Brief Papers

Variational Gaussian Process Classifiers

Mark N. Gibbs and David J. C. MacKay

***Abstract*—Gaussian processes are a promising nonlinear regression tool, but it is not straightforward to solve classification problems with them. In this paper the variational methods of Jaakkola and Jordan are applied to Gaussian processes to produce an efficient Bayesian binary classifier.**

***Index Terms*—Bound, covariance function, evidence, hyperparameter, marginalization, normalizing constant, posterior probability.**

I. Introduction

ASSUME that we have some data $\mathcal{D}$ which consist of inputs $\{\mathbf{x}_n\}_{n=1}^N$ in some space, real or discrete, and corresponding targets t_n which are binary categorical variables. We shall model data using a Bayesian conditional classifier which predicts t conditional on $\mathbf{x}$. We assume the existence of a function $a(\mathbf{x})$ which models the "logit" $\log(P(t=1|\mathbf{x})/P(t=0|\mathbf{x}))$ as a function of $\mathbf{x}$. Thus

$$P(t=1|\mathbf{x}, a(\mathbf{x})) = \frac{1}{1+\exp(-a(\mathbf{x}))}. \tag{1}$$

To complete the model we place a prior distribution over the unknown function $a(\mathbf{x})$. This can be done in two ways. In the standard parametric approach, $a(\mathbf{x})$ is a parameterized function $a(\mathbf{x};\ \mathbf{w})$ where the parameters $\mathbf{w}$ might be, say, the weights of a neural network or the coefficients of a linear expansion $a(\mathbf{x};\ \mathbf{w}) = \sum_h w_h \phi_h(\mathbf{x})$. We place a prior probability distribution $P(\mathbf{w})$ over the parameters which is traditionally taken to be Gaussian [18].

In the alternative Gaussian process approach ([14], [18]–[20]), we model $a(\mathbf{x})$ directly using a Gaussian process. This involves modeling the joint distribution of $\{a(\mathbf{x}_n)\}$ with a Gaussian distribution

$$P(\mathbf{a}_N|\Theta) = \frac{1}{Z_{\text{gp}}} \exp\left(-\frac{1}{2}\mathbf{a}_N^T \mathbf{C}_N^{-1} \mathbf{a}_N\right) \tag{2}$$

where $\mathbf{a}_N = (a(\mathbf{x}_1), a(\mathbf{x}_2), \ldots, a(\mathbf{x}_N))$ and $\mathbf{C}_N$ is an appropriate positive definite covariance matrix that is a function of the inputs $\{\mathbf{x}_n\}$ and a set of hyperparameters Θ. Most parametric models are in fact special cases of Gaussian processes, with the covariance matrix depending on the details of the choice of basis functions $\phi_h(\mathbf{x})$ and the prior $P(\mathbf{w})$ [13]. Efficient methods for implementing Gaussian processes are described in [2] and [3].

For classification models there are two well-established approaches to Bayesian inference: Gaussian approximations centred on the posterior modes [8], [9] and Monte Carlo methods [13]. Barber and Williams [1] have implemented classifiers based on Gaussian process priors using Laplace approximations. In [14] Gaussian process classifiers are implemented using a Monte Carlo approach.

In this paper another approach is suggested based on the methods of Jaakkola and Jordan [6], [7]. We obtain tractable upper and lower bounds for the unnormalized posterior density $P(\{t\}|\mathbf{a})P(\mathbf{a})$. These bounds are parameterized by variational parameters which are adjusted in order to obtain the tightest possible fit. Using the normalized versions of the optimized bounds we then compute approximations to the predictive distributions. This is similiar to an ensemble-learning approach [4], [11] in that we are adapting approximations to the posterior distribution of unknown variables.

II. Variational Gaussian Process Model

Let us look at the Gaussian process approach in more detail. We wish to make predictions at new points given the data, i.e., we want to find the predictive distribution $P(t_{N+1}|\mathbf{x}_{N+1}, \mathcal{D})$ for a new input point $\mathbf{x}_{N+1}$. This can be written

$$\begin{aligned} &P(t_{N+1}=1|\mathbf{x}_{N+1}, \mathcal{D}) \\ &\quad = \int P(t_{N+1}=1|a(\mathbf{x}_{N+1}))P(a(\mathbf{x}_{N+1})|\mathbf{x}_{N+1}, \mathcal{D}) \\ &\qquad \cdot \mathrm{d}a(\mathbf{x}_{N+1}). \end{aligned} \tag{3}$$

The first term in the integrand is a sigmoid [see (1)]. The second term can found by integrating out the dependence on $\{a(\mathbf{x}_n)\}$ (for $n = 1 \ldots N$) in the joint posterior distribution $P(\mathbf{a}_{N+1}|\mathbf{x}_{N+1}, \mathcal{D})$

$$P(a(\mathbf{x}_{N+1})|\mathbf{x}_{N+1}, \mathcal{D}) = \int P(\mathbf{a}_{N+1}|\mathbf{x}_{N+1}, \mathcal{D})\mathrm{d}^N\mathbf{a}_N. \tag{4}$$

where $\mathbf{a}_{N+1} = (\mathbf{a}, a(\mathbf{x}_{N+1}))$. We can write the joint posterior distribution as a product over the data points and a prior on the function $a(\mathbf{x})$

$$P(\mathbf{a}_{N+1}|\mathbf{x}_{N+1}, \mathcal{D}) = \frac{1}{Z}P(\mathbf{a}_{N+1}) \prod_{n=1}^{N} P(t_n|a(\mathbf{x}_n)) \tag{5}$$

Manuscript received September 24, 1998; revised April 24, 2000 and May 23, 2000.

The authors are with the Cavendish Laboratory, Cambridge CB3 0HE, U.K. (e-mail: mackay@mrao.cam.ac.uk.).

Publisher Item Identifier S 1045-9227(00)10019-0.

where Z is an appropriate normalizing constant. The prior on $\mathbf{a}_{N+1}$ shall be assumed to be a Gaussian process prior of the form

$$P(\mathbf{a}_{N+1}|\mathbf{x}_{N+1}, \{\mathbf{x}_n\}, \Theta) = \frac{1}{Z_{\rm gp}} \exp\left(-\frac{1}{2}\mathbf{a}_{N+1}^T \mathbf{C}_{N+1}^{-1} \mathbf{a}_{N+1}\right) \quad (6)$$

where $\mathbf{C}_{N+1}^{-1}$ is the $(N+1)\times(N+1)$ covariance matrix for $\mathbf{a}_{N+1}$ and $(\mathbf{C}_{N+1})_{mn} = \mathcal{C}_f(\mathbf{x}_m, \mathbf{x}_n)$. The covariance function $\mathcal{C}_f$ has the form

$$\mathcal{C}_f(\mathbf{x}_m, \mathbf{x}_n) = \theta_1 \exp\left\{-\frac{1}{2}\sum_{l=1}^{L}\frac{\left(x_m^{(l)} - x_n^{(l)}\right)^2}{r_l^2}\right\} + \theta_2 + \delta_{mn} J \quad (7)$$

where $\Theta = (\theta_1, \theta_2, J, \{r_l\})$ are appropriate hyperparameters and $x_m^{(l)}$ is the lth component of the vector $\mathbf{x}_m$. Unlike the regression covariance function [3] we assume $a(\mathbf{x})$ to be noise-free. [This does not mean that the *predictions* are deterministic: the probability of the target variable is defined in (1).] However, we do introduce a "jitter" term $\delta_{mn}J$ [14] to make the matrix computations well-conditioned. We choose the magnitude of J to be small in comparison to θ_1.

The product of sigmoid functions in (5) generally makes the integral in (4) analytically intractable. Barber and Williams [1] use a Laplace approximation in order to evaluate this integral. Neal ([14]) uses Markov chains Monte Carlo methods. Instead we introduce an upper and lower bound to the sigmoid function in order find analytic approximations to the posterior $P(\mathbf{a}_{N+1}|\mathbf{x}_{N+1}, \mathcal{D})$ and therefore find approximations to the posterior probability $P(t_{N+1} = 1|\mathbf{x}_{N+1}, \mathcal{D})$.

We can define upper and lower bounds on the sigmoid, i.e., on $P(t = 1|a(\mathbf{x}))$, as follows [6], [7]:

$$P(t_n = 1|a(\mathbf{x}_n)) \geq Q(t_n = 1|a_n, \nu_n) = g(\nu_n) \cdot \exp\left[(a_n - \nu_n)/2 - \lambda(\nu_n)(a_n^2 - \nu_n^2)\right] \quad (8)$$

$$P(t_n = 1|a(\mathbf{x}_n)) \leq R(t_n = 1|a_n, \mu_n) = \exp\left(\mu_n a_n - \mathcal{H}_2(\mu_n)\right) \quad (9)$$

where $a_n = a(\mathbf{x}_n)$ and

$$g(a_n) = \frac{1}{1+\exp(-a_n)} \quad (10)$$

$$\lambda(\nu_n) = \left[g(\nu_n) - 1/2\right]/2\nu_n. \quad (11)$$

μ_n and ν_n are variational parameters with μ_n in the interval $[0, 1]$ and $\mathcal{H}_2(x)$ is the binary entropy function

$$\mathcal{H}_2(x) = -x\log x - (1-x)\log(1-x). \quad (12)$$

Note that the bounds on $P(t_n = 0|a(\mathbf{x}_n))$ follow directly from the above as

$$P(t_n = 0|a(\mathbf{x}_n)) = 1 - P(t_n = 1|a(\mathbf{x}_n)) = 1 - \frac{1}{1+\exp(-a(\mathbf{x}_n))} = \frac{1}{1+\exp(a(\mathbf{x}_n))}. \quad (13)$$

We can use the two distributions Q and R to bound each factor $P(t_n|a(\mathbf{x}_n))$ in (5) by introducing two variational parameters μ_n and ν_n for each data point. Note we do not introduce any variational parameters for the point $\mathbf{x}_{N+1}$ (we shall introduce a seperate approximation when extracting the prediction later). We can then use these bounds on the sigmoid to approximate the posterior probability $P(t_{N+1} = 1|\mathbf{x}_{N+1}, \mathcal{D})$.

III. Making Predictions

A. Predictions Based on the Lower Bound

We can write down the following approximation for $P(\mathbf{a}_{N+1}|\mathbf{x}_{N+1}, \mathcal{D})$:

$$P(\mathbf{a}_{N+1}|\mathbf{x}_{N+1}, \mathcal{D}) \simeq \mathcal{P}_Q(\mathbf{a}_{N+1}|\mathbf{x}_{N+1}, \mathcal{D}, \{\nu_n\}, \Theta) = \frac{1}{Z'} P(\mathbf{a}_{N+1}|\mathbf{x}_{N+1}, \{\mathbf{x}_n\}, \Theta) \prod_{n=1}^{N} Q(t_n|a(\mathbf{x}_n), \nu_n) \quad (14)$$

where Z' is the appropriate normalizing constant. Now using the previous definition of $Q(t = 1|a, \nu)$ we can write

$$\prod_{n=1}^{N} Q(t_n|a(\mathbf{x}_n), \nu_n) \propto \exp\left[\mathbf{a}_N^T \Lambda_N \mathbf{a}_N - \mathbf{d}^T \mathbf{a}_N\right] \quad (15)$$

where Λ_N is a diagonal matrix with diagonal elements $(\lambda(\nu_1), \lambda(\nu_2), \ldots, \lambda(\nu_N))$, λ_n being defined in (11). The product over n is from one to N as there are no variational parameters associated with the input vector $\mathbf{x}_{N+1}$. The vector $\mathbf{d}$ reflects whether the training input vector $\mathbf{x}_n$ is a member of class 0 or class 1 and has components $d_n = (1/2)(-1)^{t_n+1}$. In the above equation, we have ignored any terms that are independent of $\mathbf{a}_N$ and $a(\mathbf{x}_{N+1})$. Such terms simply contribute to a normalizing constant which we shall not need to evaluate when making predictions.

Introducing the Gaussian process prior we can write

$$\mathcal{P}_Q(\mathbf{a}_{N+1}|\mathbf{x}_{N+1}, \mathcal{D}, \{\nu_n\}, \Theta) \propto \exp\left[-\tfrac{1}{2}\mathbf{a}_{N+1}^T(\mathbf{C}_{N+1}^{-1} + 2\Lambda_{N+1})\mathbf{a}_{N+1} + \mathbf{d}^T\mathbf{a}_N\right] \quad (16)$$

where $\Lambda_{N+1} = \mathrm{diag}(\lambda(\nu_1), \lambda(\nu_2), \ldots, \lambda(\nu_N), 0)$ We can see that the marginal distribution over $a(\mathbf{x}_{N+1})$ is a Gaussian. Using the block form of the inverse of a matrix, we can express the inverse of $\mathbf{C}_{N+1}$ in terms of $\mathbf{C}_N^{-1}$ and then find the mean a_l^{MP} and variance s_l^2 of $\mathcal{P}_Q(a(\mathbf{x}_{N+1})|\mathbf{x}_{N+1}, \mathcal{D}, \{\nu_n\}, \Theta)$

$$a_l^{MP} = \mathbf{k}_{N+1}^T \mathbf{H}_N^{-1} \mathbf{d} \quad (17)$$

$$s_l^2 = \kappa + 2\mathbf{k}_{N+1}^T \mathbf{H}_N^{-1} \Lambda_N \mathbf{k}_{N+1} \quad (18)$$

where

$$\mathbf{H}_N = \mathbf{I} + 2\Lambda_N \mathbf{C}_N \quad (19)$$

and

$$\mathbf{k}_{N+1} = (\mathcal{C}_f(\mathbf{x}_1, \mathbf{x}_{N+1}) \ldots, \mathcal{C}_f(\mathbf{x}_N, \mathbf{x}_{N+1})) \quad (20)$$

$$\kappa = \mathcal{C}_f(\mathbf{x}_{N+1}, \mathbf{x}_{N+1}). \quad (21)$$

Thus substituting the Gaussian lower bound approximation $\mathcal{P}_Q(a(\mathbf{x}_{N+1})|\mathbf{x}_{N+1}, \mathcal{D}, \{\nu_n\}, \Theta)$ into (3) we find

$$P(t_{N+1} = 1|\mathbf{x}_{N+1}, \mathcal{D}) \simeq g\left(\tau(s_l) a_l^{MP}\right) \quad (22)$$

where we have used the approximation for the integral of the product of a Gaussian and a sigmoid [10]

$$\int \mathrm{d}x\, g(x) \mathrm{Gaussian}\left(a_l^{MP}, s_l^2\right) \simeq g\left(\tau(s_l) a_l^{MP}\right) \quad (23)$$

where $\tau(s) = 1/\sqrt{1 + \pi s^2/8}$.

We should note that, although we have been using the lower bound on $P(t_n = 1|a(\mathbf{x}_n))$, we have not generated a lower bound on the probability $P(t_{N+1} = 1|\mathcal{D})$ but an approximation to it. Looking back at (14) we see that $\mathcal{P}_Q(\mathbf{a}_{N+1}|\mathbf{x}_{N+1}, \mathcal{D}, \{\nu_n\}, \Theta)$ is an approximation to $P(\mathbf{a}_{N+1}|\mathbf{x}_{N+1}, \mathcal{D})$ rather than a lower bound. The normalizing constant Z' (a lower bound to $P(\mathcal{D}|\Theta)$) is in the denominator of equation (14) and the factors $Q(t_n|a(\mathbf{x}_n), \nu_n)$ (lower bounds to $P(t_n|a(\mathbf{x}_n))$) are in the numerator. Hence the conflicting bounds introduced by these two terms mean any solution derived from $\mathcal{P}_Q(\mathbf{a}_{N+1}|\mathbf{x}_{N+1}, \mathcal{D}, \{\nu_n\})$ will be an approximation not a lower bound.

B. Predictions Based on the Upper Bound

Now let us consider the approximation to $P(\mathbf{a}_{N+1}|\mathbf{x}_{N+1}, \mathcal{D})$ found using the upper bound. We can write

$$\begin{aligned} &P(\mathbf{a}_{N+1}|\mathbf{x}_{N+1}, \mathcal{D}) \\ &\quad \simeq \mathcal{P}_R(\mathbf{a}_{N+1}|\mathbf{x}_{N+1}, \mathcal{D}, \{\mu_n\}, \Theta) \\ &\quad = \frac{1}{Z''} P(\mathbf{a}_{N+1}|\mathbf{x}_{N+1}, \{\mathbf{x}_n\}, \Theta) \prod_{n=1}^{N} R(t_n|a(\mathbf{x}_n), \mu_n). \end{aligned} \tag{24}$$

Using the previous definition of $R(t = 1|a, \mu)$ we can write

$$\prod_{n=1}^{N} R(t_n|a(\mathbf{x}_n), \mu_n) \propto \exp\left[\mathbf{b}^T \mathbf{a}_N\right] \tag{25}$$

where $b_i = \mu_i(-1)^{t_i+1}$, again reflecting which class the training data is in. As in the lower bound case, we have ignored terms that are independent of $\mathbf{a}_N$ and $a(\mathbf{x}_{N+1})$ as they simply contribute to a normalizing constant that we shall not need to evaluate when making predictions.

Introducing the Gaussian process prior we write

$$\begin{aligned} &\mathcal{P}_R(\mathbf{a}_{N+1}|\mathbf{x}_{N+1}, \mathcal{D}, \{\mu_n\}, \Theta) \\ &\quad \propto \exp\left[-\frac{1}{2}\mathbf{a}_{N+1}^T \mathbf{C}_{N+1}^{-1} \mathbf{a}_{N+1} + \mathbf{b}^T \mathbf{a}\right]. \end{aligned} \tag{26}$$

As with the lower bound, we can use the block form of the inverse to calculate the mean a_u^{MP} and the variance s_u^2 of the approximate marginal distribution $\mathcal{P}_R(a(\mathbf{x}_{N+1})|\mathcal{D}, \{\mu_n\}, \Theta)$

$$a_u^{MP} = -s_u^2 \mathbf{k}_{N+1}^T \mathbf{C}_N^{-1} \mathbf{b} \tag{27}$$

$$s_u^2 = \left(\kappa - \mathbf{k}_{N+1}^T \mathbf{C}_N^{-1} \mathbf{k}_{N+1}\right)^{-1}. \tag{28}$$

We use (23) again to calculate the approximation

$$P(t_{N+1} = 1|\mathbf{x}_{N+1}, \mathcal{D}) \simeq g(\tau(s_u) a_u^{MP}). \tag{29}$$

IV. Determining the Parameters

We have now derived two approximations to $P(t_{N+1} = 1|\mathbf{x}_{N+1}, \mathcal{D})$ using our upper and lower bounds for the sigmoid function. In order to make use of these approximations we need to find appropriate values for the variational parameters and the hyperparameters of the covariance function.

Consider the upper and lower bounds on $P(\mathcal{D}|\Theta)$

$$Z' \leq P(\mathcal{D}|\Theta) \leq Z'' \tag{30}$$

where

$$Z' = \int \mathrm{d}^N \mathbf{a}_N Q(t_n = 1|a(\mathbf{x}_n), \nu_n) \prod_{n=1}^{N} P(\mathbf{a}_N|\Theta) \tag{31}$$

$$Z'' = \int \mathrm{d}^N \mathbf{a}_N R(t_n = 1|a(\mathbf{x}_n), \mu_n) \prod_{n=1}^{N} P(\mathbf{a}_N|\Theta). \tag{32}$$

Note that Z' and Z'' are the normalizing constants from equations (14) and (24), respectively.

We wish to set the variational parameters $\{\nu_n\}$ and $\{\mu_n\}$ so that Z' and Z'' are as tight bounds on $P(\mathcal{D}|\Theta)$ as possible. We can do this by maximizing Z' with respect to $\{\nu_n\}$ and minimizing Z'' with respect to $\{\mu_n\}$. We would also like to set the hyperparameters of the covariance function to their most probable values given the data (a Monte Carlo approach could also be used [1]). This is not possible as we do not have an analytic expression for $P(\mathcal{D}|\Theta)$. However we can maximize Z' and Z'' with respect to Θ to obtain approximations to the most probable Θ given the data.

We now calculate the derivatives of the lower and upper bound on $P(\mathcal{D}|\Theta)$ with respect to the variational parameters and the hyperparameters of the covariance function. Given these derivatives we can then use a gradient-based optimization algorithm such as conjugate gradients to optimize the bounds.

A. The Lower Bound

We can write Z' as follows:

$$\begin{aligned} Z' = \prod_n g(\nu_n) \exp\left[-\left(\nu_n/2 - \lambda(\nu_n)\nu_n^2\right)\right] \int \mathrm{d}^N \mathbf{a}_N \frac{1}{Z_{\mathrm{gp}}} \\ \cdot \exp\left[-\frac{1}{2}\mathbf{a}_N^T(\mathbf{C}_N^{-1} + 2\Lambda_N)\mathbf{a}_N + \mathbf{d}^T \mathbf{a}_N\right]. \end{aligned} \tag{33}$$

The integral in Z' is tractable as the bound on the sigmoid function is a Gaussian function. Hence

$$\begin{aligned} \log(Z') = \sum_n \left(\log(g(\nu_n)) - \nu_n/2 + \lambda(\nu_n)\nu_n^2\right) \\ + \tfrac{1}{2}\mathbf{d}^T\left(\mathbf{C}_N^{-1} + 2\Lambda_N\right)\mathbf{d} - \tfrac{1}{2}\log\det(\mathbf{I} + 2\Lambda_N \mathbf{C}_N). \end{aligned} \tag{34}$$

We can show that for symmetric $\mathbf{C}_N$

$$\begin{aligned} \frac{\partial \log Z'}{\partial \nu_k} = -\frac{1}{4}\left[1 + 2g_k\left((1 - g_k)\nu_k - 1\right)\right] - \mathbf{d}^T \mathbf{C}_N \mathbf{H}_N^{-1} \frac{\partial \Lambda_N}{\partial \nu_k} \\ \cdot \mathbf{C}_N \mathbf{H}_N^{-1} \mathbf{d} - \frac{1}{2}\mathrm{tr}\left(\mathbf{H}_N^{-1} \frac{\partial \mathbf{H}_N}{\partial \nu_k}\right) \end{aligned} \tag{35}$$

$$\frac{\partial \log Z'}{\partial \theta} = \frac{1}{2}\left(\mathbf{H}_N^{-1}\mathbf{d}\right)^T \frac{\partial \mathbf{C}_N}{\partial \theta} \mathbf{H}_N^{-1}\mathbf{d} - \frac{1}{2}\,\mathrm{tr}\left(\mathbf{H}_N^{-1} \frac{\partial \mathbf{H}_N}{\partial \theta}\right) \tag{36}$$

where $g_k = g(\nu_k)$ and $\theta \in \Theta$ is some generic hyperparameter of the covariance function. We should note that in order to calculate

either of these derivatives we need only perform one inversion, i.e., find $\mathbf{H}_N^{-1}$ for any given $\{\nu_k\}$ and Θ.

B. The Upper Bound

Let us now consider the upper bound Z''

$$Z'' = \exp\left[-\sum_n \mathcal{H}_2(\mu_n)\right] \int \mathrm{d}^N \mathbf{a}_N \frac{1}{Z_{\mathrm{gp}}} \cdot \exp\left(-\tfrac{1}{2}\mathbf{a}_N^T \mathbf{C}_N^{-1} \mathbf{a}_N + \mathbf{b}^T \mathbf{a}_N\right). \quad (37)$$

Again the integral in Z'' is tractable

$$\log Z'' = -\sum_n \mathcal{H}_2(\mu_n) + \frac{1}{2}\mathbf{b}^T \mathbf{C}_N \mathbf{b} \quad (38)$$

We can calculate the derivatives of Z''

$$\frac{\partial \log Z''}{\partial \mu_k} = \log\left(\frac{\mu_k}{1-\mu_k}\right) + (-1)^{t_k+1} \sum_n (\mathbf{C}_N)_{kn} b_n \quad (39)$$

$$\frac{\partial \log Z''}{\partial \theta} = \frac{1}{2}\mathbf{b}^T \frac{\partial \mathbf{C}_N}{\partial \theta}\mathbf{b}. \quad (40)$$

The evaluation of these derivatives is trivial as no inversion is required.

C. Optimization Procedure

We have calculated the derivatives of the upper and lower bounds with respect to their variational parameters and with respect to the hyperparameters of the covariance function. However, we have not yet addressed the question of how we should go about the optimization. The lower bound case is simple as we can simultaneously maximize Z' with respect to the variational parameters $\{\nu_n\}$ and the hyperparameters of the covariance function Θ using a gradient-based optimization algorithm. The upper bound presents us with a slightly more difficult problem. Minimization of Z'' with respect to the variational parameters $\{\mu_n\}$ is straightforward. However maximizing Z'' with respect the hyperparameters of the Gaussian process is problematic. For example Z'' increases to infinity as θ_1 increases and hence no finite maximum exists. An alternative approach to the optimization of Z'' is to fix the hyperparameters Θ at the values determined by the maximization of the lower bound Z', and minimize Z'' with respect to the variational parameters $\{\mu_n\}$ alone. The justification for this procedure comes from the likely relative quality of the upper and lower bounds. Fig. 1 shows a sigmoid function and two bounds on it of the form used in our approximations. We can see that the lower bound is good across a wide range of values whereas the upper bound is poor except for the linear region at $x = 0$. This difference arises because the lower bound is constructed to touch the sigmoid in two places whereas the upper bound only touches the sigmoid once.

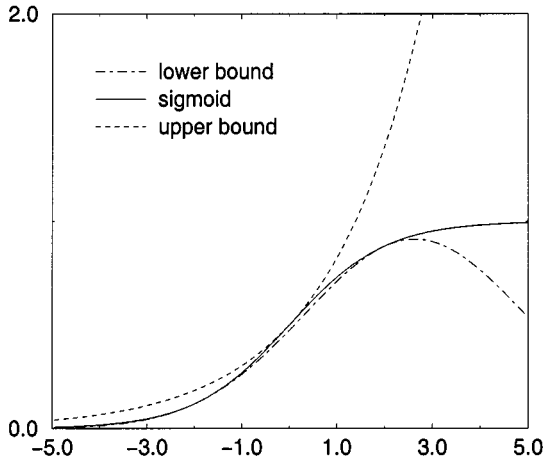


Fig. 1. **Bounds on sigmoid function:** This figure shows the sigmoid function and upper and lower bounds as defined in equations (8) and (9) with $\nu = -2.015$ and $\mu = 0.505$.

Thus we use an optimization procedure in which Z' is maximized with respect to the variational parameters $\{\nu_n\}$ and the hyperparameters Θ and then Z'' is minimized with respect to the variational parameters $\{\mu_n\}$ using the hyperparameters Θ generated by the maximization of Z'.

V. Examples

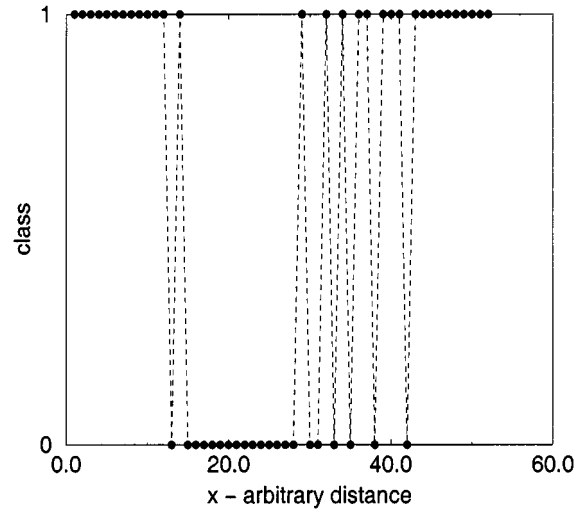
A. 1–D Toy Example

In order to illustrate various features of the variational Gaussian process classifier (VGC), we generated a one-dimensional (1-D) toy problem [see Fig. 2(a)]. We ran the optimization procedure described in Section IV-C using conjugate gradients from 20 different sets of initial conditions to guard against the presence of multiple optima in hyperparameter space. It is the authors' experience that, given sensible priors on the hyperparameters (gamma distributions were used in this case to limit the length scales $\{r_l\}$ to moderate values), such multiple minima rarely occur and did not occur in this case. The average of the predictions made by each of the 20 runs (which were almost identical) is shown in Fig. 2(b).

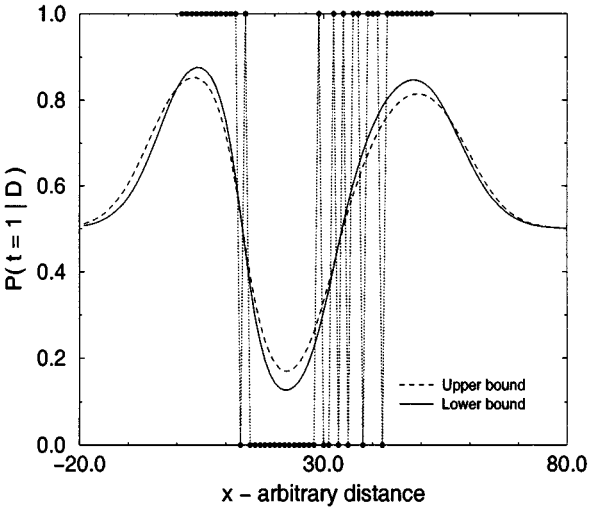
The results are much as expected. The VGC identifies the regions which belong to class 1 and class 0 and models the transitions between these regions smoothly with no over-fitting. The VGC also behaves well where there is no training data. On either side of the training data, the classifier's predictions for $P(t_{N+1} = 1|\mathcal{D})$ tend to 0.5. The lower bound gives more confident predictions than the upper bound; this is to be expected as the lower bound is generally tighter than the upper bound.

B. The CRABS and PIMA Examples

We next tried our method on two well known classification problems, the *Leptograpsus* crabs and Pima Indian diabetes



(a)



(b)

Fig. 2. **1-D Toy Example:** (a) binary training data. The data were chosen so that there would be a sharp transition from class 1 to class 0 ($x \simeq 13$) and a more gentle transition from class 0 to class 1 ($29 \leq x \leq 43$). (b) shows the upper bound and lower bound approximations of $P(t_{N+1} = 1|\mathcal{D})$. There are some interesting features to note. First, the lower bound makes more confident predictions than the upper bound. This is expected as the lower bound is generally tighter than the upper bound. Second, the classifier makes sensible predictions in regions where it has little data, i.e., $P(t_{N+1} = 1|\mathcal{D})$ tends to 0.5. Third, $P(t_{N+1} = 1|\mathcal{D})$ does not swing from zero to one on each data point on the boundary between classes but instead gives a smooth transition, i.e., there is no over-fitting. Finally, even in the regions where there are a significant number of zeros and ones, the classifier does not over-fit the data and make overconfident predictions.

datasets.[1] The results for both tasks, together with comparisons with several other methods from [1], [15], and [16] are given in Table I. The datasets are not sufficiently large for statistically significant differences between the methods to be measured.

In the **Leptograpsus** crabs problem we attempted to classify the sex of crabs based upon six characteristics. The 200 labeled examples are split into a training set of 80 and a test set of 120. The performance of the VGC is not significantly different from the best of the other methods. The Pima

[1]Available from http://markov.stats.ox.ac.uk/pub/.

TABLE I
PIMA AND CRABS RESULTS: THE TABLE SHOWS THE PERFORMANCE OF A RANGE OF DIFFERENT CLASSIFICATION MODELS ON THE PIMA AND CRABS PROBLEMS ([15] AND [16]). THE NUMBER OF CLASSIFICATION ERRORS AND THE PERCENTAGE OF ERRORS BOTH REFER TO THE TEST SET. THE ERROR BARS GIVEN ARE CALCULATED USING BINOMIAL STATISTICS. THE RESULTS QUOTED FOR THE VGC ARE THOSE OBTAINED USING THE APPROXIMATIONS FROM THE LOWER BOUND. THE HMC GAUSSIAN PROCESS IS THE CLASSIFIER USING THE LAPLACE APPROXIMATION, DESCRIBED IN [1]

	Crab		Pima	
Method	Error	% Error	Error	% Error
Neural Network (1)	3 ± 1.7	2.5 ± 1.4	-	-
Neural Network (2)	5 ± 2.1	4.2 ± 1.8	-	-
Neural Network (3)	-	-	75	22.6
Linear Discriminant	8 ± 2.7	6.7 ± 2.3	67 ± 7.3	20.2 ± 2.2
MARS (degree = 1)	8 ± 2.7	6.7 ± 2.3	75 ± 7.6	22.6 ± 2.3
2 Gaussian Mixture	-	-	64 ± 7.2	19.3 ± 2.2
HMC Gaussian process	3 ± 1.7	2.5 ± 1.4	68 ± 7.4	20.5 ± 2.2
VGC	4 ± 2	3.3 ± 1.6	70 ± 7.4	21.1 ± 2.2

TABLE II
WELD STRENGTH CLASSIFICATION PROBLEM: THIS TABLE SHOWS THE TEST ERROR AND LOG LIKELIHOOD SCORES OF THE VGC AND THE BAYESIAN NEURAL NETWORK OF [5]. THE TWO RESULTS GIVEN FOR THE VGC CORRESPOND TO THE APPROXIMATIONS USING THE LOWER AND UPPER BOUND, RESPECTIVELY

Method	Test Error	Log Likelihood
Bayesian Neural Network	8	-23.6
Variational GP Classifier	10/10	$-25.73/-31.57$

Indian diabetes problem involved the prediction of the occurence of diabetes in women of Pima Indian heritage based on seven characteristics. The 532 examples were split into 200 training examples and 332 test examples. Of the population, 33% were reported to have diabetes so an error rate of 33% can be achieved by declaring all examples to be nondiabetic. The VGC achieved an error rate of 21%—again comparable with the best of the other methods.

C. Weld Strength Example

Hot cracking can occur in welds as they cool. The occurence of such cracks depends on the chemical composition of the weld metal, the cooling rate and the weld geometry. We wish to predict whether a given weld will crack by examining the dependence of cracking on 13 specific characteristics of a weld. In a previous treatment of this problem using Bayesian neural networks [5] the relationship between cracking and carbon content was highlighted and compared with experimental data. We performed a similar analysis using VGCs.

An initial test was performed using a training set of 77 examples and a test set of 77 examples. The test error rates and test log likelihoods for the VGC and the Bayesian neural-network

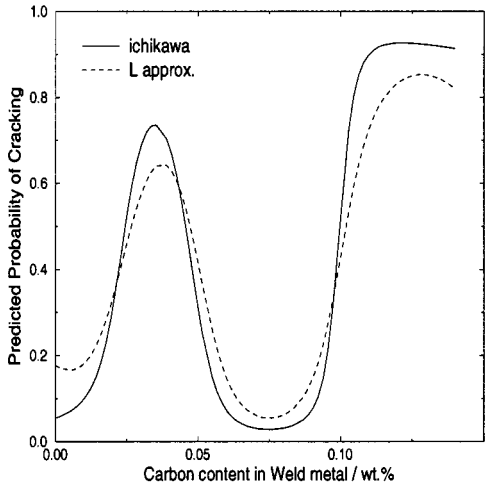


Fig. 3. **Carbon Dependence of Weld Strength:** The two plots shown on this graph are the results obtained in [5] and those found using the lower bound approximation of a VGC. Both have the same large scale features but the VGC makes less confident predictions where the training data density tails off near zero carbon content.

TABLE III
DOMINANT COMPUTATIONAL COSTS (PER GRADIENT EVALUATION) OF THE VARIATIONAL GAUSSIAN PROCESS METHOD, ASSUMING K INPUT DIMENSIONS AND N TRAINING EXAMPLES. THE TOTAL COST OF THE OPTIMIZATIONS IS THE GRADIENT-EVALUATION COST TIMES, THE NUMBER OF GRADIENT EVALUATIONS PER CONJUGATE GRADIENT LINE-SEARCH (ABOUT THREE), TIMES THE NUMBER OF LINE-SEARCHES (ABOUT THIRTY, THOUGH THIS VARIES FROM PROBLEM TO PROBLEM). THE ADDITIONAL COST OF PREDICTIONS USING EITHER BOUND IS N BASIS-FUNCTION EVALUATIONS AND N MULTIPLY-ADDS. THE VGC METHOD HAS SIMILAR COST TO THE LAPLACE APPROXIMATION METHOD OF [1], WHICH ALSO REQUIRES INVERSION OF $N \times N$ MATRICES

Lower bound optimization (number of adjustable parameters $\simeq K+N$):	
Calculation of $\mathbf{C}_N$, $\frac{\partial \mathbf{C}_N}{\partial \theta}$, and $\frac{\partial \mathbf{H}_N}{\partial \theta}$	N^2K
Calculation of $\mathbf{H}_N^{-1}$	N^3
Quadratic forms and traces	N^2
Conjugate gradient house-keeping	$K+N$

Upper bound optimization (number of adjustable parameters $\simeq N$):	
Calculation of $\mathbf{C}_N$	N^2K

approach [5] can be seen in Table II where the test log likelihood is defined as

$$\text{test log likelihood} = \sum_{n=1}^{N_{\text{test}}} t_n \log(\hat{t}_n) + (1-t_n)\log(1-\hat{t}_n) \tag{41}$$

where t_n is the true test set classification (either 0 or 1) and $\hat{t}_n$ is the prediction $P(t_n = 1|\mathcal{D})$. The performance of the VGC is slightly inferior to that of the Bayesian neural network. However the neural-network result was obtained using a committee of four networks, found after considerable experimentation, with the selected networks being *the ones with the best test error*. The VGC results required no such experimentation. Twenty runs of the VGC with differing initial conditions were performed to guard against multiple optima. The results quoted in Table II are from the first run; all of the runs produced almost identical results.

We then trained the VGC using all 154 examples in order to model the carbon dependence of the weld strength as in [5]. A plot of the carbon dependence can be seen in Fig. 3. The plot was much as expected. It shows the reduction in the probability of cracking at intermediate carbon concentrations (as found experimentally) and also shows a tendency for increased strength at low carbon concentrations. The corresponding results of Ichikawa *et al.* are also shown in Fig. 3.

VI. DISCUSSION

We have shown that Gaussian processes can be used to produce effective binary classifiers. The results using the VGC are comparable to the best of current classification models. Using Gaussian processes we obtain a parameterization of our model that is easily interpretable allowing us to perform automatic relevance determination where applicable. VGCs are moderately simple to implement and use. The very small number of parameters of the model that need to be determined by hand (generally only the priors on the hyperparameters) makes VGCs an especially useful tool for automated tasks where fine tuning for each problem is not possible. However, we do not appear to sacrifice any performance for this simplicity.

One obvious problem with any method based on Gaussian processes is the computational cost associated with calculating and inverting an $N \times N$ matrix (see Table III). The cost of direct methods of inversion may become prohibitive when the number of data points N is greater than $\simeq$1000. In [3] efficient methods for matrix inversion [17] are developed that when applied to the Gaussian process framework allow large data sets to be tackled. Another problem with the variational approach is the profileration of variational parameters when dealing with large amounts of the data. Reducing the number of these variational parameters is an important direction for further research. The extension of the method to multiple classes has been investigated in [2].

ACKNOWLEDGMENT

The authors would like to thank B. Ripley, K. Ichikawa, D. Barber, and C. Williams for generously making their data available. The authors would also like to thank C. Williams, S. Omohundro, and W. Buntine for helpful discussions.

REFERENCES

[1] D. Barber and C. K. I. Williams, "Gaussian processes for Bayesian classification via hybrid Monte Carlo," in *Neural Information Processing Systems 9*, M. C. Mozer, M. I. Jordan, and T. Petsche, Eds. Cambridge, MA: MIT Press, 1997, pp. 340–346.

[2] M. N. Gibbs, "Bayesian Gaussian processes for regression and classification," Ph.D. dissertation, Cambridge Univ., 1997.

[3] M. N. Gibbs and D. J. C. MacKay. (1996) Efficient implementation of Gaussian processes for interpolation. Tech. Rep.. [Online] Available: http://wol.ra.phy.cam.ac.uk/mackay/abstracts/gpros.html

[4] G. E. Hinton and D. van Camp, "Keeping neural networks simple by minimizing the description length of the weights," in *Proc. 6th Annu. Workshop Comput. Learning Theory*. New York, 1993, pp. 5–13.

[5] K. Ichikawa, H. K. D. H. Bhadeshia, and D. J. C. MacKay, "Model for hot cracking in low-alloy steel weld metals," *Sci. Technol. Welding Joining*, vol. 1, pp. 43–50, 1996.

[6] T. S. Jaakkola and M. I. Jordan, "Computing upper and lower bounds on likelihoods in intractable networks," in *Proc. 12th Conf. Uncertainty AI*: Morgan Kaufman, 1996.

[7] ——, "Bayesian parameter estimation via variational methods," *Statist. Comput.*, vol. 10, no. 1, pp. 25–37, 2000.
[8] S. L. Lauritzen and D. J. Spiegelhalter, "Local computations with probabilities on graphical structures and their application to expert systems," *J. Roy. Statist. Soc. B*, pp. 157–224, 1988.
[9] D. J. C. MacKay, "The evidence framework applied to classification networks," *Neural Comput.*, vol. 4, no. 5, pp. 698–714, 1992a.
[10] ——, "A practical Bayesian framework for backpropagation networks," *Neural Comput.*, vol. 4, no. 3, pp. 448–472, 1992b.
[11] ——, "Free energy minimization algorithm for decoding and cryptanalysis," *Electron. Lett.*, vol. 31, no. 6, pp. 446–447, 1995a.
[12] ——, "Probable networks and plausible predictions — A review of practical Bayesian methods for supervised neural networks," *Network: Comput. Neural Syst.*, vol. 6, pp. 469–505, 1995b.
[13] R. M. Neal, "Bayesian learning for neural networks," Ph.D. dissertation, Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, 1995.
[14] ——, "Monte Carlo implementation of Gaussian process models for Bayesian regression and classification," Dept. Comput. Sci., Univ. Toronto, Tech. Rep. CRG--TR--97--2, 1997.
[15] B. D. Ripley, "Flexible nonlinear approaches to classification," in *From Statistics to Neural Networks. Theory and Pattern Recognition Applications*, V. Cherkassky, J. H. Friedman, and H. Wechsler, Eds. New York: Springer-Verlag, 1994.
[16] B. D. Ripley, *Pattern Recognition and Neural Networks*. Cambridge, U.K.: Cambridge Univ. Press, 1996.
[17] J. Skilling, "Bayesian numerical analysis," in *Physics and Probability*, W. T. Grandy Jr. and P. Milonni, Eds. Cambridge, U.K.: Cambridge Univ. Press., 1993.
[18] C. K. I. Williams, "Regression with Gaussian processes," *Ann. Math. Artificial Intell.*, 1995.
[19] ——, "Computation with infinite neural networks," *Neural Comput.*, vol. 10, no. 5, pp. 1203–1216, 1998.
[20] C. K. Williams and C. E. Rasmussen, "Gaussian processes for regression," in *Advances in Neural Information Processing Systems 8*, D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, Eds. Cambridge, MA: MIT Press, 1996.