Pseudo-Marginal Inference for Gaussian Process Classification with Large Datasets

Daniel Williams daniel.williams@bristol.ac.uk
Dom Owens dom.owens@bristol.ac.uk
Jake Spiteri jake.spiteri@bristol.ac.uk

COMPASS CDT, Institute of Statistical Science, University of Bristol

August 18, 2026

Gaussian Process (GP) models are a powerful tool for non-parametric supervised learning tasks, popular in both machine learning and statistics. Following the methods of Filippone and Girolami, 2014, we outline and implement a GP for binary classification with a fully-Bayesian specification, using a pseudo-marginal Metropolis-Hastings algorithm to sample the model hyperparameters from their posteriors. Due to the computational expense of this approach, we extend this model to handle large datasets using the *Informative Vector Machine* of Herbrich, Lawrence, and Seeger, 2003. We detail each step of the modelling process, and compare the performance of the classifier against a Bayesian logistic regression model on the e-mail spam dataset (Dua and Graff, 2017). Our implementation is available as an `R` package at `github.com/dannyjameswilliams/gpc`.

1 Introduction

Consider the probabilistic binary classification problem: each input datum $\boldsymbol{x}$ may belong to one of two classes, $\mathcal{C}_-$ or $\mathcal{C}_+$. Membership is non-deterministic, and we denote class membership for the datum with $y = \pm 1$. After observing pairs $(\boldsymbol{x}_i, y_i), i = 1, \dots, n$, we wish to assign a new observation $\boldsymbol{x}_*$ to a class. We first predict the probability of the data belonging to each class, then assign a class by following some decision rule, chosen to take into account the possible loss incurred by each decision.

Our focus here is on Gaussian processes. These are a flexible non-parametric method, which can closely approximate a rich class of non-linear data generating processes, and the probabilistic nature of these models permits us to quantify how uncertain we are in the predictions we make.

To understand Gaussian process classification (GPC), we should begin by understanding the related problem of Gaussian process regression (GPR), where the aim is to approximate a continuous-response process; a function $\boldsymbol{f}(\cdot)$ defines the mean, and a kernel function $\boldsymbol{k}(\cdot, \cdot)$ defines the covariance. A Gaussian likelihood and a Gaussian prior distribution give rise to a Gaussian posterior, and thus all the necessary computations are analytically tractable. In the classification problem, however, a Gaussian likelihood would be nonsensical, which would mean we lack an analytical form for the posterior distribution. To overcome this, we can approximate the posterior over the latent variables $\boldsymbol{f}$ with a Gaussian distribution (via the Laplace approximation). This will provide us with conjugacy, and allows us to integrate out the latent variables.

In much the same manner that GPR can be cast as a generalisation of linear regression (Williams and Rasmussen, 2006), GPC is a generalisation of linear logistic regression. Data are mapped to class probabilities by applying a sigmoid function to a predicted continuous response; in the general case we allow this response function to be a smooth non-linear function of the data, defined by the chosen hyperparameters.

The standard approach to inferring hyperparameters would be to integrate out the latent variables using some approximation method, then optimise the resulting approximate marginal likelihood; this is called *type II Maximum Likelihood* (ML). Many deterministic approximations for integrating out $\boldsymbol{f}$ exist, including the Laplace Approximation (Section 2.2) used in this report.

It would of course be desirable to extend this to a fully Bayesian approach. We would like to (i) infer all parameters and latent variables from the data, and (ii) make predictions by integrating out latent variables and hyperparameters with respect to their posterior distributions. To do this, we require the joint posterior distribution of the latent variables and the model hyperparameters, which is intractable. Thus we must approximate this quantity using a Markov chain Monte Carlo (MCMC) method, which is itself a non-trivial task given the strong coupling of the latent variables and hyperparameters. To overcome this problem we use a *pseudo-marginal* (PM) approach as proposed in Filippone and Girolami, 2014 to sample the model hyperparameters from their posterior distribution. The PM approach allows us to approximately integrate out the latent variables whilst maintaining the correct invariant distribution, decoupling the samples and thus providing us with better mixing.

Moreover, adoption of GP models in practice has been hindered by the high computational cost of fitting and predicting for large datasets, given these tasks have cost $\mathcal{O}(n^3)$. One method to overcome this was proposed by Herbrich, Lawrence, and Seeger, 2003, which selects a much smaller subset of the training data while minimising the impact on predictive performance. We describe how this can be combined with our model in Section 2.3.

Importantly, we have found no current literature on sparse estimation for fully-Bayesian GPC models, and very little on making these models scalable. Hensman, Matthews, and Ghahramani, 2015 used a variation approach, while Hernández-Lobato and Hernández-Lobato, 2016 uses expectation propagation for sparsity, but is not fully Bayesian.

The rest of the report proceeds as follows:

- In Section 2, the entire modelling process is described, with detailed explanations of how and why quantities are obtained at each step. We present the sparse approximation for use of this model with large datasets, and we prove that the pseudo-marginal approach defines a valid Metropolis-Hastings algorithm. The procedure is presented as a schematic algorithm.
- In Section 3, the computational aspects of the model are discussed; we detail the ways in which runtime can be reduced, and explain the 'tricks' used to provide the method with numerical stability. We also compare the performance of the model to a novel Bayesian logistic regression on the spambase data set.
- We conclude in Section 4, collecting our findings and detailing areas for further work.

2 Model

We specify our model as follows: Let $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ be a set of n-many input vectors each described by d-many covariates, and each associated with a response $y_i \in \{-1, +1\}$, contained in the vector $\boldsymbol{y} = \{y_1, \ldots, y_n\}$. Let $\boldsymbol{f} = \{f_1, \ldots, f_n\}$ be the set of latent variables. Using a generative approach, the classifier assumes the class labels are Bernoulli distributed with success probability determined by a transformation of the class label and latent variable:

$$p(y_i|f_i) = \Phi(y_i f_i). \tag{1}$$

Here, Φ is the cumulative density function of the standard Gaussian distribution. By independence, the likelihood is the product over all observations

$$p(\boldsymbol{y}|\boldsymbol{f}) = \prod_{i=1}^{n} p(y_i|f_i). \tag{2}$$

We model $\boldsymbol{f}$ with a zero mean Gaussian process prior, with covariance matrix $\boldsymbol{K}$:

$$\boldsymbol{f} \sim \mathcal{N}(\boldsymbol{f}|\boldsymbol{0}, \boldsymbol{K}). \tag{3}$$

Each entry $K_{i,j}$ is determined by the evaluation

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j|\boldsymbol{\theta}) = \sigma \exp\left[-\frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top \boldsymbol{A}(\boldsymbol{x}_i - \boldsymbol{x}_j)\right] \tag{4}$$

of our chosen kernel, which is in turn defined by the hyperparameter vector $\boldsymbol{\theta} = (\sigma, \gamma)$. σ controls the variance of the marginal prior, while γ is the length-scale control such that $\boldsymbol{A} = \gamma^{-2}\boldsymbol{I}_d$, giving an isotropic covariance matrix. This specific kernel is known as the Gaussian kernel. We place Gamma priors on the hyperparameters $\boldsymbol{\theta}$ as follows:

$$\sigma \sim \text{Gamma}(\alpha_\sigma, \beta_\sigma), \tag{5}$$
$$\gamma \sim \text{Gamma}(\alpha_\gamma, \beta_\gamma), \tag{6}$$

where $\alpha_\sigma, \beta_\sigma, \alpha_\gamma, \beta_\gamma > 0$.

2.1 Predictive Distribution

With the aim of classifying new data, we want to obtain an approximate predictive distribution for a prediction y_*. Conditioning on a realisation of the hyperparameters $\boldsymbol{\theta}$, in the standard GPC setting we would compute

$$p\left(y_*|\mathbf{y}, \boldsymbol{\theta}\right) = \int p\left(y_*|f_*\right) p\left(f_*|\mathbf{f}, \boldsymbol{\theta}\right) q(\mathbf{f}|\mathbf{y}, \boldsymbol{\theta}) df_* d\mathbf{f}. \tag{7}$$

Here $\boldsymbol{\theta}$ could be a ML estimate that maximizes the approximate likelihood or one sample from the approximate posterior $\hat{p}(\boldsymbol{\theta}|\mathbf{y})$. Note that we suppress notation

for dependence on the data X when writing posterior distributions, and treat this as implicit.

However, we want a fully Bayesian approach, so we integrate out the hyperparameters also, yielding the distribution

$$p\left(y_*|\mathbf{y}\right) = \int p\left(y_*|f_*\right) p\left(f_*|\mathbf{f},\boldsymbol{\theta}\right) p(\mathbf{f},\boldsymbol{\theta}|\mathbf{y}) df_* d\mathbf{f} d\boldsymbol{\theta}. \tag{8}$$

It is not possible to analytically integrate equation (8) with respect to $\mathbf{f}$ or $\boldsymbol{\theta}$, so we draw samples from $p(\mathbf{f},\boldsymbol{\theta}|\mathbf{y})$ with a Monte Carlo method (described in Section 2.4) and use an unbiased estimate for the predictive distribution given by

$$p(y_*|\mathbf{y}) \simeq \frac{1}{N}\sum_{i=1}^{N} \int p\left(y_*|f_*\right) p\left(f_*|\mathbf{f}^{(i)},\boldsymbol{\theta}^{(i)}\right) df_*, \tag{9}$$

where $\mathbf{f}^{(i)},\boldsymbol{\theta}^{(i)}$ denotes the i-th sample from $p(\mathbf{f},\boldsymbol{\theta}|\mathbf{y})$.

For simplicity of notation, let $\boldsymbol{K}$ be the covariance matrix evaluated at $\boldsymbol{\theta}$, $\mathbf{k}_*$ be the vector whose i-th element is $k\left(\mathbf{x}_i,\mathbf{x}_*|\boldsymbol{\theta}\right)$, and $k_{**} = k\left(\mathbf{x}_*,\mathbf{x}_*|\boldsymbol{\theta}\right)$ be the kernel evaluated on the new set of observations $\boldsymbol{x}_*$.

Given the properties of multivariate Gaussian variables, and as per the specification of the GP model, f_* is distributed as $\mathcal{N}\left(f_*|\mu_*,\beta_*^2\right)$ with $\mu_* = \mathbf{k}_*^{\mathrm{T}}\boldsymbol{K}^{-1}\mathbf{f}$ and $\beta_*^2 = k_{**} - \mathbf{k}_*^{\mathrm{T}}\boldsymbol{K}^{-1}\mathbf{k}_*$.

Approximating $p(\mathbf{f}|\mathbf{y},\boldsymbol{\theta})$ with the Laplace approximation (2.2) makes it possible to perform analytic integration with respect to $\boldsymbol{f}$ in equation (8). In particular, the integration with respect to $\boldsymbol{f}$ yields $\mathcal{N}\left(f_*|m_*,s_*^2\right)$ with the predictive mean

$$m_* = \mathbf{k}_*^{\mathrm{T}}\boldsymbol{K}^{-1}\boldsymbol{\mu}_q, \tag{10}$$

and predictive variance

$$s_*^2 = k_{**} - \mathbf{k}_*^{\mathrm{T}}\boldsymbol{K}^{-1}\mathbf{k}_* + \mathbf{k}_*^{\mathrm{T}}\boldsymbol{K}^{-1}\boldsymbol{\Sigma}_q\boldsymbol{K}^{-1}\mathbf{k}_*, \tag{11}$$

where $\boldsymbol{\mu}_q$ and $\boldsymbol{\Sigma}_q$ parameterise the mean and covariance respectively of the Laplace approximation.

The integration with respect to f_* has a closed form, as it is a convolution of a Gaussian and a cumulative Gaussian:

$$t_* = \int p\left(y_*|f_*\right)\mathcal{N}\left(f_*|m_*,s_*^2\right) df_* = \Phi\left(\frac{m_*}{\sqrt{1+s_*^2}}\right). \tag{12}$$

We make a prediction y_* for $\boldsymbol{x}_*$ from the predictive distribution according to the a_-–a_+ decision rule $\delta : [0,1] \to \{-1,+1\}$ such that

$$y_* = \delta(t_*) = \begin{cases} +1, & t_* > a_+/(a_- + a_+), \\ -1, & \text{otherwise.} \end{cases} \tag{13}$$

Here, $a_-, a_+ \geq 0$ are chosen to quantify the loss associated with wrongly predicting y_- and y_+ respectively.

Note the similarities of this classification rule to the hard-margin support vector machine (SVM) (see, for instance, Williams and Rasmussen, 2006, Section 6).

2.2 Laplace Approximation

By approximating the posterior of the latent variables given the data and hyperparameters $p(\boldsymbol{f}|\boldsymbol{y},\boldsymbol{\theta})$ (the target density) with a Gaussian $\mathcal{N}(\boldsymbol{f}|\boldsymbol{\mu}_q,\boldsymbol{\Sigma}_q)$, we obtain the **Laplace approximation**:

$$p(\boldsymbol{f}|\boldsymbol{y},\boldsymbol{\theta}) \approx q(\boldsymbol{f}|\boldsymbol{y},\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{f}|\boldsymbol{\mu}_q,\boldsymbol{\Sigma}_q). \tag{14}$$

Substituting this into the predictive distribution (8) gives conjugacy, permitting analytical integration. For the approximation to be sufficiently close in location and curvature, we set $\boldsymbol{\Sigma}_q^{-1}$ equal to the negative Hessian of the target density.

Given a realisation of the hyperparameters $\boldsymbol{\theta}$, define

$$\Psi(\boldsymbol{f}) = \log[p(\boldsymbol{y}|\boldsymbol{f})] + \log[p(\boldsymbol{f}|\boldsymbol{\theta})] + C \tag{15}$$

as the logarithm of the target density, constant with respect to terms independent of $\boldsymbol{f}$ (represented by C).

We select a Gaussian $q(\boldsymbol{f}|\boldsymbol{y},\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{f}|\hat{\boldsymbol{f}},\hat{\boldsymbol{\Sigma}})$ where

$$\hat{\boldsymbol{f}} = \arg\max_{\boldsymbol{f}} \Psi(\boldsymbol{f}), \quad \hat{\boldsymbol{\Sigma}}^{-1} = -\nabla_{\boldsymbol{f}}\nabla_{\boldsymbol{f}}\Psi(\hat{\boldsymbol{f}}), \tag{16}$$

where the solution $\hat{\boldsymbol{f}}$ can be solved by iterating the Newton-Raphson formula

$$\mathbf{f}_{\text{new}} = \mathbf{f} - \left(\nabla_{\mathbf{f}}\nabla_{\mathbf{f}}\Psi(\mathbf{f})\right)^{-1}\nabla_{\mathbf{f}}\Psi(\mathbf{f}), \tag{17}$$

starting from $\boldsymbol{f} = \mathbf{0}$ until convergence. The gradient and the Hessian of the log-target density are:

$$\nabla_{\mathbf{f}}\Psi(\mathbf{f}) = \nabla_{\mathbf{f}}\log[p(\mathbf{y}|\mathbf{f})] - \boldsymbol{K}^{-1}\mathbf{f}, \tag{18}$$

$$\nabla_{\mathbf{f}}\nabla_{\mathbf{f}}\Psi(\mathbf{f}) = \nabla_{\mathbf{f}}\nabla_{\mathbf{f}}\log[p(\mathbf{y}|\mathbf{f})] - \boldsymbol{K}^{-1}. \tag{19}$$

The log-target density has a unique maximum by concavity.

2.3 Sparse Approximation

For large data sets, it is necessary to use some approximation to reduce the $\mathcal{O}(n^3)$ complexity of fitting the full model. For a discussion of possible methods, see Williams and Rasmussen, 2006, Section 8.5. We propose the use of a *subset of data* (SD) method as described in Herbrich, Lawrence, and Seeger, 2003, referred to as the *Informative Vector Machine*. This defines a sparse GPC model by greedily adding data entries to the active subset $I \subset \{1,\ldots,n\}$ until $|I| = d$, where $d << n$ is chosen as the reduced subset size. Points are added according to an entropy heuristic; at each iteration, a score function is calculated for each point in the inactive set $J = \{1,\ldots,n\}\backslash I$ and we select the maximiser as candidate.

Given a Laplace approximation $q(\boldsymbol{f}|\boldsymbol{y},\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{f}|\boldsymbol{\mu},\boldsymbol{\Sigma})$ from (14), the chosen heuristic, *differential entropy score* (26), favours the point which most decreases the posterior variance when included in the active set.

We adapt the following from Herbrich, Lawrence, and Seeger, 2003, Section 3 into our notation. We

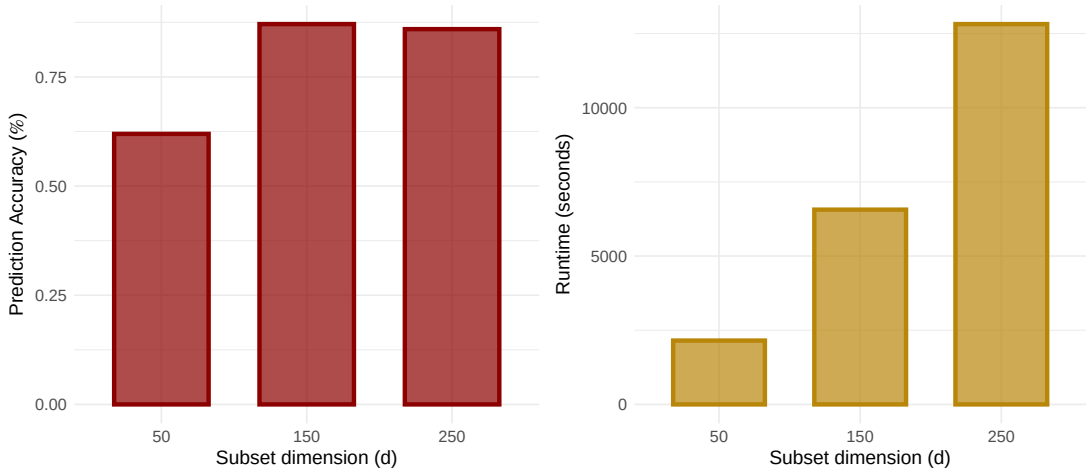

Figure 1: *Predictive accuracy and runtime, by IVM subset size.*

Algorithm 1: Informative vector machine algorithm (IVM)

input : Target sparsity $d << n$
initialise : $I = \emptyset, \boldsymbol{m} = \mathbf{0}$,
$\boldsymbol{\Pi} = \mathrm{diag}(\mathbf{0})$,
$\mathrm{diag}(\boldsymbol{\Sigma}) = \mathrm{diag}(\boldsymbol{K})$,
$\boldsymbol{\mu} = \mathbf{0}, J = \{1, \ldots, n\}$
output : $I \subset \{1, \ldots, n\}$ with $|I| = d$

repeat
  for $j \in J$ **do**
    Compute Δ_j from (26)
  end
  $i = \arg\max_{j \in J} \Delta_j$
  Do updates for p_i and m_i according to (22)
  Update matrices $\boldsymbol{L}, \boldsymbol{M}, \mathrm{diag}(\boldsymbol{\Sigma})$ and $\boldsymbol{\mu}$ according to (23)
  $I \leftarrow I \cup \{i\}, J \leftarrow J \backslash \{i\}$.
until $|I| = d$

have $\boldsymbol{\Sigma} := \left(\boldsymbol{K}^{-1} + \boldsymbol{\Pi}\right)^{-1}, \boldsymbol{\mu} := \boldsymbol{\Sigma}\Pi\boldsymbol{m}$ and $\boldsymbol{\Pi} := \nabla_{\mathbf{f}}\nabla_{\mathbf{f}} \log[p(\mathbf{y}|\mathbf{f})] = \mathrm{diag}(\boldsymbol{p})$ for vectors $\boldsymbol{m}, \boldsymbol{p}$. If I is the current active set, then all of components of the $\boldsymbol{p}$ and $\boldsymbol{m}$ which are not in I are zero, and by the Woodbury identity we have

$$\boldsymbol{\Sigma} = \boldsymbol{K} - \boldsymbol{M}^{\mathrm{T}}\boldsymbol{M}, \quad \boldsymbol{M} = \boldsymbol{L}^{-1}\boldsymbol{\Pi}_I^{1/2}\boldsymbol{K}_{I,\cdot} \in \mathbb{R}^{d,n}, \tag{20}$$

where $\boldsymbol{L}$ is the lower-triangular Cholesky factor of

$$\boldsymbol{B} = \boldsymbol{I} + \boldsymbol{\Pi}_I^{1/2}\boldsymbol{K}_I\boldsymbol{\Pi}_I^{1/2} \in \mathbb{R}^{d,d}. \tag{21}$$

In order to compute the differential entropy score for a point $j \notin I$, we have to know $\Sigma_{j,j}$ and μ_j. Thus, when including i into the active set I, we need to update $\mathrm{diag}(\boldsymbol{\Sigma})$ and $\boldsymbol{\mu}$ accordingly, which in turn requires the matrices $\boldsymbol{L}$ and $\boldsymbol{M}$ to be kept up-to-date. The update equations for p_i, m_i are

$$\begin{aligned} p_i &= \tfrac{\nu_i}{1-\Sigma_{i,i}\nu_i}, \quad m_i = \mu_i + \tfrac{\alpha_i}{\nu_i}, \quad \text{where} \\ z_i &= \tfrac{y_i \cdot (\mu_i)}{\sqrt{1+\Sigma_{i,i}}}, \quad \alpha_i = \tfrac{y_i \cdot N(z_i|0,1)}{\Phi(z_i)\sqrt{1+\Sigma_{i,i}}}, \\ \nu_i &= \alpha_i\left(\alpha_i + \tfrac{\mu_i}{1+\Sigma_{i,i}}\right). \end{aligned} \tag{22}$$

We then update $\boldsymbol{L} \rightarrow \boldsymbol{L}^{\text{new}}$ by appending the row $(\boldsymbol{l}^{\mathrm{T}}, l)$, and $\boldsymbol{M} \rightarrow \boldsymbol{M}^{\text{new}}$ by appending the row $\boldsymbol{h}^{\mathrm{T}}$, where

$$\begin{aligned} l = \sqrt{p_i}M_{\cdot,i}, \quad l &= \sqrt{1 + p_i K_{i,i} - l^{\mathrm{T}}l}, \\ \boldsymbol{h} &= l^{-1}\left(\sqrt{p_i}K_{\cdot,i} - M^{\mathrm{T}}l\right). \end{aligned} \tag{23}$$

The final updates are

$$\mathrm{diag}\left(\boldsymbol{\Sigma}^{\text{new}}\right) \leftarrow \mathrm{diag}(\boldsymbol{\Sigma}) - \left(h_j^2\right)_j, \tag{24}$$

$$\boldsymbol{\mu}^{\text{new}} \leftarrow \boldsymbol{\mu} + \alpha_i l p_i^{-1/2}\boldsymbol{h}. \tag{25}$$

The differential entropy score for $j \notin I$ can be computed based on the variables in (22) (with $i \rightarrow j$) as

$$\Delta_j = \frac{1}{2}\log\left(1 - \Sigma_{j,j}\nu_j\right) \tag{26}$$

This has total complexity $\mathcal{O}(nd^2)$. Fitting the subsequent GPC model will have complexity $\mathcal{O}(d^3)$, which leads to significant runtime reductions in practice, and does so while minimising the trade-off in predictive performance, given we have selected an information-optimal subset of data.

For the rest of the paper, we continue to refer to the full dataset without loss of generality. Any calculations can be assumed to have been performed on a sparse subset instead.

2.4 MCMC Sampling from $p(\mathrm{f}, \boldsymbol{\theta}|\mathrm{y})$

As we have pointed out, sampling from the posterior $p(\boldsymbol{f}, \boldsymbol{\theta}|\boldsymbol{y})$ with joint proposals is not feasible, given the strong correlation between the two sampled quantities. Instead, we update $\boldsymbol{f}$ and $\boldsymbol{\theta}$ in turn with Gibbs samples.

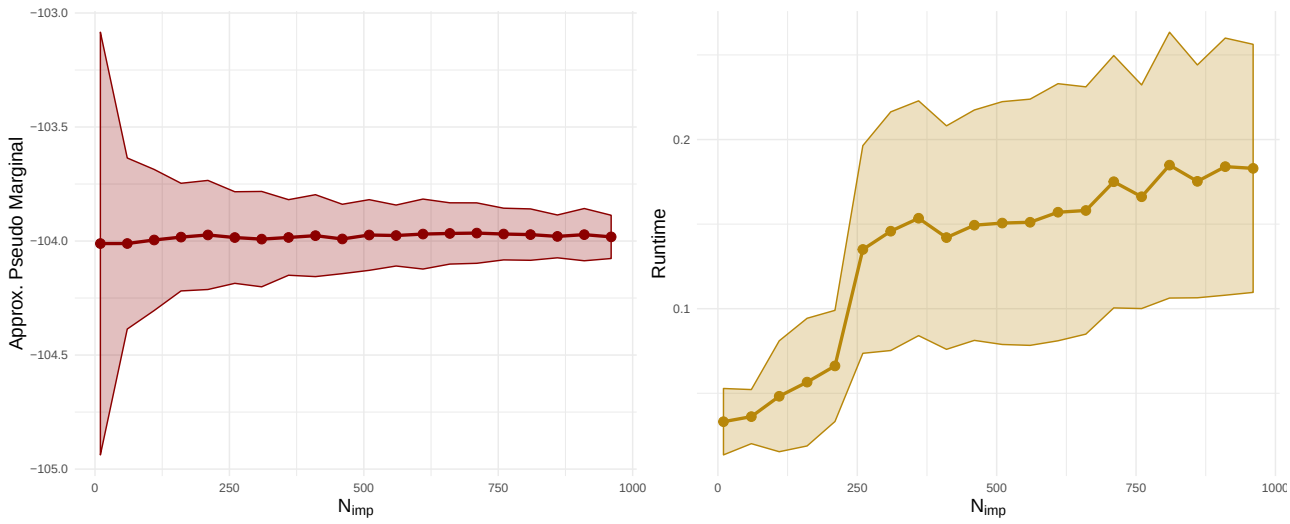

Figure 2: *Number of importance samples N_{imp} against the mean and 95% interval of the approximate pseudo-marginal evaluation calculated from (30) (left), and against the mean and 95% interval of the computation time for each step (right). At each value of N_{imp}, the approximate marginal value was calculated 50 times.*

2.4.1 Sampling from $p(\boldsymbol{f}|\boldsymbol{y},\boldsymbol{\theta})$

We use the *elliptical slice sampling* (ELL-SS) (Murray, Prescott Adams, and MacKay, 2010) method to sample our latent variables $\boldsymbol{f}$. We express the posterior for $\boldsymbol{f}$ as

$$p(\boldsymbol{f}|\boldsymbol{y},\boldsymbol{\theta}) \propto \mathcal{N}(\boldsymbol{f}|\hat{\boldsymbol{f}},\hat{\boldsymbol{\Sigma}})p(\boldsymbol{y}|\boldsymbol{f}) \tag{27}$$

with the likelihood function $p(\boldsymbol{y}|\boldsymbol{f}) = L(\boldsymbol{f})$. This method samples according to the ELL-SS algorithm below, constrained to an ellipse centred at the current mean $\boldsymbol{f}$, with eccentricity determined by the covariance $\hat{\boldsymbol{\Sigma}}$. This encourages steps of greater length in directions with greater covariance, and obviates the need for sample rejection.

The benefits of this are that no tuning is required, and each iteration has complexity $\mathcal{O}(n^2)$ (or $\mathcal{O}(d^2)$ under the reduction in Section 2.3), which is dominated by the complexity of fitting the entire model.

2.4.2 Sampling from $p(\boldsymbol{\theta}|\boldsymbol{y})$ with the Pseudo-Marginal approach

It remains to sample from $p(\boldsymbol{\theta}|\boldsymbol{y})$. While we can't do this by integrating out the hyperparameters $\boldsymbol{f}$, we can draw MCMC samples using an unbiased estimate of $p(\boldsymbol{y}|\boldsymbol{\theta})$.

Consider the Metropolis-Hastings algorithm with proposal $\pi(\boldsymbol{\theta}'|\boldsymbol{\theta})$. We aim to sample from

$$p(\boldsymbol{\theta}|\boldsymbol{y}) \propto p(\boldsymbol{y}|\boldsymbol{\theta})p(\boldsymbol{\theta}). \tag{28}$$

The acceptance ratio would be

$$z = \frac{p(\boldsymbol{y}|\boldsymbol{\theta}')p(\boldsymbol{\theta}')}{p(\boldsymbol{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})}\frac{\pi(\boldsymbol{\theta}|\boldsymbol{\theta}')}{\pi(\boldsymbol{\theta}'|\boldsymbol{\theta})}. \tag{29}$$

We are unable to do this due to the intractability of integrating out the latent variables for $p(\boldsymbol{y}|\boldsymbol{\theta})$. As the following theorem demonstrates, however, we can sample from $p(\boldsymbol{\theta}|\boldsymbol{y})$ using only an unbiased estimator for $p(\boldsymbol{y}|\boldsymbol{\theta})$.

One possible unbiased estimator $\tilde{p}(\boldsymbol{y}|\boldsymbol{\theta})$ can be found through importance sampling. Drawing N_{imp}-many samples $\boldsymbol{f}_i$ from $q(\boldsymbol{f}|\boldsymbol{y},\boldsymbol{\theta})$ allows the approximation

$$\tilde{p}(\boldsymbol{y}|\boldsymbol{\theta}) \simeq \frac{1}{N_{\text{imp}}}\sum_{i=1}^{N_{imp}}\frac{p(\boldsymbol{y}|\boldsymbol{f}_i)p(\boldsymbol{f}_i|\boldsymbol{\theta})}{q(\boldsymbol{f}_i|\boldsymbol{y},\boldsymbol{\theta})}. \tag{30}$$

Lemma 1. *$\tilde{p}(\boldsymbol{y}|\boldsymbol{\theta})$ is an unbiased estimator for $p(\boldsymbol{y}|\boldsymbol{\theta})$.*

Proof. Each importance sample $\boldsymbol{f}_i$ is independent, so we only need to show $(p(\boldsymbol{y}|\boldsymbol{f}_1)p(\boldsymbol{f}_1|\boldsymbol{\theta}))q(\boldsymbol{f}_1|\boldsymbol{y},\boldsymbol{\theta})^{-1}$ has expectation equal to $E[p(\boldsymbol{y}|\boldsymbol{\theta})]$.

The Laplace approximation $q(\boldsymbol{f}_1|\boldsymbol{y},\boldsymbol{\theta})$ has expectation $p(\boldsymbol{f}_1|\boldsymbol{y},\boldsymbol{\theta})$, so by Bayes' theorem $(p(\boldsymbol{y}|\boldsymbol{f}_1)p(\boldsymbol{f}_1|\boldsymbol{\theta}))q(\boldsymbol{f}_1|\boldsymbol{y},\boldsymbol{\theta})^{-1}$ is unbiased, and hence $\tilde{p}(\boldsymbol{y}|\boldsymbol{\theta})$ is unbiased. □

From the results of Beaumont, 2003, we can plug in an unbiased estimate $\tilde{p}(\boldsymbol{y}|\boldsymbol{\theta})$ and draw samples from the correct posterior $p(\boldsymbol{\theta}|\boldsymbol{y})$.

Theorem 1. *Using the unbiased estimator $\tilde{p}(\boldsymbol{y}|\boldsymbol{\theta})$ in place of $p(\boldsymbol{y}|\boldsymbol{\theta})$, we have a valid Metropolis-Hastings algorithm with $p(\boldsymbol{\theta}|\boldsymbol{y})$ as the invariant distribution.*

Proof. Here, we follow the argument from Beaumont, 2003, Appendix. With importance samples

$$\boldsymbol{f}'_{1:N_{imp}} = \{\boldsymbol{f}'_i, i = 1,\ldots,N_{imp}\} \tag{31}$$

and acceptance ratio

$$\tilde{z} = \frac{\tilde{p}(\boldsymbol{y}|\boldsymbol{\theta}')p(\boldsymbol{\theta}')}{\tilde{p}(\boldsymbol{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})}\frac{\pi(\boldsymbol{\theta}|\boldsymbol{\theta}')}{\pi(\boldsymbol{\theta}'|\boldsymbol{\theta})}, \tag{32}$$

Algorithm 2: Elliptical Slice Sampling (ELL-SS)

input : current state $\boldsymbol{f}$, a routine that samples from $\mathcal{N}(\hat{\boldsymbol{f}}, \hat{\boldsymbol{\Sigma}})$, log-likelihood function $\log L$

output : a new state $\boldsymbol{f}'$. When $\boldsymbol{f}$ is drawn from $p^*(\boldsymbol{f}) \propto \mathcal{N}(\boldsymbol{f}; \hat{\boldsymbol{f}}, \hat{\boldsymbol{\Sigma}}) L(\boldsymbol{f})$, the marginal distribution of $\boldsymbol{f}'$ is also p^*

1 Choose ellipse: $\nu \sim \mathcal{N}(\hat{\boldsymbol{f}}, \hat{\boldsymbol{\Sigma}})$

2 Log-likelihood threshold:

$$u \sim \text{ Uniform } [0,1]$$
$$\log y \leftarrow \log L(\boldsymbol{f}) + \log u$$

3 Draw an initial proposal, also defining a bracket:

$$\theta \sim \text{ Uniform } [0, 2\pi]$$
$$[\theta_{\min}, \theta_{\max}] \leftarrow [\theta - 2\pi, \theta]$$

4 $\boldsymbol{f}' \leftarrow \boldsymbol{f} \cos\theta + \nu \sin\theta$

5 **if** $\log L\left(\boldsymbol{f}'\right) > \log y$ **then**

6 Accept: return $\boldsymbol{f}'$

7 **else**

8 Shrink the bracket and try a new point:

9 **if** $\theta < 0$ **then**

10 $\theta_{\min} \leftarrow \theta$

11 **else**

12 $\theta_{\max} \leftarrow \theta$

13 **end**

14 $\theta \sim$ Uniform $[\theta_{\min}, \theta_{\max}]$

15 GoTo 4

16 **end**

we substitute our unbiased estimator $\tilde{p}(\boldsymbol{y}|\boldsymbol{\theta})$ for $p(\boldsymbol{y}|\boldsymbol{\theta})$ (as per Lemma 1) into (32). Expanding the summation, we may express this as follows:

$$\tilde{z} = \frac{\sum_{j=1}^{N_{imp}} \left[p(\boldsymbol{y}|\boldsymbol{f}'_j) p(\boldsymbol{f}'_j|\boldsymbol{\theta}') p(\boldsymbol{\theta}') \prod_{i \neq j} q(\boldsymbol{f}'_i|\boldsymbol{y}, \boldsymbol{\theta}') \right]}{\sum_{j=1}^{N_{imp}} \left[p(\boldsymbol{y}|\boldsymbol{f}_j) p(\boldsymbol{f}_j|\boldsymbol{\theta}) p(\boldsymbol{\theta}) \prod_{i \neq j} q(\boldsymbol{f}_i|\boldsymbol{y}, \boldsymbol{\theta}) \right]} \times \frac{\prod_{i=1}^{N_{imp}} q(\boldsymbol{f}_i|\boldsymbol{y}, \boldsymbol{\theta})}{\prod_{i=1}^{N_{imp}} q(\boldsymbol{f}'_i|\boldsymbol{y}, \boldsymbol{\theta}')} \frac{\pi(\boldsymbol{\theta}|\boldsymbol{\theta}')}{\pi(\boldsymbol{\theta}'|\boldsymbol{\theta})}. \tag{33}$$

This is a Hastings ratio with the importance sample

$$\pi\left(\boldsymbol{f}'_{1:N_{imp}}, \boldsymbol{\theta}' | \boldsymbol{f}, \boldsymbol{\theta}\right) = \prod_{i=1}^{N_{imp}} q(\boldsymbol{f}'_i|\boldsymbol{y}, \boldsymbol{\theta}') \pi(\boldsymbol{\theta}'|\boldsymbol{\theta}) \tag{34}$$

and the the target joint distribution

$$p(\boldsymbol{y}|\boldsymbol{f}) p(\boldsymbol{f}|\boldsymbol{\theta}) p(\boldsymbol{\theta}) = p(\boldsymbol{y}, \boldsymbol{f}, \boldsymbol{\theta}), \tag{35}$$

which is proportional to $p(\boldsymbol{\theta}|\boldsymbol{y})$, and we hence have a valid MH algorithm.

□

Selecting the number N_{imp} of importance samples to draw is non-trivial. In Figure 2, we demonstrate how the pseudo-marginal likelihood changes with respect to N_{imp} for the example spam dataset used in Section 3.2, with the computation time at each value of N_{imp}.

It is clear that all values of N_{imp} give a mean likelihood at approximately the same value. The lower values of N_{imp}, within the range of 1 to 100, have large but quickly decreasing variance, while the variance decreases at a far slower rate for values greater than 100. The computation time increases at a constant rate with respect to N_{imp}, other than a jump at $N_{imp} = 200$.

The calculation of the pseudo-marginal approximation is the dominant cost in every step of the Metropolis-Hastings algorithm, so the choice of N_{imp} is critical when attempting to keep computation time low. For this reason, we opt to use $N_{imp} = 100$ in further analysis to balance computation time and accuracy.

2.5 Algorithm

Algorithm 3 demonstrates the entire modelling procedure, to aid the reader in both following and implementing our work.

3 Implementation

Having outlined and justified our chosen GPC model mathematically, we now turn to the implementation of our model as software. Given the computationally-intensive nature of GP models, we should give serious consideration to the choices we make in the structure of the code. To evaluate the effectiveness of the model, we compare predictive and computational performance with the simpler logistic regression model on the spambase dataset.

3.1 Computational Considerations

Examining Algorithm 3, we can see a loop for sampling from $q(\mathbf{f}|\mathbf{y}, \boldsymbol{\theta}')$ embedded in a larger loop to sample $\boldsymbol{\theta}$. We should hence choose a compiled language over an interpreted language (in our case, `Rcpp` over `R`) when implementing to see serious performance gains. Indeed, the great number of linear algebraic operations means that using a package designed to access linear algebra suborutines would be preferable. We have opted to use `RcppArmadillo` (Eddelbuettel and Sanderson, 2014).

The algorithm can be run over multiple chains, each initiated with different initial samples but the same priors, and analysis of different chains allows us to check for good mixing and metastability.

Within each chain we parallelise the construction of the covariance matrix $\boldsymbol{K}$, sending each kernel evaluation (4) to a different worker, as well as the computation of the pseudo-marginal approximation, since neither of these operations are dependent on previous

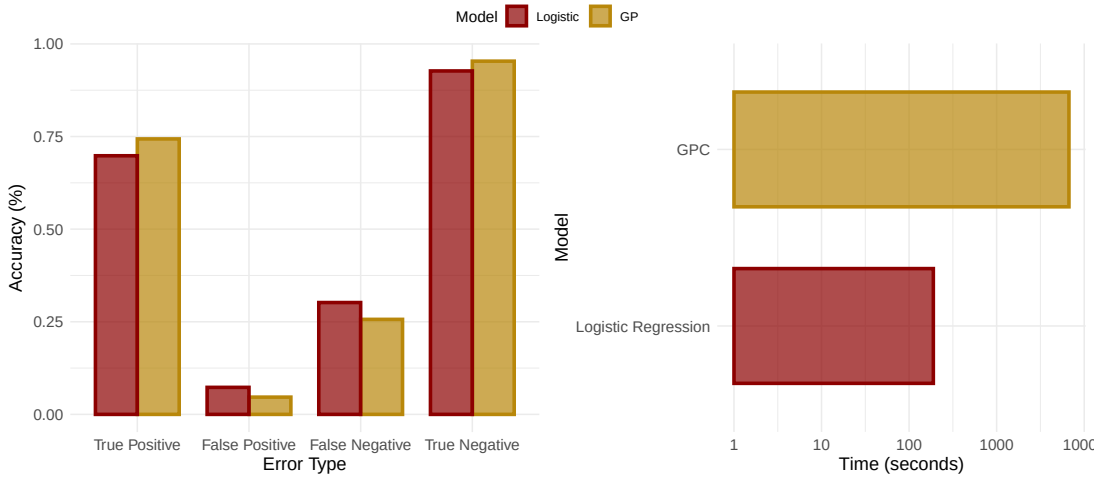

Figure 3: *Comparative confusion plots of predictive accuracy for GPC and logistic regression (left) and time plots in seconds for GPC and logistic regression, on log scale (right)*

Algorithm 3: Gaussian Process Classification via Pseudo-Marginal Approach

input : A prior for $\boldsymbol{\theta}$ and $\mathbf{f}$, number of iterations N, data $(X, \boldsymbol{y})$, subset size $d < n$

Sample $\boldsymbol{\theta}^{(1)}$ from prior
Select a subset of size d from the data $(X, \boldsymbol{y})$ using IVM (Algorithm 1)
for $i = 2 : N$ **do**
  case *Sampling* $\boldsymbol{\theta}$ **do**
    Draw $\boldsymbol{\theta}'$ from proposal $\pi(\boldsymbol{\theta}'|\boldsymbol{\theta}^{(i-1)})$
    Use Laplace approximation to approximate $p(\mathbf{f}|\mathbf{y}, \boldsymbol{\theta}')$ with $q(\mathbf{f}|\mathbf{y}, \boldsymbol{\theta}')$ from (14)
    Obtain N_{imp} samples from $q(\mathbf{f}|\mathbf{y}, \boldsymbol{\theta}')$ to get $\tilde{p}(\mathbf{y}|\boldsymbol{\theta}')$ from (30)
    Compute acceptance probability $A = \min\{1, \tilde{z}\}$ with $\tilde{z}$ from (32)
    Draw $u \sim \text{Unif}(0, 1)$
    if $A > u$ **then**
      Next pair is $(\boldsymbol{\theta}', \tilde{p}(\mathbf{y}|\boldsymbol{\theta}'))$
      $\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}'$
    else
      Next pair is $(\boldsymbol{\theta}^{(i-1)}, \tilde{p}(\mathbf{y}|\boldsymbol{\theta}^{(i-1)}))$
      $\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{(i-1)}$
    end
  end
  case *Sampling* $\mathbf{f}|\boldsymbol{\theta}$ **do**
    Plug $\boldsymbol{\theta}^{(i)}$ into the prior for $\mathbf{f}$
    Sample $\mathbf{f}$ from this prior using ELL-SS (Algorithm 2)
  end
end
Use posteriors of $\boldsymbol{\theta}$ and $\mathbf{f}|\boldsymbol{\theta}$ to give predictions with (12)

iterations, and parallelisation is straight forward. Implementing this parallel approach sped up the model run time, making it approximately twice as fast.

In evaluating the Laplace approximation (Section 2.2), when estimating $\hat{\boldsymbol{\Sigma}}^{-1}$ we are in effect adding low-rank updates to another inverted matrix $\boldsymbol{K}^{-1}$ (and similarly in evaluating $\hat{\boldsymbol{f}}$), meaning we can find computational savings with the Woodbury identity. We can also use this identity to avoid directly inverting the Gram matrix $\boldsymbol{K}$, an operation which is highly susceptible to numerical error in practice. Details on this are available in Williams and Rasmussen, 2006, Algorithm 3.1.

Then, in evaluating the approximate marginal distribution $\tilde{p}(\boldsymbol{y}|\boldsymbol{\theta})$ for a sample, we need to multiply many small probabilities. Given that these are so small, finding the product is likely to cause numerical underflow on a machine. We hence use the *LogSumExp* trick (*LogSumExp Trick*), instead summing the logarithms of the probabilities then taking the exponent. We also work with log probabilities in general to avoid numerical underflow and overflow. This includes calculating the log acceptance probability in the Metropolis-Hastings algorithm.

3.2 Model Comparison

To understand how well the GPC model performs relative to the simpler Bayesian logistic regression model, we compare the models by the loss that they induce (we use a a_-–a_+ loss with $a_- = 1, a_+ = 1$), and the time taken to produce a fit.

For a detailed explanation of the links between the two models, confer Williams and Rasmussen, 2006, Section 6.5.

We use the e-mail spam dataset provided by Dua and Graff, 2017 for comparison between the logistic regression and our GPC approach, and to test performance of our model. Briefly, this dataset contains $n = 4601$

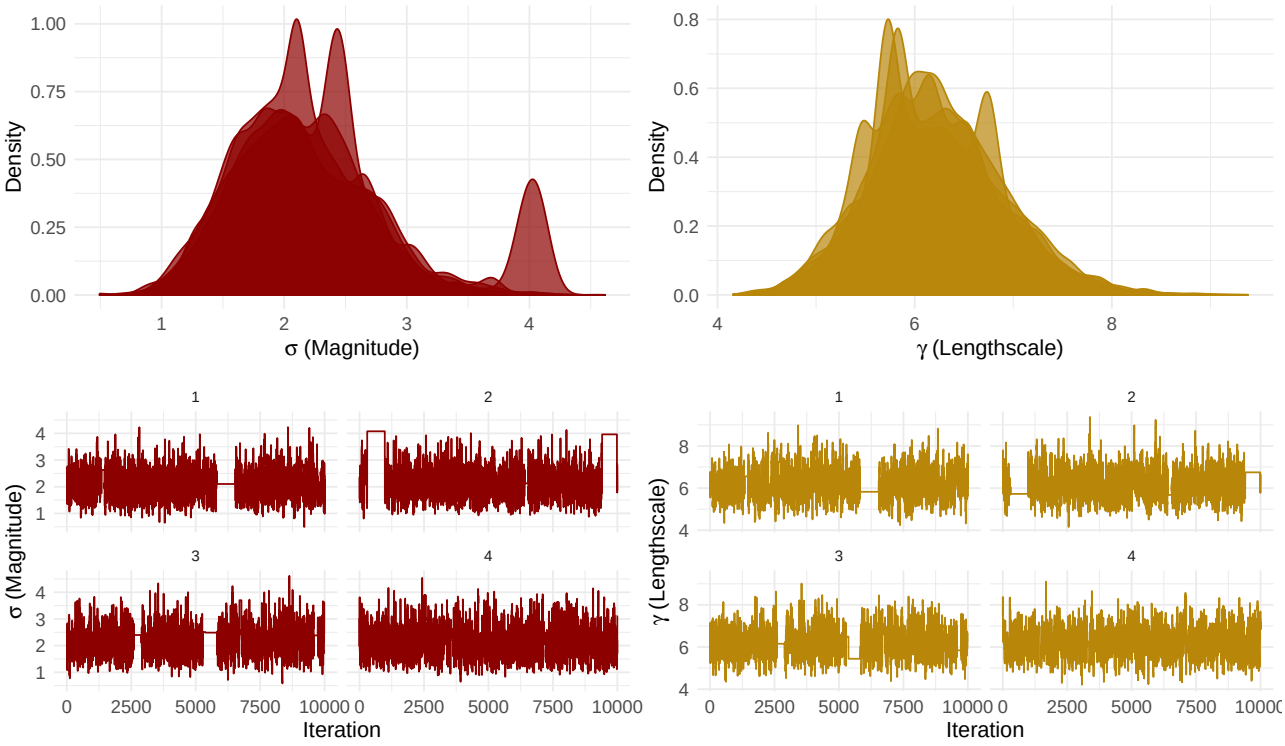

Figure 4: *Posterior distributions and trace plots across four chains for the two kernel hyperparameters. Model run with 20,000 iterations, of which the first 10,000 iterations were discarded as 'burn in'.*

observations (emails) on $p = 57$ predictors (attributes of each email), each classified as spam ($+$) or not spam ($-$).

Figure 2 suggests that we use a value of 100 for N_{imp}, as we obtain reliable estimates of the marginal likelihood, for a manageable run time. We must also decide the value d corresponding to the size of the subset selected by IVM. In doing this, we are balancing predictive accuracy and run time.

In Figure 1 we can see that runtime increases significantly with d, but predictive accuracy seems to level out beyond $d = 150$, so we select this value. We fit a model with 10000 steps, and discard the first 5000 of these as burn-in. Hyperparameters are initialised with $\boldsymbol{\theta} = (1, 1)$, and the variance of our random-walk proposal mechanism for the kernel parameters was tuned such that the algorithm's acceptance ratio was consistently between 0.2 and 0.3; doing this ensures good mixing.

The fully-Bayesian logistic regression model was implemented in `rjags` (Plummer, 2019), using a subset of 13 predictors. These predictors were hand selected according to how informative we perceived they would be, and the subset was used as it would be computationally expensive to optimise the model over a 57-dimensional space, and it may not be possible given a finite set of data. The `rjags` model was used to produce 20,000 parameter samples, of which the initial 10,000 samples were discarded as burn-in. To compare the two models, a training set containing 70% of all observations was used to train the models, and their performance was evaluated using a test set containing the remaining 30% of the observations.

In Figure 3, we can see that the GPC model consistently outperforms the logistic regression model on this data set, giving marginal but significant improvements for each error type.

Indeed, for datasets with a greater number of predictors p, our method should outperform logistic regression on both terms, since ours does not scale in complexity with p and has a more explicit control of regularity in the kernel space.

Finally, we can also see in Figure 3 that the fitting the GP model with $d = 150$ takes approximately 100 times as long as fitting logistic regression. This should be taken into account when deciding which model should be used in practice - if time is an important factor, a smaller value of d should be considered.

3.3 Fitting for a Large Number of Iterations

We now present the Gaussian process classification model, fit with a large number of iterations (20,000) and multiple chains (4) in the Metropolis-Hasting algorithm. This result is presented so that the reader may gain confidence in the convergence of the model.

Using the values of $N_{imp} = 100$ and $d = 150$ as before, Figure 4 shows the posterior distributions of the two hyperparameters used to define the Gaussian kernel (4), σ and γ: the magnitude and the lengthscale

respectively.

The posteriors produced by each chain are very similar, indicating that all four chains converged to approximately the same value for both σ and γ. We see that the posterior means have converged far from the initial values $\boldsymbol{\theta} = (1, 1)$ proposed, suggesting that the chain has efficiently explored the state space. Given the successful convergence of all chains given rather poor initial conditions, the method proposed seems to be reliable.

4 Conclusion

In this report, we have explored the pseudo-marginal approach to a fully-Bayesian Gaussian process classification model described by Filippone and Girolami, 2014, and extended this to large datasets. Motivated by the desire to extend the logistic regression classifier to capture more complicated, non-linear relationships between observed variables, we outlined the flexible specification of the Gaussian process model.

Using the Laplace approximation of the posterior of the latent variables $\boldsymbol{f}$, and combining elliptical slice sampling and a pseudo-marginal approach to sample from the joint posterior of $\boldsymbol{f}, \boldsymbol{\theta}$, we have shown a way to overcome the difficulties of classifying with a GP model, and to express our uncertainty in predictions. Using the sparse *Informative Vector Machine* representation of Herbrich, Lawrence, and Seeger, 2003, we have proposed a possible method for extending the model for large datasets.

By comparing results against the logistic regression classifier, we have found that our model is capable of making superior predictions while trading off speed in fitting. We have given details on how we have implemented the procedure efficiently in software, and have made our own implementation available as an `R` package at `github.com/dannyjameswilliams/gpc`.

Bibliography

Beaumont, Mark A (2003). "Estimation of population growth or decline in genetically monitored populations". In: *Genetics* 164.3, pp. 1139–1160.

Dua, Dheeru and Casey Graff (2017). *UCI Machine Learning Repository*. URL: `http://archive.ics.uci.edu/ml`.

Eddelbuettel, Dirk and Conrad Sanderson (2014). "RcppArmadillo: Accelerating R with high-performance C++ linear algebra". In: *Computational Statistics & Data Analysis* 71, pp. 1054–1063.

Filippone, Maurizio and Mark Girolami (2014). "Pseudo-marginal Bayesian inference for Gaussian processes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.11, pp. 2214–2226.

Hensman, James, Alexander Matthews, and Zoubin Ghahramani (2015). "Scalable variational Gaussian process classification". In:

Herbrich, Ralf, Neil D Lawrence, and Matthias Seeger (2003). "Fast sparse Gaussian process methods: The informative vector machine". In: *Advances in neural information processing systems*, pp. 625–632.

Hernández-Lobato, Daniel and José Miguel Hernández-Lobato (2016). "Scalable gaussian process classification via expectation propagation". In: *Artificial Intelligence and Statistics*, pp. 168–176.

LogSumExp Trick. URL: `https://en.wikipedia.org/wiki/LogSumExp`.

Murray, Iain, Ryan Prescott Adams, and David JC MacKay (2010). "Elliptical slice sampling". In: *JMLR: WI&CP*.

Plummer, Martyn (2019). *rjags: Bayesian Graphical Models using MCMC*. R package version 4-10. URL: `https://CRAN.R-project.org/package=rjags`.

Williams, Christopher KI and Carl Edward Rasmussen (2006). *Gaussian processes for machine learning*. Vol. 2. 3. MIT press Cambridge, MA.