regression models: Theory and implementation', http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.454.257. Accessed: 13-10-2020.

Damianou, A. C., Titsias, M. K. and Lawrence, N. D. (2016), 'Variational inference for latent variables and uncertain inputs in Gaussian processes', *Journal of Machine Learning Research* **17**(1), 1425–1486.

Danziger, P. (2005), 'Math 108 lecture notes: Complexity of the Gaussian algorithm', https://math.ryerson.ca/~danziger/professor/MTH108/Handouts/handouts.shtml. Ryerson University, accessed: 20-12-2020.

Giuliodori, A., Lillo, R. and Pea, D. (2011), Handwritten digit classification, Technical report, Universidad Carlos III de Madrid. Departamento de Estadística.

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

Green, P. J. (1984), 'Iteratively reweighted least squares for maximum likelihood estimation, and some robust and resistant alternatives', *Journal of the Royal Statistical Society: Series B (Methodological)* **46**(2), 149–170.

Hensman, J., Matthews, A. and Ghahramani, Z. (2015), Scalable variational Gaussian process classification, *in* G. Lebanon and S. V. N. Vishwanathan, eds, 'Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics', Vol. 38 of *Proceedings of Machine Learning Research*, PMLR, San Diego, California, USA, pp. 351–360.

Jogin, M., Mohana, Madhulika, M. S., Divya, G. D., Meghana, R. K. and Apoorva, S. (2018), Feature extraction using convolution neural networks (CNN) and deep learning, *in* '2018 3rd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)', pp. 2319–2323.

Lawrence, N., Seeger, M. and Herbrich, R. (2002), Fast sparse Gaussian process methods: The Informative Vector Machine, *in* 'Proceedings of the 15th International Conference on Neural Information Processing Systems', NIPS'02, MIT Press, Cambridge, MA, USA, p. 625–632.

LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998), 'Gradient-based learning applied to document recognition', *Proceedings of the IEEE* **86**(11), 2278–2324.

LeCun, Y., Matan, O., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., Jackel, L. D. and Baird, H. S. (1990), Handwritten zip code recognition with multilayer networks, *in* 'Proceedings of the International Conference on Pattern Recognition', Vol. II, IEEE, pp. 35–40.

Liu, Q. and Pierce, D. A. (1994), 'A note on Gauss-Hermite quadrature', *Biometrika* **81**(3), 624–629.

Majnik, M. and Bosnić, Z. (2011), 'ROC analysis of classifiers in machine learning: A survey', *Intelligent Data Analysis* **17**(3), 531–558.

Naish-Guzman, A. and Holden, S. (2007), The Generalized FITC approximation, *in* 'Proceedings of the 20th International Conference on Neural Information Processing Systems', NIPS'07, Curran Associates Inc., Red Hook, NY, USA, p. 1057–1064.

Nickisch, H. and Rasmussen, C. E. (2008), 'Approximations for binary Gaussian process classification', *Journal of Machine Learning Research* **9**, 2035–2078.

Notley, S. and Magdon-Ismail, M. (2018), 'Examining the use of neural networks for feature extraction: A comparative analysis using deep learning, support vector machines, and k-nearest neighbor classifiers', *arXiv e-prints* p. arXiv:1805.02294.

Pearson, K. (1901), 'LIII. On lines and planes of closest fit to systems of points in space', *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* **2**(11), 559–572.

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

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

Seabold, S. and Perktold, J. (2010), statsmodels: Econometric and statistical modeling with Python, *in* '9th Python in Science Conference'.

Snelson, E. and Ghahramani, Z. (2005), Sparse Gaussian processes using pseudo-inputs, *in* 'Proceedings of the 18th International Conference on Neural Information Processing Systems', NIPS'05, MIT Press, Cambridge, MA, USA, p. 1257–1264.

Titsias, M. (2009), Variational learning of inducing variables in sparse Gaussian processes, *in* D. van Dyk and M. Welling, eds, 'Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics', Vol. 5 of *Proceedings of Machine Learning Research*, PMLR, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, pp. 567–574.

Titsias, M. K. (2008), Variational model selection for sparse Gaussian process regression, Technical report, University of Manchester.

Williams, C. K. I. and Barber, D. (1998), 'Bayesian classification with Gaussian processes', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **20**(12), 1342–1351.

APPENDICES

Appendix A

Mathematical Derivations

Equation 2.15. $\tilde{p}(\mathbf{y}|\mathbf{u}) = \prod_{i=1}^{n} \mathcal{N}(y_i|\mathbf{k}_{nm,i}\mathbf{K}_{mm}^{-1}\mathbf{u}, \sigma^2) \left[\exp(-\frac{1}{2\sigma^2}\mathrm{Tr}(\mathbf{K}_{nn} - \mathbf{K}_{nm}\mathbf{K}_{mm}^{-1}\mathbf{K}_{mn}))\right]$

Proof. Let the setup be as in section 2.1.3. For brevity, let $\langle\cdot\rangle_{q(\cdot)}$ denote $\mathbb{E}_{q(\cdot)}(\cdot)$. Recalling that

$$\log \tilde{p}(\mathbf{y}|\mathbf{u}) = \langle \log p(\mathbf{y}|\mathbf{f})\rangle_{p(\mathbf{f}|\mathbf{u})},$$

we have, by definition,

$$\begin{aligned}
\langle \log p(\mathbf{y}|\mathbf{f})\rangle_{p(\mathbf{f}|\mathbf{u})} &= \int p(\mathbf{f}|\mathbf{u}) \log p(\mathbf{y}|\mathbf{f}) d\mathbf{f} \\
&= \int p(\mathbf{f}|\mathbf{u}) \log \left(\prod_{i=1}^{n} p(y_i|f_i)\right) d\mathbf{f} \\
&= \int p(\mathbf{f}|\mathbf{u}) \left(\sum_{i=1}^{n} \log p(y_i|f_i)\right) d\mathbf{f},
\end{aligned}$$

where $f_i = f(x_i)$. Noting that $p(y_i|f_i) \sim \mathcal{N}(y_i|f_i, \sigma^2)$, we then have

$$\begin{aligned}
\sum_{i=1}^{n} \log p(y_i|f_i) &= \sum_{i=1}^{n} \left[-\frac{1}{2}\log\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}\left(y_i - f_i\right)^2\right] \\
&= -\frac{n}{2}\log\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}\left[y_i^2 - 2y_i f_i + f_i^2\right] \\
&= -\frac{n}{2}\log\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}(\mathbf{y}^\top\mathbf{y} - 2\mathbf{y}^\top\mathbf{f} + \mathbf{f}^\top\mathbf{f}).
\end{aligned}$$

In order to simplify the computations, we will write the previous expression, like in Titsias (2008, Appendix A), as

$$\sum_{i=1}^{n} \log p(y_i|f_i) = -\frac{n}{2}\log\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}\mathrm{Tr}(\mathbf{y}\mathbf{y}^\top - 2\mathbf{y}\mathbf{f}^\top + \mathbf{f}\mathbf{f}^\top),$$

which then gives

$$\begin{aligned}
\langle \log p(\mathbf{y}|\mathbf{f})\rangle_{p(\mathbf{f}|\mathbf{u})} &= \int p(\mathbf{f}|\mathbf{u})\left(-\frac{n}{2}\log\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}\mathrm{Tr}(\mathbf{y}\mathbf{y}^\top - 2\mathbf{y}\mathbf{f}^\top + \mathbf{f}\mathbf{f}^\top)\right) d\mathbf{f} \\
&= -\frac{n}{2}\log\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}\int p(\mathbf{f}|\mathbf{u})\mathrm{Tr}(\mathbf{y}\mathbf{y}^\top - 2\mathbf{y}\mathbf{f}^\top + \mathbf{f}\mathbf{f}^\top) d\mathbf{f} \\
&= -\frac{n}{2}\log\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}\mathrm{Tr}(\mathbf{y}\mathbf{y}^\top) + \frac{1}{\sigma^2}\langle\mathrm{Tr}(\mathbf{y}\mathbf{f}^\top)\rangle_{p(\mathbf{f}|\mathbf{u})} - \frac{1}{2\sigma^2}\langle\mathrm{Tr}(\mathbf{f}\mathbf{f}^\top)\rangle_{p(\mathbf{f}|\mathbf{u})},
\end{aligned}$$

where the last line follows from the linearity of $\mathrm{Tr}(\cdot)$. Furthermore, we can use the linearity of expectations[1] to write the above as

$$\begin{aligned}
\langle \log p(\mathbf{y}|\mathbf{f})\rangle_{p(\mathbf{f}|\mathbf{u})} &= -\frac{n}{2}\log\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}\mathrm{Tr}(\mathbf{y}\mathbf{y}^\top) + \frac{1}{\sigma^2}\mathrm{Tr}(\langle\mathbf{y}\mathbf{f}^\top\rangle_{p(\mathbf{f}|\mathbf{u})}) - \frac{1}{2\sigma^2}\mathrm{Tr}(\langle\mathbf{f}\mathbf{f}^\top\rangle_{p(\mathbf{f}|\mathbf{u})}) \\
&= -\frac{n}{2}\log\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}\mathrm{Tr}(\mathbf{y}\mathbf{y}^\top) + \frac{1}{\sigma^2}\mathrm{Tr}(\mathbf{y}\boldsymbol{\eta}^\top) - \frac{1}{2\sigma^2}\mathrm{Tr}(\langle\mathbf{f}\mathbf{f}^\top\rangle_{p(\mathbf{f}|\mathbf{u})}) \\
&= -\frac{n}{2}\log\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}\mathrm{Tr}(\mathbf{y}\mathbf{y}^\top - 2\mathbf{y}\boldsymbol{\eta}^\top) - \frac{1}{2\sigma^2}\mathrm{Tr}(\mathbb{V}\mathrm{ar}(\mathbf{f})_{p(\mathbf{f}|\mathbf{u})} + \boldsymbol{\eta}\boldsymbol{\eta}^\top),
\end{aligned}$$

where $\boldsymbol{\eta} \coloneqq \langle\mathbf{f}\rangle_{p(\mathbf{f}|\mathbf{u})}$, $\mathbb{V}\mathrm{ar}(\mathbf{f})_{p(\mathbf{f}|\mathbf{u})}$ denotes the covariance matrix of $\mathbf{f}$ under $p(\mathbf{f}|\mathbf{u})$[2], and we have used

$$\mathbb{V}\mathrm{ar}(\mathbf{f})_{p(\mathbf{f}|\mathbf{u})} = \langle\mathbf{f}\mathbf{f}^\top\rangle_{p(\mathbf{f}|\mathbf{u})} - \boldsymbol{\eta}\boldsymbol{\eta}^\top.$$

Now, by definition, we have that $\boldsymbol{\eta}$ is the mean of $\mathbf{f}|\mathbf{u}$ and $\mathbb{V}\mathrm{ar}(\mathbf{f})_{p(\mathbf{f}|\mathbf{u})}$ is the covariance matrix of $\mathbf{f}|\mathbf{u}$. More specifically, (2.10) gives $\boldsymbol{\eta} = \mathbf{K}_{nm}\mathbf{K}_{mm}^{-1}\mathbf{u}$ and $\mathbb{V}\mathrm{ar}(\mathbf{f})_{p(\mathbf{f}|\mathbf{u})} = \tilde{\mathbf{K}} = \mathbf{K}_{nn} - \mathbf{K}_{nm}\mathbf{K}_{mm}^{-1}\mathbf{K}_{mn}$, which finally gives the recognisable expression

$$\begin{aligned}
\langle \log p(\mathbf{y}|\mathbf{f})\rangle_{p(\mathbf{f}|\mathbf{u})} &= -\frac{n}{2}\log\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}\mathrm{Tr}(\mathbf{y}\mathbf{y}^\top - 2\mathbf{y}\boldsymbol{\eta}^\top + \boldsymbol{\eta}\boldsymbol{\eta}^\top) - \frac{1}{2\sigma^2}\mathrm{Tr}(\mathbb{V}\mathrm{ar}(\mathbf{f})_{p(\mathbf{f}|\mathbf{u})}) \\
&= \log(\mathcal{N}(\mathbf{y}|\mathbf{K}_{nm}\mathbf{K}_{mm}^{-1}\mathbf{u}, \sigma^2\mathbf{I}_n)) - \frac{1}{2\sigma^2}\mathrm{Tr}(\mathbf{K}_{nn} - \mathbf{K}_{nm}\mathbf{K}_{mm}^{-1}\mathbf{K}_{mn}).
\end{aligned}$$

[1] More precisely, we use that for a square matrix of random variables $\mathbf{A}_{nn}$ and probability distribution $q(\cdot)$, we have $\langle\mathrm{Tr}(\mathbf{A}_{nn})\rangle_{q(\cdot)} = \langle\sum_{i=1}^{n} a_{ii}\rangle_{q(\cdot)} = \sum_{i=1}^{n}\langle a_{ii}\rangle_{q(\cdot)} = \mathrm{Tr}(\langle\mathbf{A}_{nn}\rangle_{q(\cdot)})$, where a_{ii} is the ith diagonal element of $\mathbf{A}_{nn}$.

[2] That is, $(\mathbb{V}\mathrm{ar}(\mathbf{f})_{p(\mathbf{f}|\mathbf{u})})_{i,j} = \langle(f_i - \eta_i)(f_j - \eta_j)\rangle_{p(\mathbf{f}|\mathbf{u})}$, where $\eta_i = \langle f_i\rangle_{p(\mathbf{f}|\mathbf{u})}, \eta_j = \langle f_j\rangle_{p(\mathbf{f}|\mathbf{u})}$ are the ith and jth elements of $\boldsymbol{\eta}$ respectively.

To get an expression for $\tilde{p}(\mathbf{y}|\mathbf{u})$ we simply exponentiate both sides:

$$\begin{aligned}\tilde{p}(\mathbf{y}|\mathbf{u}) &= \mathcal{N}(\mathbf{y}|\mathbf{K}_{nm}\mathbf{K}_{mm}^{-1}\mathbf{u}, \sigma^2\mathbf{I}_n)\exp(-\frac{1}{2\sigma^2}\mathrm{Tr}(\mathbf{K}_{nn}-\mathbf{K}_{nm}\mathbf{K}_{mm}^{-1}\mathbf{K}_{mn})) \\ &= \prod_{i=1}^{n}\mathcal{N}(y_i|\mathbf{k}_{nm,i}\mathbf{K}_{mm}^{-1}\mathbf{u}, \sigma^2)\left[\exp(-\frac{1}{2\sigma^2}\mathrm{Tr}(\mathbf{K}_{nn}-\mathbf{K}_{nm}\mathbf{K}_{mm}^{-1}\mathbf{K}_{mn}))\right],\end{aligned}$$

where $\mathbf{k}_{nm,i}\mathbf{K}_{mm}^{-1}\mathbf{u}$ is the ith element of $\mathbf{K}_{nm}\mathbf{K}_{mm}^{-1}\mathbf{u}$, and we have used that the marginal normal distributions are uncorrelated and thus independent.

■

Equation 2.26. $\log p(\mathbf{y}) \geq \mathbb{E}_{\mathbf{q(u)}}[\log p(\mathbf{y}|\mathbf{u})] - \mathrm{KL}[q(\mathbf{u})||p(\mathbf{u})]$

Proof. Let the setup be as in section 2.2.3. Noting that $p(\mathbf{y}) = \int p(\mathbf{y}, \mathbf{u})d\mathbf{u}$, we have

$$\begin{aligned}p(\mathbf{y}) &= \int p(\mathbf{y}, \mathbf{u})d\mathbf{u} \\ &= \int p(\mathbf{y}|\mathbf{u})p(\mathbf{u})d\mathbf{u} \\ &= \int q(\mathbf{u})\frac{p(\mathbf{y}|\mathbf{u})p(\mathbf{u})}{q(\mathbf{u})}d\mathbf{u} \\ &\geq \int q(\mathbf{u})\log\frac{p(\mathbf{y}|\mathbf{u})p(\mathbf{u})}{q(\mathbf{u})}d\mathbf{u},\end{aligned}$$

where $q(\cdot)$ is a variational distribution and the last line holds as $\log(x) \leq x \ \forall x \geq 0$. Following on with basic rules of integration gives

$$\begin{aligned}p(\mathbf{y}) &\geq \int q(\mathbf{u})\log\frac{p(\mathbf{y}|\mathbf{u})p(\mathbf{u})}{q(\mathbf{u})}d\mathbf{u} \\ &= \int q(\mathbf{u})\log p(\mathbf{y}|\mathbf{u})d\mathbf{u} - \int q(\mathbf{u})\log\frac{q(\mathbf{u})}{p(\mathbf{u})}d\mathbf{u} \\ &= \mathbb{E}_{\mathbf{q(u)}}[\log p(\mathbf{y}|\mathbf{u})] - \mathrm{KL}[q(\mathbf{u})||p(\mathbf{u})].\end{aligned}$$

■

Appendix B

Additional Plots

B.1 Handwritten Digits Classification

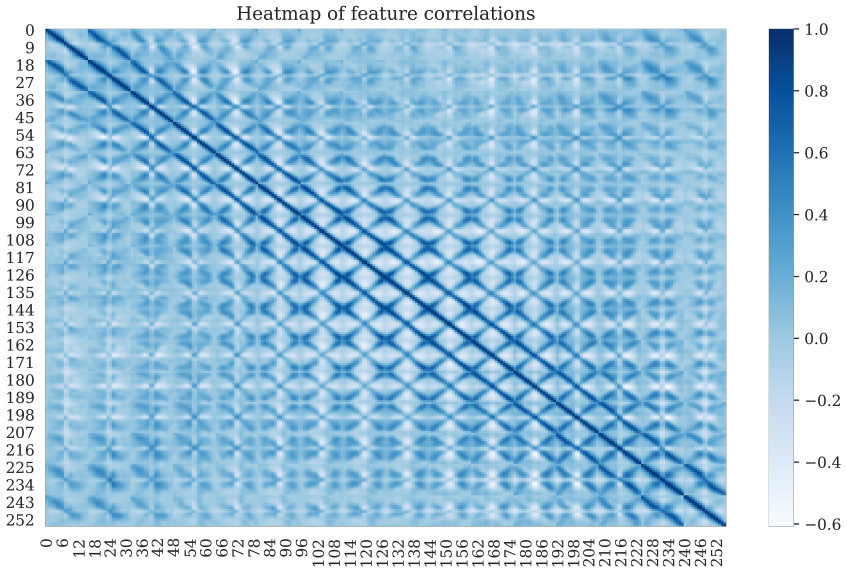


Figure B.1: Heatmap of feature correlations for the re-partitioned USPS dataset. The axes represent the ith features (pixels). The solid blue lines near the diagonal show evidence of collinearity.

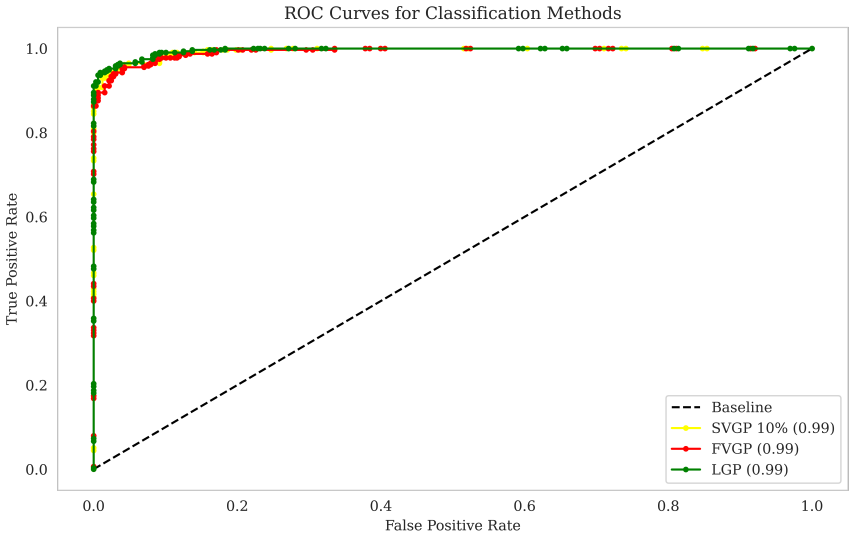


Figure B.2: Plots of the ROC curves obtained for the USPS dataset. The AUC scores for the different methods are specified in brackets. The baseline represents the ROC curve obtained from random prediction.

Chapter 5

Peer-Review Reflection

(See next page)

Here, we will provide feedback on the peer-review process as well as on the feedback received. Firstly, regarding the feedback received:

- Scalars have been represented by lowercase letters (Latin or Greek), vectors by boldfaced numbers or lowercase letters (Latin or Greek), and matrices by boldfaced uppercase letters (Latin or Greek). This notation is used consistently throughout the dissertation and made clear where not already evident from the context or stated in the List of Symbols. Moreover, "x" has been referred to appropriately for when it is a scalar or vector.

- Since the time of the review, there is no longer a mention of sparse matrices, but rather of sparse *methods*. The interest and advantage of sparse methods comes from their reduced computational complexity and scalability. This is explicitly mentioned in chapter 1 as well as in several other parts of the dissertation (e.g. section 2.1.3). The specific impact of sparse methods on computational complexity is mentioned for the sparse approaches discussed (e.g. in section 2.1.3 or 2.2.3).

- Cost of inverting $\mathbf{K}_{nn} + \sigma^2 \mathbf{I}_n$: The standard algorithm for inverting matrices is Gaussian elimination and this has computational complexity $\mathcal{O}(n^3)$ for an n-dimensional matrix. A reference to this is given in the text as an explanation would deviate too much from the main topic. For similar reasons, computational complexities for other approaches have been stated either with or without a reference where a reference was not possible to obtain.

- Multivariate Gaussian distributions have not been introduced because they would deviate too much from the main topic and also because it is expected that the reader will have encountered them in their previous courses. Due to the very theoretical nature of the topic there are also not many visual aids available for explaining the concepts and as a result there are very few visual aids for explaining Gaussian processes in this dissertation.

In terms of the peer-review process, I found it very rewarding. Reviewing other people's work gave me the opportunity to realise aspects of my own dissertation I could improve on, e.g. being more clear with my aims or mathematical notation. Receiving feedback was also greatly helpful for improving my work, such as making my mathematical notation clearer or being more concise about the purpose of sparse approaches.