Published in *IEEE Trans. Pattern Analysis and Machine Intelligence*, 20(12) 1342-1351, (1998).

Bayesian Classification with Gaussian Processes

Christopher K. I. Williams

Department of Artificial Intelligence

University of Edinburgh

Edinburgh EH1 2QL, Scotland, UK

ckiw@dai.ed.ac.uk

David Barber

RWCP, Theoretical Foundation SNN

University of Nijmegen, 6525 EZ Nijmegen

The Netherlands

davidb@mbfys.kun.nl

January 17, 1999

Abstract

We consider the problem of assigning an input vector $\boldsymbol{x}$ to one of m classes by predicting $P(c|\boldsymbol{x})$ for $c = 1, \ldots, m$. For a two-class problem, the probability of class 1 given $\boldsymbol{x}$ is estimated by $\sigma(y(\boldsymbol{x}))$, where $\sigma(y) = 1/(1 + e^{-y})$. A Gaussian process prior is placed on $y(\boldsymbol{x})$, and is combined with the training data to obtain predictions for new $\boldsymbol{x}$ points.

We provide a Bayesian treatment, integrating over uncertainty in y and in the parameters that control the Gaussian process prior; the necessary integration over y is carried out using Laplace's approximation. The method is generalized to multi-class problems ($m > 2$) using the softmax function. We demonstrate the effectiveness of the method on a number of datasets.

Index terms: Gaussian processes, Classification problems, Parameter Uncertainty, Markov Chain Monte Carlo, Hybrid Monte Carlo, Bayesian classification.

1 Introduction

We consider the problem of assigning an input vector $\boldsymbol{x}$ to one out of m classes by predicting $P(c|\boldsymbol{x})$ for $c = 1, \ldots, m$. A classic example of this method is logistic regression. For a two-class problem, the probability of class 1 given $\boldsymbol{x}$ is estimated by $\sigma(\boldsymbol{w}^T\boldsymbol{x} + b)$, where $\sigma(y) = 1/(1 + e^{-y})$. However, this method is not at all "flexible", i.e. the discriminant surface is simply a hyperplane in $\boldsymbol{x}$-space. This problem can be overcome, to some extent, by expanding the input $\boldsymbol{x}$ into a set of basis functions $\{\phi(\boldsymbol{x})\}$, for example quadratic functions of the components of $\boldsymbol{x}$. For a high-dimensional input space there will be a large number of basis functions, each one with an associated parameter, and one risks "overfitting" the training data. This motivates a Bayesian treatment of the problem, where the priors on the parameters encourage smoothness in the model.

Putting priors on the parameters of the basis functions indirectly induces priors over the functions that can be produced by the model. However, it is possible (and we would argue, perhaps more natural) to put priors directly over the functions themselves. One advantage of function-space priors is that they can impose a general smoothness constraint without being tied to a limited number of basis functions. In the regression case where the task is to predict a real-valued output, it is possible to carry out *non-parametric* regression using Gaussian Processes (GPs); see, e.g. [25], [28]. The solution for the regression problem under a GP prior (and Gaussian noise model) is to place a kernel function on each training data point, with coefficients determined by solving a linear system. If the parameters $\boldsymbol{\theta}$ that describe the Gaussian process are unknown, Bayesian inference can be carried out for them, as described in [28].

The Gaussian Process method can be extended to classification problems by defining a GP over y, the input to the sigmoid function. This idea has been used by a number of authors, although previous treatments typically do not take a fully Bayesian approach, ignoring uncertainty in both the posterior distribution of y given the data, and uncertainty in the parameters $\boldsymbol{\theta}$. This paper attempts a fully Bayesian treatment of the problem, and also introduces a particular form of covariance function for the Gaussian process prior which, we believe, is useful from a modelling point of view.

The structure of the remainder of the paper is as follows: Section 2 discusses the use of Gaussian processes for regression problems, as this is essential background for the classification case. In Section 3 we describe the application of Gaussian processes to two-class classification problems, and extend this to multiple-class problems in section 4. Experimental results are presented in section 5, followed by a discussion in section 6. This paper is a revised and expanded version of [1].

2 Gaussian Processes for regression

It will be useful to first consider the regression problem, i.e. the prediction of a real valued output $y_* = y(\boldsymbol{x}_*)$ for a new input value $\boldsymbol{x}_*$, given a set of training data $\mathcal{D} = \{(\boldsymbol{x}_i, t_i),\ i = 1 \dots n\}$. This is of relevance because our strategy will be to transform the classification problem into a regression problem by dealing with the input values to the logistic transfer function.

A stochastic process prior over functions allows us to specify, given a set of inputs, $\boldsymbol{x}_1, \dots \boldsymbol{x}_n$, the distribution over their corresponding outputs $\boldsymbol{y} \stackrel{def}{=} (y(\boldsymbol{x}_1), y(\boldsymbol{x}_2), \dots, y(\boldsymbol{x}_n))$. We denote this prior over functions as $P(\boldsymbol{y})$, and similarly, $P(y_*, \boldsymbol{y})$ for the joint distribution including y_*. If we also specify $P(\boldsymbol{t}|\boldsymbol{y})$, the probability of observing the particular values $\boldsymbol{t} = (t_1, \dots t_n)^T$ given actual values $\boldsymbol{y}$ (i.e. a noise model) then we have that

$$
\begin{aligned}
P(y_*|\boldsymbol{t}) &= \int P(y_*, \boldsymbol{y}|\boldsymbol{t}) d\boldsymbol{y} && (1) \\
&= \frac{1}{P(\boldsymbol{t})} \int P(y_*|\boldsymbol{y}) P(\boldsymbol{y}) P(\boldsymbol{t}|\boldsymbol{y}) d\boldsymbol{y} && (2) \\
&= \int P(y_*|\boldsymbol{y}) P(\boldsymbol{y}|\boldsymbol{t}) d\boldsymbol{y} && (3)
\end{aligned}
$$

Hence the predictive distribution for y_* is found from the marginalization of the product of the prior and the noise model. Note that in order to make predictions it is not necessary to deal directly with priors over function space, only n- or $n+1$-dimensional joint densities. However, it is still not easy to carry out these calculations unless the densities involved have a special form.

If $P(\boldsymbol{t}|\boldsymbol{y})$ and $P(y_*, \boldsymbol{y})$ are Gaussian then $P(y_*|\boldsymbol{t})$ is a Gaussian whose mean and variance can be calculated using matrix computations involving matrices of size $n \times n$. Specifying $P(y_*, \boldsymbol{y})$ to be a multidimensional Gaussian (for all values of n and placements of the points $\boldsymbol{x}_*, \boldsymbol{x}_1, \dots \boldsymbol{x}_n$) means that the prior over functions is a Gaussian process. More formally, a stochastic process is a collection of random variables $\{Y(\boldsymbol{x})|\boldsymbol{x} \in X\}$ indexed by a set X. In our case X will be the input space with dimension d, the number of inputs. A GP is a stochastic process which can be fully specified by its mean function $\mu(\boldsymbol{x}) = E[Y(\boldsymbol{x})]$ and its covariance function $C(\boldsymbol{x}, \boldsymbol{x}') = E[(Y(\boldsymbol{x}) - \mu(\boldsymbol{x}))(Y(\boldsymbol{x}') - \mu(\boldsymbol{x}'))]$; any finite set of Y-variables will have a joint multivariate Gaussian distribution. Below we consider GPs which have $\mu(\boldsymbol{x}) \equiv 0$.

If we further assume that the noise model $P(\boldsymbol{t}|\boldsymbol{y})$ is Gaussian with mean zero and variance $\sigma^2 I$, then the predicted mean and variance at $\boldsymbol{x}_*$ are given by

$$\begin{aligned} \hat{y}(\boldsymbol{x}_*) &= \boldsymbol{k}^T(\boldsymbol{x}_*)(K + \sigma^2 I)^{-1}\boldsymbol{t} \\ \sigma^2_{\hat{y}}(\boldsymbol{x}_*) &= C(\boldsymbol{x}_*, \boldsymbol{x}_*) - \boldsymbol{k}^T(\boldsymbol{x}_*)(K + \sigma^2 I)^{-1}\boldsymbol{k}(\boldsymbol{x}_*), \end{aligned}$$

where $[K]_{ij} = C(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $\boldsymbol{k}(\boldsymbol{x}_*) = (C(\boldsymbol{x}_*, \boldsymbol{x}_1), \ldots, C(\boldsymbol{x}_*, \boldsymbol{x}_n))^T$ (see, e.g. [25]).

2.1 Parameterizing the covariance function

There are many reasonable choices for the covariance function. Formally, we are required to specify functions which will generate a non-negative definite covariance matrix for any set of points $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$. From a modelling point of view we wish to specify covariances so that points with nearby inputs will give rise to similar predictions. We find that the following covariance function works well:

$$C(\boldsymbol{x}, \boldsymbol{x}') = v_0 \exp\{-\frac{1}{2}\sum_{l=1}^{d} w_l (x_l - x_l')^2\} + v_1 \tag{4}$$

where x_l is the lth component of $\boldsymbol{x}$ and $\boldsymbol{\theta} = (\log v_0, \log v_1, \log w_1, \ldots, \log w_d)$ is the vector of parameters that are needed to define the covariance function. Note that $\boldsymbol{\theta}$ is analogous to the *hyper*parameters in a neural network. We define the parameters to be the log of the variables in equation (4) since these are positive scale-parameters. This covariance function can be obtained from a network of Gaussian radial basis functions in the limit of an infinite number of hidden units [27].

The w_l parameters in equation 4 allow a different length scale on each input dimension. For irrelevant inputs, the corresponding w_l will become small, and the model will ignore that input. This is closely related to the Automatic Relevance Determination (ARD) idea of MacKay [10] and Neal [15]. The v_0 variable specifies the overall scale of the prior. v_1 specifies the variance of a zero-mean offset which has a Gaussian distribution.

The Gaussian process framework allows quite a wide variety of priors over functions. For example, the Ornstein-Uhlenbeck process (with covariance function $C(x, x') = e^{-|x-x'|}$) has very rough sample paths which are not mean-square differentiable. On the other hand the squared exponential covariance function of equation 4 gives rise to an infinitely m.s. differentiable process. In general we believe that the GP method is a quite general-purpose route for imposing prior beliefs about the desired amount of smoothness. For reasonably high-dimensional problems, this needs to be combined with other modelling assumptions such as ARD. Another modelling assumption that may be used is to build up the covariance function as a sum of covariance functions, each one of which may depend on only some of the input variables (see section 3.3 for further details).

2.2 Dealing with parameters

Given a covariance function it is straightforward to make predictions for new test points. However, in practical situations we are unlikely to know which covariance function to use. One option is to choose a parametric family of covariance functions (with a parameter vector $\boldsymbol{\theta}$) and then either to estimate the parameters (for example, using the method of maximum likelihood) or to use a Bayesian approach where a posterior distribution over the parameters is obtained.

These calculations are facilitated by the fact that the log likelihood $l = \log P(\mathcal{D}|\boldsymbol{\theta})$ can be calculated analytically as

$$l = -\frac{1}{2}\log|\tilde{K}| - \frac{1}{2}\boldsymbol{t}^T\tilde{K}^{-1}\boldsymbol{t} - \frac{n}{2}\log 2\pi, \tag{5}$$

where $\tilde{K} = K + \sigma^2 I$ and $|\tilde{K}|$ denotes the determinant of $\tilde{K}$. It is also possible to express analytically the partial derivatives of the log likelihood with respect to the parameters

$$\frac{\partial l}{\partial \theta_i} = -\frac{1}{2}\mathrm{tr}\left(\tilde{K}^{-1}\frac{\partial \tilde{K}}{\partial \theta_i}\right) + \frac{1}{2}\boldsymbol{t}^T\tilde{K}^{-1}\frac{\partial \tilde{K}}{\partial \theta_i}\tilde{K}^{-1}\boldsymbol{t}, \tag{6}$$

(see, e.g. [11]).

Given l and its derivatives with respect to $\boldsymbol{\theta}$ it is straightforward to feed this information to an optimization package in order to obtain a local maximum of the likelihood.

In general one may be concerned about making point estimates when the number of parameters is large relative to the number of data points, or if some of the parameters may be poorly determined, or if there may be local maxima in the likelihood surface. For these reasons the Bayesian approach of defining a prior distribution over the parameters and then obtaining a posterior distribution once the data $\mathcal{D}$ has been seen is attractive. To make a prediction for a new test point $\boldsymbol{x}_*$ one simply averages over the posterior distribution $P(\boldsymbol{\theta}|\mathcal{D})$, i.e.

$$P(y_*|\mathcal{D}) = \int P(y_*|\boldsymbol{\theta}, \mathcal{D}) P(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta}. \tag{7}$$

For GPs it is not possible to do this integration analytically in general, but numerical methods may be used. If $\boldsymbol{\theta}$ is of sufficiently low dimension, then techniques involving grids in $\boldsymbol{\theta}$-space can be used.

If $\boldsymbol{\theta}$ is high-dimensional it is very difficult to locate the regions of parameter-space which have high posterior density by gridding techniques or importance sampling. In this case Markov chain Monte Carlo (MCMC) methods may be used. These work by constructing a Markov chain whose equilibrium distribution is the desired distribution $P(\boldsymbol{\theta}|\mathcal{D})$; the integral in equation 7 is then approximated using samples from the Markov chain.

Two standard methods for constructing MCMC methods are the Gibbs sampler and Metropolis-Hastings algorithms (see, e.g., [5]). However, the conditional parameter distributions are not amenable to Gibbs sampling if the covariance function has the form given by equation 4, and the Metropolis-Hastings algorithm does not utilize the derivative information that is available, which means that it tends to have an inefficient random-walk behaviour in parameter-space. Following the work of Neal [15] on Bayesian treatment of neural networks, Williams and Rasmussen [28] and Rasmussen [17] have used the Hybrid Monte Carlo (HMC) method of Duane *et al* [4] to obtain samples from $P(\boldsymbol{\theta}|\mathcal{D})$. The HMC algorithm is described in more detail in Appendix D.

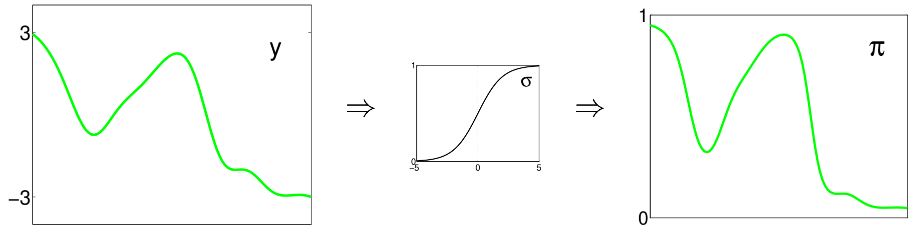

Figure 1: $\pi(\boldsymbol{x})$ is obtained from $y(\boldsymbol{x})$ by "squashing" it through the sigmoid function σ.

3 Gaussian Processes for two-class classification

For simplicity of exposition we will first present our method as applied to two-class problems; the extension to multiple classes is covered in section 4.

By using the logistic transfer function to produce an output which can be interpreted as $\pi(\boldsymbol{x})$, the probability of the input $\boldsymbol{x}$ belonging to class 1, the job of specifying a prior over functions π can be transformed into that of specifying a prior over the input to the transfer function, which we shall call the *activation*, and denote by y (see Figure 1). For the two-class problem we can use the logistic function $\pi(\boldsymbol{x}) = \sigma(y(\boldsymbol{x}))$ where $\sigma(y) = 1/(1 + e^{-y})$. We will denote the probability and activation corresponding to input $\boldsymbol{x}_i$ by π_i and y_i respectively. Fundamentally, the GP approaches to classification and regression problems are similar, except that the error model which is $t \sim \mathrm{N}(y, \sigma^2)$ in the regression case is replaced by $t \sim \mathrm{Bern}(\sigma(y))$. The choice of v_0 in equation 4 will affect how "hard" the classification is; i.e. if $\pi(\boldsymbol{x})$ hovers around 0.5 or takes on the extreme values of 0 and 1.

Previous and related work to this approach is discussed in section 3.3.

As in the regression case there are now two problems to address (a) making predictions with fixed parameters and (b) dealing with parameters. We shall discuss these issues in turn.

3.1 Making predictions with fixed parameters

To make predictions when using fixed parameters we would like to compute $\hat{\pi}_* = \int \pi_* P(\pi_*|\boldsymbol{t})\ d\pi_*$, which requires us to find $P(\pi_*|\boldsymbol{t}) = P(\pi(\boldsymbol{x}_*)|\boldsymbol{t})$ for a new input $\boldsymbol{x}_*$. This can be done by finding the distribution $P(y_*|\boldsymbol{t})$ (y_* is the activation of π_*) and then using the appropriate Jacobian to transform the distribution. Formally the equations for obtaining $P(y_*|\boldsymbol{t})$ are identical to equations 1, 2, and 3. However, even if we use a GP prior so that $P(y_*, \boldsymbol{y})$ is Gaussian, the usual expression for $P(\boldsymbol{t}|\boldsymbol{y}) = \prod_i \pi_i^{t_i}(1-\pi_i)^{1-t_i}$ for classification data (where the t's take on values of 0 or 1), means that the marginalization to obtain $P(y_*|\boldsymbol{t})$ is no longer analytically tractable.

Faced with this problem there are two routes that we can follow: (i) to use an analytic approximation to the integral in equations 1-3 or (ii) to use Monte Carlo methods, specifically MCMC methods, to approximate it. Below we consider an analytic approximation based on Laplace's approximation; some other approximations are discussed in section 3.3.

In Laplace's approximation, the integrand $P(y_*, \boldsymbol{y}|\boldsymbol{t}, \boldsymbol{\theta})$ is approximated by a Gaussian distribution centered at a maximum of this function with respect to $y_*, \boldsymbol{y}$ with an inverse covariance matrix given by $-\nabla\nabla \log P(y_*, \boldsymbol{y}|\boldsymbol{t}, \boldsymbol{\theta})$. Finding a maximum can be carried out using the Newton-Raphson iterative method on $\boldsymbol{y}$, which then allows the approximate distribution of y_* to be calculated. Details of the maximization procedure can be found in Appendix A.

3.2 Integration over the parameters

To make predictions we integrate the predicted probabilities over the posterior $P(\boldsymbol{\theta}|\boldsymbol{t}) \propto P(\boldsymbol{t}|\boldsymbol{\theta})P(\theta)$, as we saw in 2.2. For the regression problem $P(\boldsymbol{t}|\boldsymbol{\theta})$ can be calculated exactly using $P(\boldsymbol{t}|\boldsymbol{\theta}) = \int P(\boldsymbol{t}|\boldsymbol{y})P(\boldsymbol{y}|\boldsymbol{\theta})d\boldsymbol{y}$, but this integral is not analytically tractable for the classification problem. Let $\Psi = \log P(\boldsymbol{t}|\boldsymbol{y}) + \log P(\boldsymbol{y})$. Using $\log P(t_i|y_i) = t_i y_i - \log(1 + e^{y_i})$, we obtain

$$\Psi = \boldsymbol{t}^T\boldsymbol{y} - \sum_{i=1}^{n} \log(1 + e^{y_i}) - \frac{1}{2}\boldsymbol{y}^T K^{-1}\boldsymbol{y} - \frac{1}{2}\log|K| - \frac{n}{2}\log 2\pi. \tag{8}$$

By using Laplace's approximation about the maximum $\tilde{\boldsymbol{y}}$ we find that

$$\log P(\boldsymbol{t}|\boldsymbol{\theta}) \simeq \Psi(\tilde{\boldsymbol{y}}) - \frac{1}{2}\log|K^{-1} + W| + \frac{n}{2}\log 2\pi. \tag{9}$$

We denote the right-hand side of this equation by $\log P_a(\boldsymbol{t}|\boldsymbol{\theta})$ (where a stands for approximate).

The integration over $\boldsymbol{\theta}$-space also cannot be done analytically, and we employ a Markov Chain Monte Carlo method. Following Neal [15] and Williams and Rasmussen [28] we have used the Hybrid Monte Carlo (HMC) method of Duane *et al* [4] as described in Appendix D. We use $\log P_a(\boldsymbol{t}|\boldsymbol{\theta})$ as an approximation for $\log P(\boldsymbol{t}|\boldsymbol{\theta})$, and use broad Gaussian priors on the parameters.

3.3 Previous and related work

Our work on Gaussian processes for regression and classification developed from the observation in [15] that a large class of neural network models converge to GPs in the limit of an infinite number of hidden units. The computational Bayesian treatment of GPs can be easier than for neural networks. In the regression case an infinite number of weights are effectively integrated out, and one ends up dealing only with the (hyper)parameters. Results from [17] show that Gaussian processes for regression are comparable in performance to other state-of-the-art methods.

Non-parametric methods for classification problems can be seen to arise from the combination of two different strands of work. Starting from linear regression, McCullagh and Nelder [12] developed *generalized linear models* (GLMs). In the two-class classification context, this gives rise to logistic regression. The other strand of work was the the development of non-parametric smoothing for the regression problem. Viewed as a Gaussian process prior over functions this can be traced back at least as far as the work of Kolmogorov and Wiener in the 1940s. Gaussian process prediction is well known in the geostatistics field (see, e.g. [3]) where it is known as "kriging". Alternatively, by considering "roughness penalties" on functions, one can obtain spline methods; for recent overviews, see [25] and [8]. There is a close connection between the GP and roughness penalty views, as explored in [9]. By combining GLMs with non-parametric regression one obtains what we shall call a non-parametric GLM method for classification. Early references to this method include [21] and [16], and discussions can also be found in texts such as [8] and [25].

There are two differences between the non-parametric GLM method as it is usually described and a Bayesian treatment. Firstly, for fixed parameters the non-parametric GLM method ignores the uncertainty in y_* and hence the need to integrate over this (as described in section 3.1).

The second difference relates to the treatment of the parameters $\boldsymbol{\theta}$. As discussed in section 2.2, given parameters $\boldsymbol{\theta}$, one can either attempt to obtain a point estimate for the parameters or to carry out an integration over the posterior. Point estimates may be obtained by maximum likelihood estimation of $\boldsymbol{\theta}$, or by cross-validation or generalized cross-validation (GCV) methods, see e.g. [25, 8]. One problem with CV-type methods is that if the dimension of $\boldsymbol{\theta}$ is large, then it can be computationally intensive to search over a region/grid in parameter-space looking for the parameters that maximize the criterion. In a sense the HMC method described above are doing a similar search, but using gradient information[1], and carrying out averaging over the posterior distribution of parameters. In defence of (G)CV methods, we note Wahba's comments (e.g. in [26], referring back to [24]) that these methods may be more robust against an unrealistic prior.

One other difference between the kinds of non-parametric GLM models usually considered and our method is the exact nature of the prior that is used. Often the roughness penalties used are expressed in terms of a penalty on the kth derivative of $y(\boldsymbol{x})$, which gives rise to a power law power spectrum for the prior on $y(\boldsymbol{x})$. There can also be differences over parameterization of the covariance function; for example it is unusual to find parameters like those for ARD introduced in equation 4 in non-parametric GLM models. On the other hand, Wahba et al [26] have considered a smoothing spline analysis of variance (SS-ANOVA) decomposition. In Gaussian process terms, this builds up a prior on y as a sum of priors on each of the functions in the decomposition

$$y(\boldsymbol{x}) = \mu + \sum_{\alpha} y_{\alpha}(x_{\alpha}) + \sum_{\alpha,\beta} y_{\alpha\beta}(x_{\alpha}, x_{\beta}) + \dots . \tag{10}$$

The important point is that functions involving all orders of interaction (from univariate functions, which on their own give rise to an additive model) are included in this sum, up to the full interaction term which is the only one that

[1] It would be possible to obtain derivatives of the CV-score with respect to $\boldsymbol{\theta}$, but this has not, to our knowledge, been used in practice.

we are using. From a Bayesian point of view questions as to the kinds of priors that are appropriate is an interesting modelling issue.

There has also been some recent work which is related to the method presented in this paper. In section 3.1 we mentioned that it is necessary to approximate the integral in equations 1-3 and described the use of Laplace's approximation.

Following the preliminary version of this paper presented in [1], Gibbs and MacKay [7] developed an alternative analytic approximation, by using variational methods to find approximating Gaussian distributions that bound the marginal likelihood $P(\boldsymbol{t}|\boldsymbol{\theta})$ above and below. These approximate distributions are then used to predict $P(y_*|\boldsymbol{t}, \boldsymbol{\theta})$ and thus $\hat{\pi}(\boldsymbol{x}_*)$. For the parameters, Gibbs and MacKay estimated $\boldsymbol{\theta}$ by maximizing their lower bound on $P(\boldsymbol{t}|\boldsymbol{\theta})$.

It is also possible to use a fully MCMC treatment of the classification problem, as discussed in the recent paper of Neal [14]. His method carries out the integrations over the posterior distributions of $\boldsymbol{y}$ and $\boldsymbol{\theta}$ simultaneously. It works by generating samples from $P(\boldsymbol{y}, \boldsymbol{\theta}|\mathcal{D})$ in a two stage process. Firstly, for fixed $\boldsymbol{\theta}$, each of the n individual y_i's are updated sequentially using Gibbs sampling. This "sweep" takes time $O(n^2)$ once the matrix K^{-1} has been computed (in time $O(n^3)$), so it actually makes sense to perform quite a few Gibbs sampling scans between each update of the parameters, as this probably makes the Markov chain mix faster. Secondly, the parameters are updated using the Hybrid Monte Carlo method. To make predictions, one averages over the predictions made by each $\boldsymbol{y}, \boldsymbol{\theta}$ sample.

4 GPs for multiple-class classification

The extension of the preceding framework to multiple classes is essentially straightforward, although notationally more complex.

Throughout we employ a one-of-m class coding scheme[2], and use the multi-class analogue of the logistic function—the softmax function—to describe the class probabilities. The probability that an instance labelled by i is in class c is denoted by π_c^i, so that an upper index to denotes the example number, and a lower index the class label. Similarly, the activations associated with the probabilities are denoted by y_c^i. Formally, the softmax link function relates the activations and probabilities through

$$\pi_c^i = \frac{\exp y_c^i}{\Sigma_{c'} \exp y_{c'}^i}$$

which automatically enforces the constraint $\sum_c \pi_c^i = 1$. The targets are similarly represented by t_c^i, and are specified using a one-of-m coding.

The log likelihood takes the form $\mathcal{L} = \sum_{i,c} t_c^i \ln \pi_c^i$, which for the softmax link function gives

$$\mathcal{L} = \sum_{i,c} t_c^i \left(y_c^i - \ln \sum_{c'} \exp \pi_{c'}^i \right). \tag{11}$$

As for the two class case, we shall assume that the GP prior operates in activation space; that is we specify the correlations between the activations y_c^i.

One important assumption we make is that our prior knowledge is restricted to correlations between the activations of a particular class. Whilst there is no difficulty in extending the framework to include inter-class correlations, we have not yet encountered a situation where we felt able to specify such correlations.

[2]That is, the class is represented by a vector of length m with zero entries everywhere except for the correct component which contains 1.

Formally, the activation correlations take the form,

$$\langle y_c^i y_{c'}^{i'} \rangle = \delta_{c,c'} K_c^{i,i'} \tag{12}$$

where $K_c^{i,i'}$ is the i, i' element of the covariance matrix for the cth class. Each individual correlation matrix K_c has the form given by equation 4 for the two-class case. We shall use a separate set of parameters for each class. The use of m independent processes to perform the classification is redundant, but forcing the activations of one process to be (say) zero would introduce an arbitrary asymmetry into the prior.

For simplicity, we introduce the augmented vector notation,

$$\boldsymbol{y}_+ = \left(y_1^1, ..., y_1^n, y_1^*, y_2^1, ..., y_2^n, y_2^*, ..., y_m^1, ..., y_c^i, ..., y_m^*\right)$$

where, as in the two-class case, y_c^* denotes the activation corresponding to input $\boldsymbol{x}_*$ for class c; this notation is also used to define $\boldsymbol{t}_+$ and $\boldsymbol{\pi}_+$. In a similar manner, we define $\boldsymbol{y}$, $\boldsymbol{t}$ and $\boldsymbol{\pi}$ by excluding the values corresponding to the test point $\boldsymbol{x}_*$. Let $\boldsymbol{y}^* = (y_1^*, y_2^*, \ldots, y_m^*)$.

With this definition of the augmented vectors, the GP prior takes the form,

$$P(\boldsymbol{y}_+) \propto \exp\left\{-\frac{1}{2}\boldsymbol{y}_+^T (K^+)^{-1} \boldsymbol{y}_+\right\} \tag{13}$$

where, from equation 12, the covariance matrix K^+ is block diagonal in the matrices, $K_1^+, ..., K_m^+$. Each individual matrix K_c^+ expresses the correlations of activations within class c.

As in the two-class case, to use Laplace's approximation we need to find the mode of $P(\boldsymbol{y}_+|\boldsymbol{t})$. The procedure is described in Appendix C. As for the two-class case, we make predictions for $\boldsymbol{\pi}(\boldsymbol{x}^*)$ by averaging the softmax function over the Gaussian approximation to the posterior distribution of $\boldsymbol{y}^*$. At present, we simply estimate this integral using 1000 draws from a Gaussian random vector generator.

5 Experimental results

When using the Newton-Raphson algorithm, $\boldsymbol{\pi}$ was initialized each time with entries $1/m$, and iterated until the mean relative difference of the elements of W between consecutive iterations was less than 10^{-4}.

For the HMC algorithm, the same step size ε is used for all parameters, and should be as large as possible while keeping the rejection rate low. We have used a trajectory made up of $L = 20$ leapfrog steps, which gave a low correlation between successive states. The priors over parameters were set to be Gaussian with a mean of -3 and a standard deviation of 3. In all our simulations a step size $\varepsilon = 0.1$ produced a low rejection rate ($< 5\%$). The parameters corresponding to the w_l's were initialized to -2 and that for v_0 to 0. The sampling procedure was run for 200 iterations, and the first third of the run was discarded; this "burn-in" is intended to give the parameters time to come close to their equilibrium distribution. Tests carried out using the R-CODA package[3] on the examples in section 5.1 suggested that this was indeed effective in removing the transients, although we note that it is widely recognized (see, e.g. [2]) that determining when the equilibrium distribution has been reached is a difficult problem. Although the number of iterations used is much less than typically used for MCMC methods it should be remembered that (i) each iteration involves $L = 20$ leapfrog steps and (ii) that by using HMC we aim to reduce the "random walk" behaviour seen in methods such as the Metropolis

[3] Available from the Comprehensive R Archive Network at `http://www.ci.tuwien.ac.at`.

algorithm. Autocorrelation analysis for each parameter indicated, in general, that low correlation was obtained after a lag of a few iterations.

The MATLAB code which we used to run our experiments is available from `ftp://cs.aston.ac.uk/neural/willicki/gpclass/`.

5.1 Two classes

We have tried out our method on two well known two class classification problems, the Leptograpsus crabs and Pima Indian diabetes datasets[4]. We first rescale the inputs so that they have mean of zero and unit variance on the training set. Our Matlab implementations for the HMC simulations for both tasks each take several hours on a SGI Challenge machine (200MHz R10000), although good results can be obtained in much less time. We also tried a standard Metropolis MCMC algorithm for the Crabs problem, and found similar results, although the sampling by this method is somewhat slower than that for HMC.

The results for the Crabs and Pima tasks, together with comparisons with other methods (from [20] and [18]) are given in Tables 1 and 2 respectively. The tables also include results obtained for Gaussian processes using (a) estimation of the parameters by maximizing the penalised likelihood (found using 20 iterations of a scaled conjugate gradient optimiser) and (b) Neal's MCMC method. Details of the set-up used for Neal's method are given in Appendix E.

In the Leptograpsus crabs problem we attempt to classify the sex of crabs on the basis of five anatomical attributes, with an optional additional colour attribute. There are 50 examples available for crabs of each sex and colour, making a total of 200 labelled examples. These are split into a training set of

[4] Available from `http://markov.stats.ox.ac.uk/pub/PRNN`.

20 crabs of each sex and colour, making 80 training examples, with the other 120 examples used as the test set. The performance of our GP method is equal to the best of the other methods reported in [20], namely a 2 hidden unit neural network with direct input to output connections, a logistic output unit and trained with maximum likelihood (Network(1) in Table 1). Neal's method gave a very similar level of performance. We also found that estimating the parameters using maximum penalised likelihood (MPL) gave similar performance with less than a minute of computing time.

For the Pima Indians diabetes problem we have used the data as made available by Prof. Ripley, with his training/test split of 200 and 332 examples respectively [18]. The baseline error obtained by simply classifying each record as coming from a diabetic gives rise to an error of 33%. Again, ours and Neal's GP methods are comparable with the best alternative performance, with an error of around 20%. It is encouraging that the results obtained using Laplace's approximation and Neal's method are similar[5]. We also estimated the parameters using maximum penalised likelihood, rather than Monte Carlo integration. The performance in this case was a little worse, with 21.7% error, but for only 2 minutes computing time.

Analysis of the posterior distribution of the w parameters in the covariance function (equation 4) can be informative. Figure 5.1 plots the posterior marginal mean and 1 standard deviation error bars for each of the seven input dimensions. Recalling that the variables are scaled to have zero mean and unit variance, it would appear that variables 1 and 3 have the shortest lengthscales (and therefore

[5]The performance obtained by Gibbs and MacKay in [7] was similar. Their method made 4 errors in the crab task (with colour given), and 70 errors on the Pima dataset.

Method	Colour given	Colour not given
Neural Network(1)	3	3
Neural Network(2)	5	3
Linear Discriminant	8	8
Logistic regression	4	4
MARS (degree = 1)	8	4
PP regression (4 ridge functions)	3	6
Gaussian Process (Laplace Approximation, HMC)	3	3
Gaussian Process (Laplace Approximation, MPL)	4	3
Gaussian Process (Neal's method)	4	3

Table 1: Number of test errors for the Leptograpsus crabs task. Comparisons are taken from from Ripley (1996) and Ripley (1994) respectively. Network(2) used two hidden units and the predictive approach (Ripley, 1993) which uses Laplace's approximation to weight each network local minimum.

the most variability) associated with them.

5.2 Multiple classes

Due to the rather long time taken to run our code, we were only able to test it on relatively small problems, by which we mean only a few hundred data points and several classes. Furthermore, we found that a full Bayesian integration over possible parameter settings was beyond our computational means, and we therefore had to be satisfied with a maximum penalised likelihood approach.

Method	Pima Indian diabetes
Neural Network	75+
Linear Discriminant	67
Logistic Regression	66
MARS (degree = 1)	75
PP regression (4 ridge functions)	75
Gaussian Mixture	64
Gaussian Process (Laplace Approximation, HMC)	68
Gaussian Process (Laplace Approximation, MPL)	69
Gaussian Process (Neal's method)	68

Table 2: Number of test errors on the Pima Indian diabetes task. Comparisons are taken from from Ripley (1996) and Ripley (1994) respectively. The neural network had one hidden unit and was trained with maximum likelihood; the results were worse for nets with two or more hidden units (Ripley, 1996).

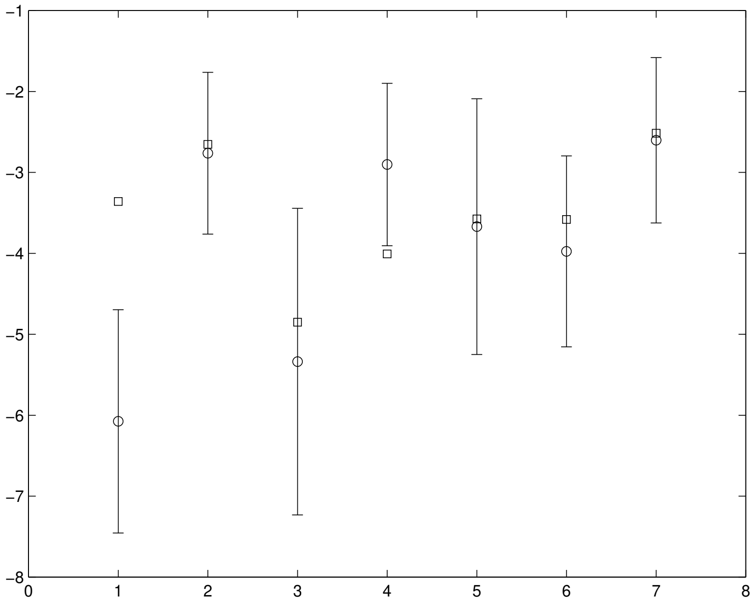

Figure 2: Plot of the $\log w$ parameters for the Pima dataset. The circle indicates the posterior marginal mean obtained from the HMC run (after burn-in), with one standard deviation error bars. The square symbol shows the $\log w$-parameter values found by maximizing the penalized likelihood. The variables are 1. the number of pregnancies, 2. plasma glucose concentration, 3. diastolic blood pressure, 4. triceps skin fold thickness, 5. body mass index, 6. diabetes pedigree function, 7. age. For comparison, Wahba *et al* (1995) using generalized linear regression, found that variables 1, 2 5 and 6 were the most important.

Rather than using the potential and its gradient in a HMC routine, we now simply used them as inputs to a scaled conjugate gradient optimiser (based on [13]) instead, attempting to find a mode of the class posterior, rather than to average over the posterior distribution.

We tested the multiple class method on the Forensic Glass dataset described in [18]. This is a dataset of 214 examples with 9 inputs and 6 output classes. Because the dataset is so small, the performance is estimated from using 10-fold cross validation. Computing the penalised maximum likelihood estimate of our multiple GP method took approximately 24 hours on our SGI Challenge and gave a classification error rate of 23.3%. As we see from Table 3, this is comparable to the best of the other methods. The performance of Neal's method is surprisingly poor; this may be due to the fact that we allow separate parameters for each of the y processes, while these are constrained to be equal in Neal's code. There are also small but perhaps significant differences in the specification of the prior (see Appendix E for details).

6 Discussion

In this paper we have extended the work of Williams and Rasmussen [28] to classification problems, and have demonstrated that it performs well on the datasets we have tried. We believe that the kinds of Gaussian process prior we have used are more easily interpretable than models (such as neural networks) in which the priors are on the parameterization of the function space. For example, the posterior distribution of the ARD parameters (as illustrated in Figure 5.1 for the Pima Indians diabetes problem) indicates the relative importance of

Method	Forensic Glass
Neural Network (4HU)	23.8%
Linear Discriminant	36%
MARS (degree = 1)	32.2%
PP regression (5 ridge functions)	35%
Gaussian Mixture	30.8%
Decision Tree	32.2%
Gaussian Process (LA, MPL)	23.3%
Gaussian Process (Neal's method)	31.8%

Table 3: Percentage of test error for the Forensic Glass problem. See Ripley (1996) for details of the methods.

various inputs. This interpretability should also facilitate the incorporation of prior knowledge into new problems.

There are quite strong similarities between GP classifiers and support-vector machines (SVMs) [23]. The SVM uses a covariance kernel, but differs from the GP approach by using a different data fit term (the maximum margin), so that the optimal $\boldsymbol{y}$ is found using quadratic programming. The comparison of these two algorithms is an interesting direction for future research.

A problem with methods based on GPs is that they require computations (trace, determinants and linear solutions) involving $n \times n$ matrices, where n is the number of training examples, and hence run into problems on large datasets. We have looked into methods using Bayesian numerical techniques to calculate the trace and determinant [22, 6], although we found that these techniques did not work well for the (relatively) small size problems on which we tested our methods. Computational methods used to speed up the quadratic programming problem for SVMs may also be useful for the GP classifier problem. We are also investigating the use of different covariance functions and improvements on the approximations employed.

Acknowledgements

We thank Prof. B. Ripley for making available the Leptograpsus crabs, Pima Indian diabetes and Forensic Glass datasets. This work was partially supported by EPSRC grant GR/J75425, *Novel Developments in Learning Theory for Neural Networks*, and much of the work was carried out at Aston University. The authors gratefully acknowledge the hospitality provided by the Isaac Newton Institute for Mathematical Sciences (Cambridge, UK) where this paper was written up. We thank Mark Gibbs, David MacKay and Radford Neal for helpful discussions, and the anonymous referees for their comments which helped improve the paper.

Appendix A: Maximizing $P(\boldsymbol{y}_+|\boldsymbol{t})$: Two-class case

We describe how to find iteratively the vector $\boldsymbol{y}_+$ so that $P(\boldsymbol{y}_+|\boldsymbol{t})$ is maximized. This material is also covered in [8] §5.3.3 and [25] §9.2. We provide it here for completeness and so that the terms in equation 9 are well-defined.

Let $\boldsymbol{y}_+$ denote $(y_*, \boldsymbol{y})$, the complete set of activations. By Bayes' theorem $\log P(\boldsymbol{y}_+|\boldsymbol{t}) = \log P(\boldsymbol{t}|\boldsymbol{y}) + \log P(\boldsymbol{y}_+) - \log P(\boldsymbol{t})$, and let $\Psi_+ = \log P(\boldsymbol{t}|\boldsymbol{y}) + \log P(\boldsymbol{y}_+)$. As $P(\boldsymbol{t})$ does not depend on $\boldsymbol{y}_+$ (it is just a normalizing factor), the maximum of $P(\boldsymbol{y}_+|\boldsymbol{t})$ is found by maximizing Ψ_+ with respect to $\boldsymbol{y}_+$. Using

$\log P(t_i|y_i) = t_i y_i - \log(1+e^{y_i})$, we obtain

$$\Psi_+ = \boldsymbol{t}^T \boldsymbol{y} - \sum_{i=1}^{n} \log(1+e^{y_i}) - \frac{1}{2}\boldsymbol{y}_+^T K_+^{-1} \boldsymbol{y}_+ - \frac{1}{2}\log|K_+| - \frac{n+1}{2}\log 2\pi \quad (14)$$

where K_+ is the covariance matrix of the GP evaluated at $\boldsymbol{x}_1, \ldots \boldsymbol{x}_n, \boldsymbol{x}_*$. Ψ is defined similarly in equation 8. K_+ can be partitioned in terms of an $n \times n$ matrix K, a $n \times 1$ vector $\boldsymbol{k}$ and a scalar k_*, viz.

$$K_+ = \begin{pmatrix} K & \boldsymbol{k} \\ \boldsymbol{k}^T & k_* \end{pmatrix}. \quad (15)$$

As y_* only enters into equation 14 in the quadratic prior term and has no data point associated with it, maximizing Ψ_+ with respect to $\boldsymbol{y}_+$ can be achieved by first maximizing Ψ with respect to $\boldsymbol{y}$ and then doing the further quadratic optimization to determine y_*. To find a maximum of Ψ we use the Newton-Raphson iteration $\boldsymbol{y}^{new} = \boldsymbol{y} - (\nabla\nabla\Psi)^{-1}\nabla\Psi$. Differentiating equation 8 with respect to $\boldsymbol{y}$ we find

$$\nabla\Psi = (\boldsymbol{t} - \boldsymbol{\pi}) - K^{-1}\boldsymbol{y} \quad (16)$$

$$\nabla\nabla\Psi = -K^{-1} - W \quad (17)$$

where the 'noise' matrix is given by $W = \mathrm{diag}(\pi_1(1-\pi_1), .., \pi_n(1-\pi_n))$. This results in the iterative equation,

$$\boldsymbol{y}^{new} = (K^{-1} + W)^{-1} W(\boldsymbol{y} + W^{-1}(\boldsymbol{t} - \boldsymbol{\pi})). \quad (18)$$

To avoid unnecessary inversions, it is usually more convenient to rewrite this in the form

$$\boldsymbol{y}^{new} = K(I + WK)^{-1}(W\boldsymbol{y} + (\boldsymbol{t} - \boldsymbol{\pi})). \quad (19)$$

Note that $-\nabla\nabla\Psi$ is always positive definite, so that the optimization problem is convex.

Given a converged solution $\tilde{\boldsymbol{y}}$ for $\boldsymbol{y}$, y_* can easily be found using $y_* = \boldsymbol{k}^T K^{-1}\tilde{\boldsymbol{y}} = \boldsymbol{k}^T(\boldsymbol{t} - \tilde{\boldsymbol{\pi}})$, as $K^{-1}\tilde{\boldsymbol{y}} = (\boldsymbol{t} - \boldsymbol{\pi})$ from equation 16. $\mathrm{var}(y_*)$ is given by $(K_+^{-1} + W_+)^{-1}_{(n+1)(n+1)}$, where W_+ is the W with a zero appended in the $(n+1)$th diagonal position. Given the mean and variance of y_* it is then easy to find $\hat{\pi}_* = \int \pi_* P(\pi_*|\boldsymbol{t}) d\pi_*$, the mean of the distribution of $P(\pi_*|\boldsymbol{t})$. In order to calculate the Gaussian integral over the logistic sigmoid function, we employ an approximation based on the expansion of the sigmoid function in terms of the error function. As the Gaussian integral of an error function is another error function, this approximation is fast to compute. Specifically, we use a basis set of five scaled error functions to interpolate the logistic sigmoid at chosen points[6]. This gives an accurate approximation (to 10^{-4}) to the desired integral with a small computational cost.

The justification of Laplace's approximation in our case is somewhat different from the argument usually put forward, e.g. for asymptotic normality of the maximum likelihood estimator for a model with a finite number of parameters. This is because the dimension of the problem grows with the number of data points. However, if we consider the "infill asymptotics" (see, e.g. [3]), where the number of data points in a *bounded* region increases, then a local average of the training data at any point $\boldsymbol{x}$ will provide a tightly localized estimate for $\pi(\boldsymbol{x})$ and hence $y(\boldsymbol{x})$ (this reasoning parallels more formal arguments found in [29]). Thus we would expect the distribution $P(\boldsymbol{y})$ to become more Gaussian with increasing data.

[6]In detail, we used the basis functions $\mathrm{erf}(\lambda x))$ for $\lambda = [0.41,\ 0.4,\ 0.37,\ 0.44,\ 0.39]$. These were used to interpolate $\sigma(x)$ at $x = [0,\ 0.6,\ 2,\ 3.5,\ 4.5,\ \infty]$.

Appendix B: Derivatives of $\log P_a(\boldsymbol{t}|\boldsymbol{\theta})$ wrt $\boldsymbol{\theta}$.

For both the HMC and MPL methods we require the derivative of $l_a = \log P_a(\boldsymbol{t}|\boldsymbol{\theta})$ with respect to components of $\boldsymbol{\theta}$, for example θ_k. This derivative will involve two terms, one due to explicit dependencies of $l_a = \Psi(\tilde{\boldsymbol{y}}) - \frac{1}{2}\log|K^{-1}+W| + \frac{n}{2}\log 2\pi$ on θ_k, and also because a change in $\boldsymbol{\theta}$ will cause a change in $\tilde{\boldsymbol{y}}$. However, as $\tilde{\boldsymbol{y}}$ is chosen so that $\nabla\Psi(\boldsymbol{y})|_{\boldsymbol{y}=\tilde{\boldsymbol{y}}} = 0$, we obtain

$$\frac{\partial l_a}{\partial \theta_k} = \left.\frac{\partial l_a}{\partial \theta_k}\right|_{explicit} - \frac{1}{2}\sum_{i=1}^{n} \frac{\partial \log|K^{-1}+W|}{\partial \tilde{y}_i} \frac{\partial \tilde{y}_i}{\partial \theta_k}. \tag{20}$$

The dependence of $|K^{-1}+W|$ on $\tilde{\boldsymbol{y}}$ arises through the dependence of W on $\tilde{\boldsymbol{\pi}}$, and hence $\tilde{\boldsymbol{y}}$. By differentiating $\tilde{\boldsymbol{y}} = K(\boldsymbol{t} - \tilde{\boldsymbol{\pi}})$, one obtains

$$\frac{\partial \tilde{\boldsymbol{y}}}{\partial \theta_k} = (I + KW)^{-1}\frac{\partial K}{\partial \theta_k}(\boldsymbol{t} - \tilde{\boldsymbol{\pi}}), \tag{21}$$

and hence the required derivative can be calculated.

Appendix C: Maximizing $P(\boldsymbol{y}_+|\boldsymbol{t})$: Multiple-class case

The GP prior and likelihood, defined by equations 13 and 11, define the posterior distribution of activations, $P(\boldsymbol{y}_+|\boldsymbol{t})$. As in Appendix A we are interested in a Laplace approximation to this posterior, and therefore need to find the mode with respect to $\boldsymbol{y}_+$. Dropping unnecessary constants, the multi-class analogue of equation 14 is

$$\Psi_+ = -\frac{1}{2}\boldsymbol{y}_+^T K_+^{-1}\boldsymbol{y}_+ - \frac{1}{2}\log|K_+| + \boldsymbol{t}^T\boldsymbol{y} - \sum_i \ln\sum_c \exp y_c^i.$$

By the same principle as in Appendix A, we define Ψ by analogy with equation 8, and first optimize Ψ with respect to $\boldsymbol{y}$, afterwards performing the quadratic optimization of Ψ_+ with respect to $\boldsymbol{y}_*$.

In order to optimize Ψ with respect to $\boldsymbol{y}$, we make use of the Hessian given by

$$\nabla\nabla\Psi = -K^{-1} - W, \tag{22}$$

where K is the $mn \times mn$ block-diagonal matrix with blocks K_c, $c = 1, \ldots, m$. Although this is in the same form as for the two class case, equation 17, there is a slight change in the definition of the 'noise' matrix, W. A convenient way to define W is by introducing the matrix Π which is a $mn \times n$ matrix of the form $\Pi = (\mathrm{diag}(\pi_1^1..\pi_1^n), .., \mathrm{diag}(\pi_m^1..\pi_m^n)$. Using this notation, we can write the noise matrix in the form of a diagonal matrix and an outer product,

$$W = -\mathrm{diag}(\pi_1^1..\pi_1^n, .., \pi_m^1..\pi_m^n) + \Pi\Pi^T. \tag{23}$$

As in the two-class case, we note that $-\nabla\nabla\Psi$ is again positive definite, so that the optimization problem is convex.

The update equation for iterative optimization of Ψ with respect to the activations $\boldsymbol{y}$ then follows the same form as that given by equation 18. The advantage of the representation of the noise matrix in equation 23 is that we can then invert matrices and find their determinants using the identities,

$$(A + \Pi\Pi^T)^{-1} = A^{-1} - A^{-1}\Pi(I_n + \Pi^T A^{-1}\Pi)^{-1}\Pi^T A^{-1} \tag{24}$$

and

$$\det(A + \Pi\Pi^T) = \det(A)\det(I_n + \Pi^T A^{-1}\Pi) \tag{25}$$

where $A = K^{-1} + \mathrm{diag}(\pi_1^1..\pi_m^n)$. As A is block-diagonal, it can be inverted blockwise. Thus, rather than requiring determinants and inverses of a $mn \times mn$

matrix, we only need to carry out expensive matrix computations on $n \times n$ matrices. The resulting update equations for $\boldsymbol{y}$ are then of the same form as given in equation 18, where the noise matrix and covariance matrices are now in their multiple class form.

Essentially, these are all the results needed to generalize the method to the multiple-class problem. Although, as we mentioned above, the time complexity of the problem does not scale with the m^3, but rather m (due to the identities in equations 24, 25), calculating the function and its gradient is still rather expensive. We have experimented with several methods of mode finding for the Laplace approximation. The advantage of the Newton iteration method is its fast quadratic convergence. An integral part of each Newton step is the calculation of the inverse of a matrix M acting upon a vector, i.e. $M^{-1}\boldsymbol{b}$. In order to speed up this particular step, we used a conjugate gradient (CG) method to solve iteratively the corresponding linear system $M\boldsymbol{z} = \boldsymbol{b}$. As we repeatedly need to solve the system (because W changes as $\boldsymbol{y}$ is updated), it saves time not to run the CG method to convergence each time it is called. In our experiments the CG algorithm was terminated when $1/n \sum_{i=1}^{n} |r_i| < 10^{-9}$, where $\boldsymbol{r} = M\boldsymbol{z} - \boldsymbol{b}$.

The calculation of the derivative of $\log P_a(\boldsymbol{t}|\boldsymbol{\theta})$ wrt $\boldsymbol{\theta}$ in the multiple-class case is analogous to the two-class case described in Appendix B.

Appendix D: Hybrid Monte Carlo

HMC works by creating a fictitious dynamical system in which the parameters are regarded as position variables, and augmenting these with momentum variables $\boldsymbol{p}$. The purpose of the dynamical system is to give the parameters "inertia" so that random-walk behaviour in $\boldsymbol{\theta}$-space can be avoided. The total energy, H, of the system is the sum of the kinetic energy, $K = \boldsymbol{p}^T \boldsymbol{p}/2$ and the potential energy, E. The potential energy is defined such that $p(\boldsymbol{\theta}|D) \propto \exp(-E)$, i.e. $E = -\log P(\boldsymbol{t}|\boldsymbol{\theta}) - \log P(\boldsymbol{\theta})$. We sample from the joint distribution for $\boldsymbol{\theta}$ and $\boldsymbol{p}$ given by $P(\boldsymbol{\theta}, \boldsymbol{p}) \propto \exp(-E - K)$; the marginal of this distribution for $\boldsymbol{\theta}$ is the required posterior. A sample of parameters from the posterior can therefore be obtained by simply ignoring the momenta.

Sampling from the joint distribution is achieved by two steps: (i) finding new points in phase space with near-identical energies H by simulating the dynamical system using a discretised approximation to Hamiltonian dynamics, and (ii) changing the energy H by Gibbs sampling the momentum variables.

Hamilton's first order differential equations for H are approximated using the leapfrog method which requires the derivatives of E with respect to $\boldsymbol{\theta}$. Given a Gaussian prior on $\boldsymbol{\theta}$, $\log P(\boldsymbol{\theta})$ is straightforward to differentiate. The derivative of $\log P_a(\boldsymbol{t}|\boldsymbol{\theta})$ is also straightforward, although implicit dependencies of $\tilde{\boldsymbol{y}}$ (and hence $\tilde{\boldsymbol{\pi}}$) on $\boldsymbol{\theta}$ must be taken into account as described in Appendix B. The calculation of the energy can be quite expensive as for each new $\boldsymbol{\theta}$, we need to perform the maximization required for Laplace's approximation, equation 9. This proposed state is then accepted or rejected using the Metropolis rule depending on the final energy H^* (which is not necessarily equal to the initial energy H because of the discretization).

Appendix E: Simulation set-up for Neal's code

We used the `fbm` software available from

`http://www.cs.utoronto.ca/~radford/fbm.software.html`. For example, the commands used to run the Pima example are

```
gp-spec pima1.log 7 1 - - 0.1 / 0.05:0.5 x0.2:0.5:1

model-spec pima1.log binary

pima1.log 7 1 2 / pima1.tr@1:200 . mypima.te@1:332 .

gp-gen pima1.log fix 0.5 1

mc-spec pima1.log repeat 4 scan-values 200 heatbath hybrid 6 0.5

gp-mc pima1.log 500
```

which follow closely the example given in Neal's documentation.

The `gp-spec` command specifies the form of the Gaussian process, and in particular the priors on the parameters v_0 and the w's (see equation 4). The expression `0.05:0.5` specifies a Gamma-distribution prior on v_0, and `x0.2:0.5:1` specifies a hierarchical Gamma prior on the w's. Note that a "jitter" of 0.1 is also specified on the prior covariance function; this improves conditioning of the covariance matrix.

The `mc-spec` command gives details of the MCMC updating procedure. It specifies 4 repetitions of 200 scans of the y values followed by 6 HMC updates of the parameters (using a step-size adjustment factor of 0.5). `gp-mc` specifies that this is sequence is carried out 500 times.

We aimed for a rejection rate of around 5%. If this was exceeded, the stepsize reduction factor was reduced and the simulation run again.

References

[1] D. Barber and C. K. I Williams. Gaussian Processes for Bayesian Classification via Hybrid Monte Carlo. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*. MIT Press, 1997.

[2] M. K. Cowles and B. P. Carlin. Markov-Chain Monte-Carlo Convergence Diagnostics—A Comparative Review. *J. American Stat. Assoc.*, 91:883–904, 1996.

[3] N. A. C. Cressie. *Statistics for Spatial Data*. Wiley, New York, 1993.

[4] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195:216–222, 1987.

[5] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis*. Chapman and Hall, London, 1995.

[6] M. Gibbs and D. J. C. MacKay. Efficient Implementation of Gaussian Processes. Draft manuscript, available from `http://wol.ra.phy.cam.ac.uk/mackay/homepage.html.`, 1997.

[7] M. Gibbs and D. J. C. MacKay. Variational Gaussian Process Classifiers. Draft manuscript, available via `http://wol.ra.phy.cam.ac.uk/mackay/homepage.html.`, 1997.

[8] P. J. Green and B. W. Silverman. *Nonparametric regression and generalized linear models*. Chapman and Hall, London, 1994.

[9] G. Kimeldorf and G. Wahba. A correspondence between Bayesian estimation of stochastic processes and smoothing by splines. *Annals of Mathematical Statistics*, 41:495–502, 1970.

[10] D. J. C. MacKay. Bayesian Methods for Backpropagation Networks. In J. L. van Hemmen, E. Domany, and K. Schulten, editors, *Models of Neural Networks II.* Springer, 1993.

[11] K. V. Mardia and R. J. Marshall. Maximum likelihood estimation for models of residual covariance in spatial regression. *Biometrika*, 71(1):135–146, 1984.

[12] McCullagh, P. and Nelder, J. *Generalized Linear Models.* Chapman and Hall, 1983.

[13] M. Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6(4):525–533, 1993.

[14] R. M. Neal. Monte Carlo Implementation of Gaussian Process Models for Bayesian Regression and Classification. Technical Report 9702, Department of Statistics, University of Toronto, 1997. Available from `http://www.cs.toronto.edu/~radford/`.

[15] Neal, R. M. *Bayesian Learning for Neural Networks.* Springer, New York, 1996. Lecture Notes in Statistics 118.

[16] F. O'Sullivan, B. S. Yandell, and W. J. Raynor. Automatic Smoothing of Regression Functions in Generalized Linear Models. *Journal of the American Statistical Association*, 81:96–103, 1986.

[17] C. E. Rasmussen. *Evaluation of Gaussian Processes and Other Methods for Non-linear Regression.* PhD thesis, Dept. of Computer Science, University of Toronto, 1996. Available from `http://www.cs.utoronto.ca/~carl/`.

[18] B. Ripley. *Pattern Recognition and Neural Networks.* Cambridge University Press, Cambridge, UK, 1996.

[19] B. D. Ripley. Statistical aspects of neural networks. In O. E. Barndorff-Nielsen, J. L. Jensen, and W. S. Kendall, editors, *Networks and Chaos—Statistical and Probabilistic Aspects*, pages 40–123. Chapman and Hall, 1993.

[20] B. D. Ripley. Flexible Non-linear Approaches to Classification. In V. Cherkassy, J. H. Friedman, and H. Wechsler, editors, *From Statistics to Neural Networks*, pages 105–126. Springer, 1994.

[21] B. W. Silverman. Density Ratios, Empirical Likelihood and Cot Death. *Applied Statistics*, 27(1):26–33, 1978.

[22] J. Skilling. Bayesian numerical analysis. In W. T. Grandy, Jr. and P. Milonni, editors, *Physics and Probability.* Cambridge University Press, 1993.

[23] V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer Verlag, New York, 1995.

[24] G. Wahba. A Comparison of GCV and GML for Choosing the Smoothing Parameter in the Generalized Spline Smoothing Problem. *Annals of Statistics*, 13:1378–1402, 1985.

[25] G. Wahba. *Spline Models for Observational Data.* Society for Industrial and Applied Mathematics, 1990. CBMS-NSF Regional Conference series in applied mathematics.

[26] G. Wahba, C. Gu, Y. Wang, and R. Chappell. Soft Classification, a.k.a. Risk Estimation, via Penalized Log Likelihood and Smoothing Spline Analysis of Variance. In D. H. Wolpert, editor, *The Mathematics of Generalization.* Addison-Wesley, 1995. Proceedings volume XX in the Santa Fe Institute Studies in the Sciences of Complexity.

[27] C. K. I. Williams. Computing with infinite networks. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*. MIT Press, 1997.

[28] C. K. I. Williams and C. E. Rasmussen. Gaussian processes for regression. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 514–520. MIT Press, 1996.

[29] S. J. Yakowitz and F. Szidarovszky. A Comparison of Kriging with Nonparametric Regression Methods. *J. Multivariate Analysis*, 16:21–53, 1985.