Bayesian Gaussian Process Classification with the EM-EP Algorithm

Hyun-Chul Kim and Zoubin Ghahramani, *Member*, *IEEE*

Abstract—Gaussian process classifiers (GPCs) are Bayesian probabilistic kernel classifiers. In GPCs, the probability of belonging to a certain class at an input location is monotonically related to the value of some latent function at that location. Starting from a Gaussian process prior over this latent function, data are used to infer both the posterior over the latent function and the values of hyperparameters to determine various aspects of the function. Recently, the expectation propagation (EP) approach has been proposed to infer the posterior over the latent function. Based on this work, we present an approximate EM algorithm, the EM-EP algorithm, to learn both the latent function and the hyperparameters. This algorithm is found to converge in practice and provides an efficient Bayesian framework for learning hyperparameters of the kernel. A multiclass extension of the EM-EP algorithm for GPCs is also derived. In the experimental results, the EM-EP algorithms are as good or better than other methods for GPCs or Support Vector Machines (SVMs) with cross-validation.

Index Terms—Gaussian process classification, Bayesian methods, kernel methods, expectation propagation, EM-EP algorithm.

1 Introduction

Kernel classifiers have recently received much attention from the machine learning community. Some popular kernel classifiers are the support vector machine (SVM), Bayes point machine (BPM), and Gaussian process classifier (GPC). The SVM was proposed as a classifier maximizing the margin, which is the smallest distance between data points and the class boundary [1]. SVMs have been a popular tool and have resulted in many successful applications. The BPM is a kernel classifier whose goal is to approximate Bayes-optimal classification by finding the center of the mass of version space, which is the set of hyperplanes in feature space that separate the data [2]. It was also shown that SVMs can be viewed as a form of Bayes point machine which tries to find the center of the largest ball to fit in version space. In contrast with the above two classifiers, GPCs are Bayesian kernel classifiers derived from Gaussian process priors over functions which were developed originally for regression [3], [4], [5].

Gaussian processes for regression [5], [6], [7], [8] assume that the target function has a Gaussian process prior. This means that the density of any collection of target function values is modeled as a multivariate Gaussian density. Usually, the mean of this Gaussian is assumed to be zero and the covariance between the targets at two different points is a decreasing function of their distance in input space. This decreasing function is controlled by a small set of *hyperparameters* that capture interpretable properties of the function, such as the length scale of autocorrelation, the overall scale of the function, and the amount of noise. The posterior distributions of these hyperparameters given the data can be inferred in a Bayesian way via Markov Chain Monte Carlo (MCMC) methods [5], [7] or they can be selected by maximizing the marginal likelihood (also known as the evidence) [8]. A Bayesian treatment of multilayer perceptrons for regression has been shown to converge to a Gaussian process as the number of hidden nodes approaches to infinity, if the prior on input-to-hidden weights and hidden unit biases are independent and identically distributed [9]. Empirically, Gaussian processes have been shown to be an excellent method for nonlinear regression [10].

In GPCs, the target values are discrete class labels while the target values in GP regression are continuous real values. It is not appropriate to assume that the target function with discrete outputs has a Gaussian process prior. We assume that there is some latent function whose value at a certain input location is monotonically related to the probability of belonging to a certain class at that location and that the latent function rather than the target function has a Gaussian process prior. We can use a Gaussian process as a prior of the latent function, and for multiclass classification, one can use multiple GPs or a multivariate GP. Since only the class labels are observed in GPCs, we need to integrate not only over hyperparameters but also over latent values of these functions at the data points. Williams and Barber [3] used a Laplace approximation to integrate over the latent values and Hybrid Monte Carlo (HMC) to integrate over the hyperparameters. Neal [5] used Gibbs sampling to integrate over latent values and used HMC to integrate over hyperparameters. Gibbs and Mackay [4] used a variational approximation method to integrate over latent values and determined hyperparameters by maximizing the marginal likelihood. Opper and Winther

- *H.-C. Kim is with the Department of Industrial and Management Engineering, Pohang University of Science and Technology, San 31 Hyoja-Dong, Nam-gu, Pohang, 790-784, Republic of Korea. E-mail: grass@postech.ac.kr.*
- *Z. Ghahramani is with the Department of Engineering, University of Cambridge, Trumpington Street, Cambridge CB2 1PZ, UK. E-mail: zoubin@eng.cam.ac.uk.*

Manuscript received 2 Nov. 2005; revised 3 Apr. 2006; accepted 12 Apr. 2006; published online 12 Oct. 2006.
Recommended for acceptance by M. Figueiredo.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0595-1105.

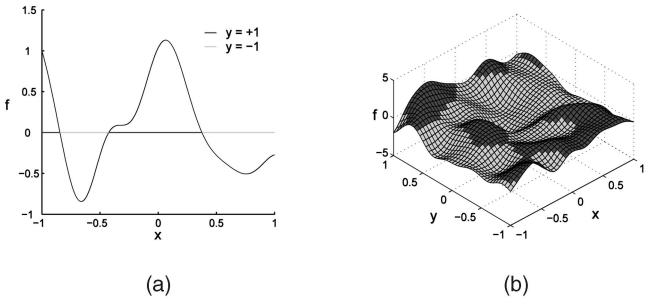
[11] used the TAP approach originally proposed in statistical physics of disordered systems to integrate over latent values.

It turns out that the TAP approach for GPCs is equivalent to the Expectation Propagation (EP) algorithm for approximate inference in Bayes point kernel machines [12]. EP has been shown to give results which are superior to Laplace's method and are very similar to MCMC methods both in terms of predictive distributions and marginal likelihood estimates [13]. In these previous papers, the focus has been on approximate inference rather than determining hyperparameters. Two potential methods for determining the hyperparameters have been proposed in [14]. The first method, which they called mean field method I, is to maximize the variational lower bound of the evidence under the assumption that the densities of latent values are independent and Gaussian. The second method, which they called mean field method II, is to maximize the evidence approximated by Fourier transformation of the likelihood and a saddle point approximation. An EM algorithm for learning the kernel length scales using an L1 prior has also been proposed [15].

In this paper, we propose and investigate a conceptually simple EM-like algorithm to learn the hyperparameters which we call EM-EP.[1] In the E-step, we use EP to estimate the joint density of latent values under the assumption that the joint density is multivariate Gaussian. This multivariate approximation is better than factorized approximations such as the mean field method I. In the M-step, we maximize with respect to the hyperparameters the variational lower bound on the marginal likelihood given by using the density of latent values obtained from the E-step. These two steps are repeated until convergence. The idea of using the variational lower bound for model selection in GPC was suggested in [17]. Here, we use a slightly different formulation for GPC and provide experimental results. Another emphasis of this paper is examining the role of the different hyperparameters and comparing these algorithms with several variants of SVMs. We also propose an extension of the EM-EP algorithm for multiclass classification. Finally, although improving computational complexity of GPC learning through sparsification methods is an important research topic ([18], [19], [20]), we will not address this problem in this paper.

The paper is organized as follows: Section 2 introduces Gaussian process classification. In Section 3, we introduce the EP method for GPCs, derive the EM-EP algorithm, and show the experimental results. In Section 4, we derive the EP method and the EM-EP algorithm for multiclass GPCs and show experimental results. In Section 5, we discuss our approach and related work. Software implementing the EM-EP algorithm is available at http://home.postech.ac.kr/~grass/software/.

1. An earlier version of this paper focusing on the EM-EP algorithm for binary classification was presented at a workshop [16].



(a) (b)

Fig. 1. In Gaussian processes for classification the class label is related to a latent function. (a) Examples of one-dimensional and (b) two-dimensional data showing a latent function sampled from a Gaussian process prior and the corresponding class label under the threshold model.

2 Gaussian Process Classification

Let us assume that we have a data set $\mathcal{D}$ of data points $\mathbf{x}_i$ with binary class labels $y_i \in \{-1, +1\}$:

$$\mathcal{D} = \{(\mathbf{x}_i, y_i)|i = 1, 2, \ldots, n\},$$
$$X = \{\mathbf{x}_i|i = 1, 2, \ldots, n\},$$
$$\mathbf{y} = \{y_i|i = 1, 2, \ldots, n\}.$$

Given this data set, the classification problem is to output the correct class label for a new data point. To represent our uncertainty over class labels, one may want a method that outputs probabilities over the different labels for each new data point.

We assume that the probability over class labels as a function of $\mathbf{x}$ depends on the value of some latent real-valued function $f(\mathbf{x})$. That is, for binary classification, given the value of $f(\mathbf{x})$ the probability of class label is independent of all other quantities: $p(y = +1|\mathbf{x}, f(\mathbf{x}), \mathcal{D}) = p(y = +1|f(\mathbf{x}))$. The probability of observing $y = +1$ is assumed to be a monotonically increasing function of $f(\mathbf{x})$. This can take several forms, for example for $f_i = f(\mathbf{x}_i)$:

$$p(y_i|f_i) = \begin{cases} \frac{1}{1+\exp(-y_i f_i)} & \text{sigmoid} \\ \Phi(y_i f_i) & \text{cumulative normal} \\ H(y_i f_i) & \text{threshold} \\ \epsilon + (1 - 2\epsilon)H(y_i f_i) & \text{noisy threshold,} \end{cases} \quad (1)$$

where $H(z) = 1$ iff $z > 0$, $\Phi(z)$ is the cumulative normal function $\Phi(z) = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} \exp(-\frac{x^2}{2}) dx$, and $0 \leq \epsilon < 0.5$.

We put a Gaussian process (see Fig. 1) prior on this function, meaning that any number of points evaluated from the function have a multivariate Gaussian density (see [7] for a review of GPs). Assume that this GP prior is parameterized by Θ which we will call the hyperparameters. We can write the probability of interest given Θ as:

$$p(\tilde{y}|\tilde{\mathbf{x}}, \mathcal{D}, \Theta) = \int p(\tilde{y}|\tilde{f}, \Theta) p(\tilde{f}|\mathcal{D}, \tilde{\mathbf{x}}, \Theta)\, d\tilde{f}. \quad (2)$$

The second part of (2) is obtained by further integration over $\mathbf{f} = [f_1, f_2 \cdots f_n]$, the values of the latent function at the data points.

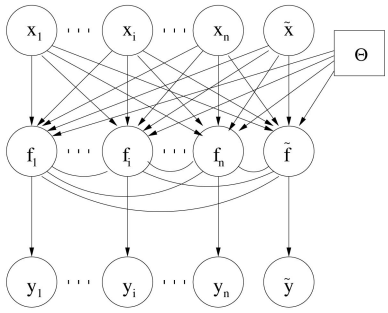


Fig. 2. Graphical model for GPCs with n training data points and one test data point. $\mathbf{x}_i$ and y_i are observed, $\tilde{\mathbf{x}}$ is given, $\tilde{y}$ is what should be predicted, f_i and $\tilde{f}$ are latent and jointly Gaussian, hence have the undirected edges.

$$\begin{aligned} p(\tilde{f}|\mathcal{D},\tilde{\mathbf{x}},\Theta) &= \int p(\mathbf{f},\tilde{f}|\mathcal{D},\tilde{\mathbf{x}},\Theta)\, d\mathbf{f} \\ &= \int p(\tilde{f}|\tilde{\mathbf{x}},\mathbf{f},X,\Theta)p(\mathbf{f}|\mathcal{D},\Theta)d\mathbf{f}, \end{aligned} \tag{3}$$

where

$$p(\mathbf{f}|\mathcal{D},\Theta) \propto p(\mathbf{y}|\mathbf{f},X,\Theta)p(\mathbf{f}|X,\Theta) = \left(\prod_{i=1}^{n} p(y_i|f_i,\Theta)\right) p(\mathbf{f}|X,\Theta). \tag{4}$$

The first term is the probability of each observed class label given the latent function value, which can be of one of the forms in (1), while the second term is the GP prior over functions evaluated at the data. Writing the dependence of $\mathbf{f}$ on $\mathbf{x}$ implicitly, the GP prior over functions can be written

$$p(\mathbf{f}|X,\Theta) = \frac{1}{(2\pi)^{N/2}|\mathbf{C}_\Theta|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{f}-\boldsymbol{\mu})^\top \mathbf{C}_\Theta^{-1}(\mathbf{f}-\boldsymbol{\mu})\right\}, \tag{5}$$

where the mean $\boldsymbol{\mu}$ is usually assumed to be zero $\boldsymbol{\mu} = \vec{0}$ and each term of a covariance matrix C_{ij} is a function of $\mathbf{x}_i$ and $\mathbf{x}_j$, i.e., $c(\mathbf{x}_i,\mathbf{x}_j)$. This *covariance function* is the *kernel* which defines how data points generalize to nearby data points. The covariance function is parameterized by the hyperparameters Θ, which we can learn from the data. We will describe the particular covariance functions used in this paper later on. The Gaussian process classifier can be represented using the graphical model shown in Fig. 2.

In general, the class probability at a test point would be obtained by integrating over the hyperparameters weighted by their posterior probability

$$p(\tilde{y}|\tilde{\mathbf{x}},\mathcal{D}) = \int p(\tilde{y}|\tilde{\mathbf{x}},\mathcal{D},\Theta)p(\Theta|\mathcal{D})\, d\Theta. \tag{6}$$

This integral is costly and there are usually many fewer hyperparameters than data points. Therefore, in this paper, rather than integrating over the hyperparameters, we fit them by maximizing the marginal likelihood as $\hat{\Theta} = \arg\max_\Theta p(\mathcal{D}|\Theta)$ and predict using these best fit hyperparameters: $p(\tilde{y}|\tilde{\mathbf{x}},\mathcal{D},\hat{\Theta})$. The marginal likelihood and $p(\mathbf{f}|\mathcal{D},\Theta)$ in (4) are both intractable due to the nonlinearities in (1). We use the Expectation-Propagation (EP) algorithm to approximate both.

3 The EM-EP Algorithm

3.1 Expectation Propagation

The Expectation Propagation (EP) algorithm is an approximate Bayesian inference method [12]. We briefly review EP in its general form before describing its application to GPCs.

Consider a Bayesian inference problem where the posterior over some latent value (or parameter) $\mathbf{f}$ is proportional to the prior times likelihood terms for an i.i.d. data set

$$p(\mathbf{f}|y_1,\ldots,y_n) \propto p(\mathbf{f})\prod_{i=1}^{n} p(y_i|\mathbf{f}) = \prod_{i=0}^{n} t_i(\mathbf{f}), \tag{7}$$

where $t_0(\mathbf{f}) = p(\mathbf{f})$ and $t_i(\mathbf{f}) = p(y_i|\mathbf{f})$ for $i = 1,\ldots,n$. Notice that, dropping some variables being conditioned on, (4) is of this form. We approximate this distribution with a product of simple terms

$$q(\mathbf{f}) \propto \tilde{t}_0(\mathbf{f})\prod_{i=1}^{n} \tilde{t}_i(\mathbf{f}), \tag{8}$$

where each term (and therefore q) is assumed to be in the exponential family. EP iterates the following procedure over i until convergence:

1. Remove the ith term from $q(\mathbf{f})$: $q^{\setminus i}(\mathbf{f}) = \prod_{j\neq i}^{n} \tilde{t}_j(\mathbf{f})$.
2. Multiply by the true ith factor: $q^{\setminus i}(\mathbf{f})t_i(\mathbf{f}) = q(\mathbf{f})\frac{t_i(\mathbf{f})}{\tilde{t}_i(\mathbf{f})}$.
3. Find a new $\tilde{t}_i(\mathbf{f}) = t(\mathbf{f})$ such that it minimizes the Kullback-Leibler divergence[2] from $q^{\setminus i}(\mathbf{f})t_i(\mathbf{f})$ to $q^{\setminus i}(\mathbf{f})t(\mathbf{f})$:

$$\begin{aligned} \tilde{t}_i^{\text{new}}(\mathbf{f}) &= \arg\min_{t(\mathbf{f})} \mathrm{KL}\left(q^{\setminus i}(\mathbf{f})t_i(\mathbf{f}) \middle\| q^{\setminus i}(\mathbf{f})t(\mathbf{f})\right) \\ &= \arg\min_{t(\mathbf{f})} \mathrm{KL}\left(\frac{q(\mathbf{f})}{\tilde{t}^{\text{old}}{}_i(\mathbf{f})}p(y_i|\mathbf{f}) \middle\| \frac{q(\mathbf{f})}{\tilde{t}^{\text{old}}{}_i(\mathbf{f})}t(\mathbf{f})\right). \end{aligned} \tag{9}$$

Since q is in the exponential family, this minimization is solved by matching moments of the approximated distribution.

The algorithm is not guaranteed to converge although it did in practice in all our examples. Assumed Density Filtering (ADF)[3] is a special online form of EP where only one pass through the data is performed ($i = 1,\ldots n$). EP can be seen as an extension of ADF to batch situations.

EP has been applied to several Bayesian learning problems and its excellent performance has been demonstrated on other problems. Minka showed that EP is better than Laplace's method and the variational Bayes method in terms of accuracy and computational cost for simple Bayesian learning problems such as the clutter problem and mixture weights learning problem [12]. It has been shown that EP provides better accuracy than variational methods at a comparable cost for the generative aspect model [21]. EP was also applied to the signal detection problem in flat-fading channels which can be formulated as

2. The Kullback-Leibler divergence between two distributions $\mathrm{p}(\mathbf{f})$ and $\mathrm{q}(\mathbf{f})$ is defined as: $KL(p(\mathbf{f})||q(\mathbf{f})) = \int p(\mathbf{f})\log\frac{p(\mathbf{f})}{q(\mathbf{f})}\, d\mathbf{f}$.

3. ADF appears in several fields such as control, statistics, and artificial intelligence as different names such as online Bayesian learning, moment matching, and weak marginalization.

an estimation problem in a hybrid dynamic system with both continuous and discrete variables [22]. In this problem, EP provides a much lower computational cost than Monte Carlo filter and smoothers. Tree-structured EP showed better accuracy and convergence than normal belief propagation, and a lower cost than variational trees or double-loop algorithms [23]. A Bayes point machine with EP which allows only hard decision boundaries showed better performance than a hard-margin support vector machine in most cases [12]. EP is not guaranteed to converge but in practice it converges in many cases. Its generalized version which is convergent but slower has been proposed [24].

3.2 EP for Gaussian Process Classification

We describe EP for GPC referring to [11], [12], [20]. The form of the likelihood we use in the GPC is

$$p(y_i|f_i) = \epsilon + (1-2\epsilon)H(y_i f_i), \tag{10}$$

where $H(x) = 1$ if $x > 0$, and otherwise 0. The hyperparameter, ϵ in (10) models labeling error outliers. The EP algorithm approximates the posterior

$$p(\mathbf{f}|D,\Theta) = \frac{p(\mathbf{f}|X,\Theta)p(\mathbf{y}|\mathbf{f},\Theta)}{p(\mathbf{y}|X,\Theta)} \tag{11}$$

as a Gaussian having the form

$$q(\mathbf{f}) = \mathcal{N}(\mathbf{h},\mathbf{A}), \tag{12}$$

where the GP prior $p(\mathbf{f}|X,\Theta) \sim \mathcal{N}(\mathbf{0},\mathbf{C})$ has the covariance matrix $\mathbf{C}$ with elements C_{ij} defined by the covariance function

$$C_{ij} = c(\mathbf{x}_i,\mathbf{x}_j) = \theta_0 \exp\left\{-\frac{1}{2}\sum_{m=1}^{d} l_m d_m(x_i^m, x_j^m)\right\} + \theta_1 + \theta_2\delta(i,j), \tag{13}$$

where x_i^m is the mth element of $\mathbf{x}_i$,

$$d_m(x_i^m, x_j^m) = \begin{cases} (x_i^m - x_j^m)^2 & \text{if } x^m \text{ is continous;} \\ 1-\delta(x_i^m, x_j^m) & \text{if } x^m \text{ is discrete,} \end{cases} \tag{14}$$

and $\delta(x_i^m, x_j^m)$ is a Kronecker delta function. $1-\delta(x_i^m, x_j^m)$ is a more reasonable distance measure than $(x_i^m - x_j^m)^2$ for discrete data.

The hyperparameter θ_0 specifies the overall vertical scale of variation of the latent values, θ_1 the overall bias of the latent values from zero mean, θ_2 the latent noise variance, and l_m the (inverse) lengthscale for feature dimension m. The cumulative normal density likelihood term in (1) is equivalent to using the threshold function in (10) with $\epsilon = 0$ and nonzero latent noise θ_2.

EP tries to approximate the posterior (11) which can be written as:

$$p(\mathbf{f}|D,\Theta) = \frac{p(\mathbf{f}|X,\Theta)\prod_{i=1}^{n} p(y_i|\mathbf{f})}{p(\mathbf{y}|X,\Theta)}. \tag{15}$$

$p(y_i|\mathbf{f}) = t_i(\mathbf{f})$ in (10) is approximated by

$$\tilde{t}_i(\mathbf{f}) = s_i \exp\left(-\frac{1}{2v_i}(f_i - m_i)^2\right). \tag{16}$$

From this initial setting, we can derive EP for GPC by applying the general idea described above. The details of the derivation are in [25]. The resulting EP procedure is virtually identical to the one derived for BPMs in [12]. We define the following notation:[4]

$$\begin{aligned} &\mathbf{\Lambda} = \text{diag}(v_1,\ldots,v_n); h_i = E[f_i]; h_i^{\backslash i} = E[f_i^{\backslash i}]; \\ &\lambda_i = Var[f_i]; \lambda_i^{\backslash i} = Var[f_i^{\backslash i}], \end{aligned} \tag{17}$$

where $h_i^{\backslash i}$ and $f_i^{\backslash i}$ are values obtained from a whole set except for $\mathbf{x}_i$. The EP algorithm is as follows: After the initialization

$$v_i = \infty, m_i = 0, s_i = 1, h_i = 0, \lambda_i^{\backslash i} = C_{ii}, \tag{18}$$

the following process is performed until all (m_i, v_i, s_i) converge: Loop $i = 1, 2, \ldots, n$:

1. Remove the approximate density $\tilde{t}_i$ (for ith data point) from the posterior $q(\mathbf{f})$ to get an "old" posterior $q^{\backslash i}(\mathbf{f})$, and get a marginal $q^{\backslash i}(f_i) = \mathcal{N}(h_i^{\backslash i}, \lambda_i^{\backslash i})$:

$$h_i^{\backslash i} = h_i + \lambda_i^{\backslash i} v_i^{-1}(h_i - m_i); \lambda_i^{\backslash i} = (1/A_{ii} - 1/v_i)^{-1}. \tag{19}$$

2. Find $q^{new}(f_i) \sim \mathcal{N}(h_i, \lambda_i)$ which minimizes KL divergence from $q^{\backslash i}(f_i)t_i(f_i)$ to $q^{new}(f_i)$:

$$\begin{aligned} &z = \frac{y_i h_i^{\backslash i}}{\sqrt{\lambda_i^{\backslash i}}}; Z_i = \epsilon + (1-2\epsilon)\Phi(z); \\ &\alpha_i = \frac{y_i}{\sqrt{\lambda_i^{\backslash i}}}\frac{(1-2\epsilon)\mathcal{N}(z;0,1)}{Z_i}; h_i = h_i^{\backslash i} + \lambda_i^{\backslash i}\alpha_i, \end{aligned} \tag{20}$$

   where $\Phi(z)$ is a cumulative normal density function defined in (1).

3. Get a new $\tilde{t}_i(f_i) = s_i \exp(-\frac{1}{2v_i}(f_i - m_i)^2)$:

$$\begin{aligned} &v_i = \lambda_i^{\backslash i}\left(\frac{1}{\alpha_i h_i} - 1\right); m_i = h_i + v_i\alpha_i; \\ &s_i = Z_i\sqrt{1 + v_i^{-1}\lambda_i^{\backslash i}}\exp\left(\frac{\lambda_i^{\backslash i}\alpha_i}{2h_i}\right). \end{aligned} \tag{21}$$

4. Obtain a new $q(\mathbf{f}) \sim \mathcal{N}(\mathbf{h},\mathbf{A})$ using a new $\tilde{t}_i(f_i)$:

$$\mathbf{A} = (\mathbf{C}^{-1} + \mathbf{\Lambda}^{-1})^{-1}; \mathbf{h} = \mathbf{A}\mathbf{\Lambda}^{-1}\mathbf{m}. \tag{22}$$

The approximate evidence $\mathcal{Z}(\Theta)(\approx p(\mathbf{y}|X,\Theta))$ is as follows:

$$\mathcal{Z}(\Theta) = \frac{t_0(\mathbf{f})\prod_{i=1}^{n}\tilde{t}_i(f_i)}{q(\mathbf{f})} = \frac{|\mathbf{\Lambda}|^{1/2}}{|\mathbf{C}+\mathbf{\Lambda}|^{1/2}}\exp(-r/2)\prod_{i=1}^{n} s_i, \tag{23}$$

where $r = \sum_i \frac{m_i^2}{v_i} - \sum_{ij} A_{ij}\frac{m_i m_j}{v_i v_j}$. One iteration of the above EP algorithm can be executed in $O(n^3)$ because (22) can be done in $O(n^2)$ by the Woodbury formula also known as the matrix inversion lemma [26]. Our approximated posterior

4. $\text{diag}(v_1,\ldots,v_n)$ means a diagonal matrix whose diagonal elements are $v_1,\ldots,v_n$. Similarly for $\text{diag}(\mathbf{v})$.

over latent values is $q(\mathbf{f}) = \mathcal{N}(\mathbf{h}, \mathbf{A})$. According to [11], it can also be written as $q(\mathbf{f}) = \mathcal{N}(\mathbf{C}\boldsymbol{\alpha}, \mathbf{A})$. The approximate evidence in (23) can be used to measure the quality of fit of kernels or their hyperparameters to the data for model selection. However, it is difficult to obtain an updating rule from (23). In the following section, we derive the algorithm to find the hyperparameters automatically, based not on (23) but a variational lower bound of the evidence.

We will demonstrate how to find the hyperparameters soon, but for the moment, we'd like to concentrate on how we predict the class probabilities at a new point, $\tilde{\mathbf{x}}$. To begin with, we obtain the density for a latent value $\tilde{f}$ corresponding to $\tilde{\mathbf{x}}$ from (3). We obtain, using the approximation in (12),

$$\begin{aligned} p(\tilde{f}|\mathcal{D}, \tilde{\mathbf{x}}, \Theta) &= \int p(\tilde{f}|\tilde{\mathbf{x}}, \mathbf{f}, \Theta) p(\mathbf{f}|\mathcal{D}, \Theta)\, d\mathbf{f} \\ &\approx \mathcal{N}(\mathbf{k}^T(\mathbf{C})^{-1}\mathbf{h}, \kappa - \mathbf{k}^T(\boldsymbol{\Lambda} + \mathbf{C})^{-1}\mathbf{k}), \end{aligned} \tag{24}$$

where $\mathbf{k} = [c(\tilde{\mathbf{x}}, \mathbf{x}_1), c(\tilde{\mathbf{x}}, \mathbf{x}_2), \ldots, c(\tilde{\mathbf{x}}, \mathbf{x}_n)]$ and $\kappa = c(\tilde{\mathbf{x}}, \tilde{\mathbf{x}})$. Probability of $\tilde{\mathbf{x}}$ being class $\tilde{y}$ is obtained from (2) as follows:

$$\begin{aligned} p(\tilde{y}|\tilde{\mathbf{x}}, \mathcal{D}, \Theta) &= \int p(\tilde{y}|\tilde{f}, \Theta) p(\tilde{f}|\mathcal{D}, \tilde{\mathbf{x}}, \Theta)\, d\tilde{f} \\ &= \Phi\left(\frac{\tilde{y}\mathbf{k}^T\mathbf{C}^{-1}\mathbf{h}}{\sqrt{\kappa - \mathbf{k}^T(\boldsymbol{\Lambda} + \mathbf{C})^{-1}\mathbf{k})}} \right), \end{aligned} \tag{25}$$

if we assume the test data point does not have labeling errors and

$$p(\tilde{y}|\tilde{\mathbf{x}}, \mathcal{D}, \Theta) = \epsilon + (1 - 2\epsilon)\Phi\left(\frac{\tilde{y}\mathbf{k}^T\mathbf{C}^{-1}\mathbf{h}}{\sqrt{\kappa - \mathbf{k}^T(\boldsymbol{\Lambda} + \mathbf{C})^{-1}\mathbf{k})}} \right), \tag{26}$$

if we assume the test data point might also have labeling errors.

Strict classification of a new data point $\tilde{\mathbf{x}}$ can be done according to

$$\arg\max_{\tilde{y}} p(\tilde{y}|\tilde{\mathbf{x}}, \mathcal{D}, \Theta) = \mathrm{sgn}(E[\tilde{f}]) = \mathrm{sgn}(\mathbf{k}^T\mathbf{C}^{-1}\mathbf{h}).$$

Equivalently, since $\mathbf{h} = \mathbf{C}\boldsymbol{\alpha}$, classification can be conducted according to $\mathrm{sgn}(\sum_{i=1}^n \alpha_i c(\mathbf{x}_i, \tilde{\mathbf{x}}))$ which is the expression found in [11].

More information about EP for GPC can be found on the Computer Society Digital Library at http://computer.org/tpami/archives.htm.

3.3 The EM-EP Algorithm

In the last section, we have presented the EP algorithm for Gaussian process classification. It supplies the posterior over latent functions, the predictive class probability for new data points, and the approximate evidence. One important component missing from the paper so far is an algorithm to estimate the hyperparameters of the covariance function.

We address the problem of estimating hyperparameters of the covariance function in the framework of Gaussian process regression with incomplete target values. This idea makes it possible to apply an EM-like algorithm. In the E-step, we infer the approximate (Gaussian) density for latent function values $q(\mathbf{f})$ using EP. In the M-step, using $q(\mathbf{f})$ obtained in the E-step, we maximize a lower bound on $p(\mathbf{y}|X, \Theta)$ as a function of Θ. The E-step and M-step are alternated until convergence.

- **E-step**. EP iterations are performed given the hyperparameters. $p(\mathbf{f}|\mathcal{D})$ is approximated as a Gaussian density $q(\mathbf{f})$:

$$q(\mathbf{f}) = \mathcal{N}(\mathbf{h}, \mathbf{A}) = \mathcal{N}(\mathbf{C}\boldsymbol{\alpha}, \mathbf{A}). \tag{27}$$

- **M-step**. Given $q(\mathbf{f})$ obtained from the E-step, find the covariance function hyperparameters and the labeling error hyperparameter which maximize a lower bound of the log evidence $\log p(\mathbf{y}|X, \Theta)$. We define $\mathbf{y} = [y_1, y_2, \ldots, y_n]^\top$, let Θ represent all of the hyperparameters of the model: $\epsilon, \theta_0, \theta_1, \theta_2, l_1, l_2, \ldots, l_p$, and let Θ_{cov} represent all those in Θ except for ϵ. Then, we obtain the evidence

$$p(\mathbf{y}|X, \Theta) = \int p(\mathbf{y}|\mathbf{f}, \epsilon) p(\mathbf{f}|X, \Theta_{\mathrm{cov}})\, d\mathbf{f}. \tag{28}$$

Since the above integral is intractable, we use an approximation technique. We take a lower bound F for the log evidence by Jensen's inequality, as follows:

$$\log p(\mathbf{y}|X, \Theta) = \log \int p(\mathbf{y}|\mathbf{f}, \epsilon) p(\mathbf{f}|X, \Theta_{\mathrm{cov}}) d\mathbf{f} \tag{29}$$

$$\geq \int q(\mathbf{f}) \log \frac{p(\mathbf{y}|\mathbf{f}, \epsilon) p(\mathbf{f}|X, \Theta_{\mathrm{cov}})}{q(\mathbf{f})}\, d\mathbf{f} = F. \tag{30}$$

Using the E-step result (27) and the fact that $p(\mathbf{f}|X, \Theta_{\mathrm{cov}}) = \mathcal{N}(\mathbf{0}, \mathbf{C}_\Theta)$ and $\tilde{\mathbf{C}} = \mathbf{C}\mathrm{diag}(\mathbf{y})$, we obtain the following gradient update rule with respect to a covariance hyperparameter $\theta (\in \Theta_{\mathrm{cov}})$:[5]

$$\begin{aligned} \frac{\partial F}{\partial \theta} = &\frac{1}{2}\boldsymbol{\alpha}^\top \frac{\partial \mathbf{C}}{\partial \theta}\boldsymbol{\alpha} - \frac{1}{2} tr\left(\mathbf{C}^{-1}\frac{\partial \mathbf{C}}{\partial \theta} \right) \\ &+ \frac{1}{2} tr\left(\mathbf{C}^{-1}\frac{\partial \mathbf{C}}{\partial \theta}\mathbf{C}^{-1}\mathbf{A} \right). \end{aligned} \tag{31}$$

The detailed derivation is in Appendix A.

3.4 A Property of the EM-EP Algorithm

It turns out that the gradient of the lower bound of the evidence $p(\mathbf{y}|X, \Theta)$ in the M-step of the EM-EP algorithm is in the same direction as the gradient of the approximate evidence obtained by EP when we deal with only the hyperparameters $\boldsymbol{\theta}_{\mathrm{cov}}$ in the prior density. The proof is as follows:

Theorem 1. *In the M-step of the EM-EP algorithm, the gradient of the lower bound F of $p(\mathbf{y}|X, \Theta)$ under $q(\mathbf{f})$ with respect to the hyperparameters $\boldsymbol{\theta}_{\mathrm{cov}}$[6] of the covariance function is in the same direction as the gradient of approximate evidence $\mathcal{Z}(\boldsymbol{\theta}_{\mathrm{cov}})$ ($\approx p(\mathbf{y}|X, \boldsymbol{\theta}_{\mathrm{cov}})$) in (23).*

5. $\frac{\partial \mathbf{C}}{\partial \theta}$ is an elementwise differentiation of $\mathbf{C}$.
6. $\boldsymbol{\theta}_{\mathrm{cov}} = [\theta_0, \theta_1, \theta_2, l_1, l_2, \ldots, l_d]$.

TABLE 1
Comparison of GPCs with Laplace's Method, the Variational Method, and the EM-EP Algorithm

Methods:	Laplace's method	the variational method	the EM-EP algorithm	the true values
l_1	0.3493±0.1336	0.3266±0.1782	0.4559±0.1200	0.5000
l_2	1.2828±0.5171	0.9787±0.5878	1.7389±0.6768	2.0000

Proof. The gradient of F with respect to $\boldsymbol{\theta}_{\text{cov}}$ is expressed:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}_{\text{cov}}} F &= \nabla_{\boldsymbol{\theta}_{\text{cov}}} \int q(\mathbf{f}) \log \frac{p(\mathbf{y}|\mathbf{f},\epsilon)p(\mathbf{f}|X,\boldsymbol{\theta}_{\text{cov}})}{q(\mathbf{f})} \\ d\mathbf{f} &= \nabla_{\boldsymbol{\theta}_{\text{cov}}} \int q(\mathbf{f}) \log p(\mathbf{f}|X,\boldsymbol{\theta}_{\text{cov}})\, d\mathbf{f}.\end{aligned} \tag{32}$$

The gradient of $\mathcal{Z}(\boldsymbol{\theta}_{\text{cov}})$ with respect to $\boldsymbol{\theta}_{\text{cov}}$ is expressed using (23):

$$\begin{aligned}\nabla_{\boldsymbol{\theta}_{\text{cov}}} \log \mathcal{Z}(\boldsymbol{\theta}_{\text{cov}}) &= \nabla_{\boldsymbol{\theta}_{\text{cov}}} \int q(\mathbf{f}) \log \mathcal{Z}(\boldsymbol{\theta}_{\text{cov}}) \\ d\mathbf{f} &= \nabla_{\boldsymbol{\theta}_{\text{cov}}} \int q(\mathbf{f}) \log \frac{\prod_{i=1}^{n} \tilde{t}_i(\mathbf{f}) t_0(\mathbf{f}|\boldsymbol{\theta}_{\text{cov}})}{q(\mathbf{f})}\, d\mathbf{f} \qquad (33)\end{aligned}$$

$$= \nabla_{\boldsymbol{\theta}_{\text{cov}}} \int q(\mathbf{f}) \log t_0(\mathbf{f}|\boldsymbol{\theta}_{\text{cov}})\, d\mathbf{f}. \tag{34}$$

From (32), (34) and the fact that $t_0(\mathbf{f}|\boldsymbol{\theta}_{\text{cov}}) = p(\mathbf{f}|X,\boldsymbol{\theta}_{\text{cov}})$, we obtain the following equation:

$$\nabla_{\boldsymbol{\theta}_{\text{cov}}} F = \nabla_{\boldsymbol{\theta}_{\text{cov}}} \log \mathcal{Z}(\boldsymbol{\theta}_{\text{cov}}) = \frac{1}{\mathcal{Z}(\boldsymbol{\theta}_{\text{cov}})} \nabla_{\boldsymbol{\theta}_{\text{cov}}} \mathcal{Z}(\boldsymbol{\theta}_{\text{cov}}).$$

□

According to Theorem 1, when we use the EM-EP algorithm with only covariance hyperparmeters, the M-step uses the same direction as the gradient of the approximate evidence $\mathcal{Z}(\boldsymbol{\theta}_{\text{cov}})$. On the other hand, when we use the EM-EP algorithm with some hyperparameters related to the likelihood, the M-step does not use the same direction as the gradient of the approximate evidence $\mathcal{Z}(\boldsymbol{\theta}_{\text{cov}})$.

Even though the theoretical justification for the EM-EP algorithm is harder, in practice generally better inference (E-step) should lead to better (hyperparameter) learning. Some examples have shown that the approximate evidence from EP agrees very well with the one from an MCMC method [13]. The EM-EP algorithm is more likely to learn the hyperparameter which is a maximum of the approximate evidence by MCMC method, when its M-step uses the gradient of the approximate evidence (Theorem 1).

3.5 Experimental Results

To demonstrate the EM-EP procedure, we start with hyperparameter learning in synthetic data sets. We then use binary-class real world data sets to compare the proposed algorithm with SVMs and other classification methods. In the M-step, we used the conjugate gradient method with line searches.[7] All covariance function hyperparameters are optimized in log transformed spaces so as to avoid constrained optimization.

3.5.1 Synthetic Data Sets

First, we show with a simple intuitive example that the EM-EP algorithm learns the hyperparameters better than Laplace's method and the variational method. We have sampled a latent function in a two-dimensional input space from a Gaussian process prior with inverse lengthscales 0.5 and 2.0 in the two dimensions. We then sampled 200 data points randomly from a uniform(-10,10) distribution and used the sign of the latent function to define the class labels of the points. Using this data, we learned the hyperparameters of a GPC with Laplace's method [3], the variational method [4], and the EM-EP algorithm. We performed this experiment 10 times for different latent functions. Table 1 shows the means and standard deviation of the lengthscale hyperparameters learned by the three methods. All methods seem to underestimate the lengthscale parameters in (13), which corresponds to assuming functions with longer lengthscales (i.e., more slowly varying). This may indicate underfitting due to limited data. The EM-EP algorithm shows the best results, which are fairly close to the true value.

To show the usefulness of lengthscale hyperparameters, we generated a simple data set with six features distributed as follows: $x_1, x_2, x_3 \sim \mathcal{N}(y, 1)$ and $x_4, x_5, x_6 \sim \mathcal{N}(0, 1)$, where $y \in \{\pm 1\}$ is the class label. That is, x_1, x_2, x_3 are relevant features while x_4, x_5, x_6 are irrelevant to the classification problem. We generated 300 data samples for a training set and 10,000 data samples for a test set. We tried the EM-EP algorithm with a single lengthscale hyperparameter for all dimensions, or with multiple lengthscale hyperparameters. As would be hoped, we saw that the lengthscale hyperparameters for the irrelevant features (x_4, x_5, x_6) decreased to near zero. The approximate log evidence and classification error are shown in Table 2. The result show that GPC with multiple lengthscale hyperparameters was significantly better than one with a single lengthscale, as measured both by classification error rates as well as approximate log evidence $\log \mathcal{Z}(\boldsymbol{\theta}_{\text{cov}})$.

3.5.2 Real-World Data Sets

We applied the proposed algorithm to several real-world data sets. The detailed information for the real-world data sets we used is in Table 3. Thyroid, Heart disease, and Ionosphere data sets were obtained from the UCI Machine

7. The optimization procedure is described in Appendix B in [10] and the code is available from http://www.kyb.tuebingen.mpg.de/bs/people/carl/code/minimize/.

TABLE 2
Comparison of GPCs with a Single Lengthscale Hyperparameter and with Multiple Ones

Methods:	single lengthscale	multiple lengthscale
$\log \mathcal{Z}(\Theta_{\text{cov}})$	-55.4480	-35.4119
Error rate	0.0928	0.0556

TABLE 3
Detailed Information on the Real-World Data Sets

Data set	dis.	con.	classes	data points
Thyroid	0	5	3 (to 2)	215
Heart disease	7	6	2	294
Ionosphere	0	34	2	351
Crabs	2	5	2	200
Pima (Tr)	0	7	2	200
Boston Housing	3	14	2	506

(The items "dis.," "con.," "classes," and "data points" mean the number of discrete variables, continuous variables, classes, and data points in each data set.)

Learning Repository,[8] Crabs, and Pima data sets were obtained from the PRNN site,[9] and Boston Housing data set were obtained from the R software site.[10] The Thyroid data set originally had three classes: "normal," "hyper," and "hypo," but we created a binary classification problem by grouping hyper and hypo into "not normal." The Pima data set has a training set of 200 and two kinds of test sets, but we used only the training set as a whole set for experiments. The Boston Housing data set has 506 data points and 20 variables. It has a pair of duplicated variables, one of which is wrong and the other is a corrected one for one attribute, and has another pair of duplicated variables which are town name and town number. We made a binary-class data set by assigning a class label according to whether housing price is greater than USD25000 or not. So, we actually have 17 variables for our classification problem.

Table 4 and Table 5 show the classification error rates of various methods. Each data set was divided into 10 folds. Each fold was subsequently used as a test set, while the other nine folds were used as a training set. The numbers in Table 4 and Table 5 are the means of those 10 error rates and standard errors on the means. We tried three versions of the EM-EP Gaussian Process Classifier: **GPC-EP(s,soft)** used a *single* lengthscale hyperparameter for all feature dimensions, while **GPC-EP(m,soft)** used a different lengthscale hyperparameter for each feature dimension.[11] Finally, **GPC-EP(s,hard)** was a GPC with a single lengthscale hyperparameter where the decision boundary was "hard" in the sense that the latent function noise parameter θ_2 was fixed to

TABLE 4
Classification Error Rates of Various Methods for Real-World Data Sets (I)

	Heart disease	Thyroid	Ionosphere
1-NN	0.3724±0.0421	0.0561±0.0177	0.1398±0.0159
k-NN	0.3340±0.0237	0.0561±0.0177	0.1427±0.0163
LDA	0.1664±0.0186	0.1251±0.0204	0.1256±0.0226
SVM-CV(hard)	0.2349±0.0241	0.0420±0.0153	0.0568±0.0167
SVM-CV(soft)	0.1943±0.0212	0.0366±0.0224	0.0598±0.0158
SVM-EP(s,soft)	0.2011±0.0289	**0.0229±0.0148**	**0.0429±0.0163**
GPC-VL(m,soft)	0.1905±0.0109	0.0554±0.0186	0.1338±0.0179
GPC-VU(m,soft)	0.1803±0.0109	0.0649±0.0178	0.1395±0.0187
GPC-L(m,soft)	0.2038±0.0112	0.0370±0.0157	0.0686±0.0150
GPC-EP(s,hard)	0.2043±0.0193	0.0277±0.0147	0.0513±0.0141
GPC-EP(s,soft)	0.1568±0.0212	0.0277±0.0147	0.0485±0.0156
GPC-EP(m,soft)	**0.1559±0.0329**	0.0277±0.0147	0.0514±0.0173

TABLE 5
Classification Error Rates of Various Methods for Real-World Data Sets (II)

	Crabs	Pima	Boston
1-NN	0.0750±0.0196	0.2950±0.0254	0.1542±0.0142
k-NN	0.0800±0.0196	0.2850±0.0409	0.1424±0.0154
LDA	0.0550±0.0166	0.2550±0.0288	0.1165±0.0122
SVM-CV(hard)	0.0600±0.0205	0.3400±0.0483	0.0514±0.0251
SVM-CV(soft)	0.0550±0.0146	0.2750±0.0403	0.0534±0.0077
SVM-EP(s,soft)	0.0500±0.0176	0.2550±0.0448	**0.0514±0.0089**
GPC-VL(m,soft)	0.0450±0.0123	0.2500±0.0369	0.0829±0.0130
GPC-VU(m,soft)	0.0350±0.0112	0.2550±0.0396	0.0809±0.0133
GPC-L(m,soft)	0.0350±0.0158	0.2750±0.0453	0.0553±0.0074
GPC-EP(s,hard)	0.0650±0.0158	0.3250±0.0473	0.0534±0.0071
GPC-EP(s,soft)	0.0650±0.0158	**0.2450±0.0448**	0.0534±0.0071
GPC-EP(m,soft)	**0.0250±0.0118**	0.2800±0.0510	0.0573±0.0091

8. Available from http://www.ics.uci.edu/~mlearn/MPRepository.html.

9. Available from http://www.stats.ox.ac.uk/pub/PRNN.

10. Available from http://www.maths.lth.se/help/R/.R/library/spdep/html/boston.html.

11. The initial values of hyperparameters for GPC-EP(s,soft) for the first fold were as follows: $\theta_0^0 = 1$, $\theta_1^0 = 0.0001$, $\theta_2^0 = 0.001$, $l_m^0 = 0.05, \forall m$, and those for subsequent folds are the results for the former fold. For GPC-EP(m,soft), the initial values of the hyperparameters for every fold were the results learned for the same fold in GPC-EP(s,soft).

zero or a very small number. In all GPC models, ϵ (cf (1)) was set to zero.[12]

We also tried other methods for GPC: **GPC-VL(m,soft)** used a variational lower bound, and **GPC-VU(m,soft)** used a variational upper bound to infer latent values [4]. **GPC-L(m,soft)** used Laplace's method to infer latent values [3]. All use an optimization scheme for hyperparameters. All use a multiple lengthscale hyperparameter, but, in the cases of GPC-VL(m,soft) and GPC-VU(m,soft), they used a lengthscale hyperparameter of type $\frac{1}{\sigma_m^2}$ instead of type l_m. Even though they do not have latent value noise hyperparameter θ_2, all have a soft decision boundary, because they use a sigmoid function as a likelihood and they have a signal variance hyperparameter θ_0.

We compared our results to several variants of SVMs.[13] We wanted to distinguish the effect of the kernel choice from the effect of the different loss functions and noise model in SVMs vs GPCs. Thus, in **SVM-EP(s,soft)**, the kernel, (i.e., covariance function) was set to be the same, with the *same* hyperparameters as the corresponding GPC-EP(s,soft) trained using EM-EP except for the latent noise variance θ_2. Instead, the penalty parameter C allowing training errors (i.e., penalizing the SVM slack variables) was selected by five-fold cross-validation.[14]

We also applied both hard and soft-margin SVMs with a Gaussian kernel with a single lengthscale hyperparameter (without θ_0, θ_1, and θ_2) selected by 5-fold cross-validation.[15] For hard-margin SVM, **SVM-CV(hard)**, we only needed to perform a two-level grid search for l. For soft-margin SVMs, **SVM-CV(soft)**, we also had to determine the penalty parameter C, so we performed a two-level grid search over a two-dimensional parameter space (C, l).[16] Finally, for comparison to baseline methods, we also examined the performance of One Nearest Neighbor (**1-NN**), k Nearest Neighbor (k-**NN**)[17] and linear discriminant analysis (**LDA**).

The experimental results (in Table 4 and Table 5) for the three versions of EM-EP applied to GPC models provide interesting insights. GPC with latent function noise (GPC-EP(s,soft)), i.e., which explicitly allows soft boundaries, is better than or as good as the harder version (GPC-EP(s,hard)). This shows that allowing ambiguity at the boundary is important. For these size data sets, the model with multiple lengthscale hyperparameters (GPC-EP(m,-soft)) did not always outperform the single lengthscale model (GPC-EP(s,soft)). However, multiple lengthscales did seem to be essential in learning the Crabs data set, where its error rate was less than half the nearest competitor, and the multiple lengthscale model usually performed among the top methods. For higher-dimensional data sets, fitting too many lengthscale hyperparameters can clearly lead to the danger of overfitting, and it might be advisable to do Bayesian averaging over these parameters.

The experimental results for the three variants of SVMs are also enlightening. The SVM with the same hyperparameters as GPC trained by EM-EP (SVM-EP(s,soft)) is worse than (Heart disease, Crabs, and Pima) or comparable to or slightly better than (Thyroid, Ionosphere, and Boston Housing) the corresponding GPC (GPC-EP(s,soft)). Hard-margin SVM with cross-validation is worse than GPC with a hard decision boundary on four out of six data sets and is slightly better than (or almost as good as) that on the other data sets. In all data sets, GPC with a soft decision boundary (GPC-EP(m,soft) or GPC-EP(s,soft)) is better than or as good as soft-margin SVM (SVM-CV(soft)) with cross-validation. *In all cases, the EM-EP procedure seems to perform better than cross-validation, even when it comes to fitting the SVM kernel hyperparameters*. Moreover, cross-validation would be computationally prohibitive for models with many hyperparameters, such as the multiple lengthscale models.

The experiment results for GPCs with other approximation methods than EP are interesting. GPC-EP (m,soft) is better than GPC-VL(m,soft), GPC-VU(m,soft), and GPC-L(m,soft) on four out of six data sets and is slightly worse than (or almost as good as) the best of them on the other data sets. EP seems to work better than the variational approximation method and Laplace approximation method in Gaussian process classification.

GPC and SVM both have a time complexity of $O(n^3)$, but SVM is usually faster since it uses a sparse scheme. SVM-CV methods are very slow because of the need to solve many quadratic programs during cross validation.

More information about the experiment results of the GPCs can be found on the Computer Society Digital Library at http://computer.org/tpami/archives.htm.

4 Multiclass Classification

In previous sections, we dealt with binary-class Gaussian process classification. Here, we consider a multiclass extension of Gaussian process classification.

4.1 The Traditional Multiclass GPC Formulation

If the data has J classes, each data point has latent functions

$$f^1(\cdot), f^2(\cdot), \ldots, f^J(\cdot).$$

For a data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)|i = 1, 2, \ldots, n\}$, where

$$y_i \in \{1, \ldots, J\},$$

latent function values are represented as

$$\mathbf{f} = [f_1^1, f_2^1, \ldots, f_n^1, f_1^2, f_2^2, \ldots, f_n^2, \ldots, f_1^J, f_2^J, \ldots, f_n^J]^T, \quad (36)$$

where J is the number of classes, n is the number of data points, and f_i^j is $f^j(\mathbf{x}_i)$, a latent function value of ith data point related to class j.

In the literature, the GP prior for multiclass classification has usually been chosen to have only intraclass correlations [3], [27]. The covariance matrix $\mathbf{C}$ for the prior of latent values is defined as

12. An outlier robust classification algorithm with ϵ updated was proposed in [25].

13. Using the MATLAB Support Vector Machine Toolbox available from http://theoval.sys.uea.ac.uk/~gcc/svm/toolbox with modified kernel functions.

14. First, we did a coarse grid search over $\{C|\log_{10} C = 0, 0.5, 1, 1.5, 2, 2.5, 3\}$ to obtain C_1. Then, we did a finer grid search over $\{C|\log_{10} C = -0.4 + \log C_1, -0.3 + \log C_1, \ldots, 0.4 + \log C_1\}$.

15. Similarly to the selection of C, we did a two-level grid search over $\{l|\log_{10} l = -3, -2.5, -2, -1.5, -1, -0.5, 0\}$ and $\{l|\log_{10} l = -0.4 + \log_{10} l_1, -0.3 + \log l_1, \ldots, 0.4 + \log l_1\}$.

16. The same grids as above for parameters C, l were used.

17. k was selected by five-fold cross validation.

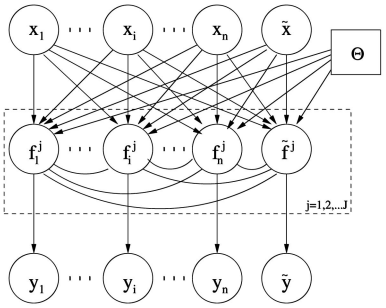


Fig. 3. Graphical model for the traditional multiclass GPC formulation with n training data points and one test data point. $\mathbf{x}_i$ and y_i are observed, $\tilde{\mathbf{x}}$ is given, $\tilde{y}$ is what should be predicted, The function values $\{f_i^j | i = 1, 2, \ldots, n\}$ and $\tilde{f}^j$ in the plate are latent and jointly Gaussian for each j, hence we have the undirected edges, and are conditionally independent over different j, hence we have no edge over different j.

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}_{f^1} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{f^2} & \ldots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ldots & \mathbf{C}_{f^J} \end{bmatrix}, \tag{37}$$

where $\mathbf{C}_{f^j}$ is a covariance matrix of latent values related to class j. The covariance function for covariance matrix $\mathbf{C}$ will be defined as

$$Cov(f_i^j, f_k^l) = \delta(j, l) c(\mathbf{x}_i, \mathbf{x}_k). \tag{38}$$

Since we can assume that the mean is zero, the prior for the latent function values $\mathbf{f}$ is $p(\mathbf{f}) = \mathcal{N}(\mathbf{0}, \mathbf{C})$.

The likelihood term $p(y_i|\mathbf{f}_i)$, where $\mathbf{f}_i = [f_i^1, f_i^2 \ldots f_i^J]$, and $y_i \in \{1, \ldots, J\}$, is defined by using a softmax function as follows:

$$p(y_i|\mathbf{f}_i) = \frac{\exp(f_i^{y_i})}{\sum_j \exp(f_i^j)}. \tag{39}$$

The graphical model for this version of multiclass GPC is shown in Fig. 3.

Now that we have the prior and likelihood for latent values $\mathbf{f}$, we can get the posterior of $\mathbf{f}$ by Bayes' theorem:

$$p(\mathbf{f}|\mathcal{D}, \Theta) = \frac{p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}|X, \Theta)}{p(\mathcal{D}|\Theta)} \propto \prod_i \frac{\exp(f_i^{y_i})}{\sum_j \exp(f_i^j)} p(\mathbf{f}|X, \Theta). \tag{40}$$

The class probability for the new data point $p(\tilde{y}|\tilde{\mathbf{x}})$ can be obtained in the same way as in the binary case.

We review the multiclass GPCs with the traditional GPC formulation. In [3], the latent function was inferred by Laplace's method. They used the Hybrid Monte Carlo method to integrate over the hyperparamters. In [27], the latent function was inferred by variational methods. They maximized the lower bound or the upper bound of the evidence to determine the hyperparameters. In [5], the Markov Chain Monte Carlo method was used both to estimate the posterior of latent function values and to integrate over the hyperparameters. Recently, a sparse approximation method for multiclass classification in the traditional GPC formulation has been proposed in [28]. It uses EP with greedy active set selection of a training set based on information-theoretic criteria.

4.2 A New Multiclass GPC Formulation

We now introduce a different representation for multiclass classification using a new type of latent functions $g^{y_i,j}(\cdot)$, which are differences between $f^{y_i}(\cdot)$ and $f^j(\cdot)$. This makes it possible to straightforwardly extend the EP algorithm for binary-class classification to the multiclass case. Using the notation $g_i^{y_i,j} = g^{y_i,j}(\mathbf{x}_i)$, where $g_i^{y_i,j} = f_i^{y_i} - f_i^j$, we get, using (38):

$$Cov(g_i^{y_i,j}, g_k^{y_k,l}) = E[g_i^{y_i,j} g_k^{y_k,l}] = E[(f_i^{y_i} - f_i^j)(f_k^{y_k} - f_k^l)] \tag{41}$$

$$= E[f_i^{y_i} f_k^{y_k}] - E[f_i^{y_i} f_k^l] - E[f_i^j f_k^{y_k}] + E[f_i^j f_k^l] \tag{42}$$

$$= (\delta(y_i, y_k) - \delta(y_i, l) - \delta(y_k, j) + \delta(j, l)) c(\mathbf{x}_i, \mathbf{x}_k). \tag{43}$$

This makes up the prior $p(\mathbf{g}|X)$.

Using $g_i^{y_i,j} = f_i^{y_i} - f_i^j$, we can rewrite:

$$\begin{aligned} p(y_i|f_i) &= \frac{\exp(f_i^{y_i})}{\sum_{j=1}^J \exp(f_i^j)} = \frac{\exp(f_i^{y_i} - f_i^{y_i})}{\sum_{j=1}^J \exp(f_i^j - f_i^{y_i})} \\ &= \frac{1}{1 + \sum_{j \neq y^i} \exp(-g^{y_i,j})}. \end{aligned} \tag{44}$$

Using the vector $\mathbf{g}$ to denote:

$$\begin{aligned} \mathbf{g} = [&g_1^{y_1,1}, \ldots, g_1^{y_1,y_1-1}, g_1^{y_1,y_1+1}, \ldots, g_1^{y_1,J}, \\ &g_2^{y_2,1}, \ldots, g_2^{y_2,y_2-1}, g_2^{y_2,y_2+1}, \ldots, g_2^{y_2,J}, \\ &\ldots, g_n^{y_n,1}, \ldots, g_n^{y_n,y_n-1}, g_n^{y_n,y_n+1}, \ldots, g_n^{y_n,J}]^T, \end{aligned} \tag{45}$$

we get a whole formulation:

$$\begin{aligned} p(\mathbf{g}|\mathcal{D}) &\propto \left[\prod_i p(y_i|\mathbf{g}_i) \right] p(\mathbf{g}|X, \Theta) \\ &= \left[\prod_i \frac{1}{1 + \sum_{j \neq y_i} \exp(-g^{y_i,j})} \right] p(\mathbf{g}|X, \Theta). \end{aligned} \tag{46}$$

This formulation does not decrease the expressive power of the model. Actually, (39) in Section 4.1 has a troubling redundancy [5]. Neal [5] suggested that the redundancy could be removed and an arbitrary asymmetry into the prior could be produced by forcing one of latent functions to always be zero. If one thinks of the case of $J = 2$, it is clear that the GPC formulation with (39) has an extra redundant latent function comparing to the binary GPC formulation. The formulation in this section does not have this redundancy and becomes equivalent to the binary-class GPC formulation with a sigmoid likelihood function, which can be easily seen when we look at (46).

Similarly to the binary classification case, we can define the likelihood function as $p(y_i|\mathbf{g}_i) = (1 - 2\epsilon) \prod_{j \neq y_i} H(g_i^{y_i,j}) + \epsilon$. If we put $p(y_i|\mathbf{g}_i) = \prod_{j \neq y_i} H(g_i^{y_i,j})$ without ϵ (or when $\epsilon = 0$), the resultant posterior of $\mathbf{g}$ is:

$$p(\mathbf{g}|\mathcal{D}) \propto \left[\prod_i \prod_{j \neq y_i} H(g_i^{y_i,j}) \right] p(\mathbf{g}|X, \Theta). \tag{47}$$

TABLE 6
Detailed Information on the Real-World Data Sets

Data set	dis.	con.	classes	data points
New Thyroid	0	5	3	215
Auto-Mpg	2	5	3	398
Boston Housing	1	12	3	506

(The labels "dis." and "con." mean the number of discrete and continuous variables in each data set, respectively.)

For EP, we can label the prior and likelihood terms in (47):

$$t_0(\mathbf{g}) = p(\mathbf{g}|X, \Theta), \tag{48}$$

$$\begin{aligned} &t_{(i,j)}(\mathbf{g}) = H(g_i^{y_i,j}) \\ &(\text{for } j = 1, \ldots, y_i - 1, y_i + 1, \ldots, J, i = 1, \ldots, n). \end{aligned} \tag{49}$$

Then, by considering $t_{(i,j)}(\mathbf{g})$ as a likelihood term in binary classification, we can apply EP to this multiclass GPC in the same way we applied it to the binary GPC. Likewise, the EM-EP algorithm can be straightforwardly applied to the multiclass GPC. The multiclass versions of the EP and the EM-EP algorithm work well in practice.

For prediction, we need:

$$\begin{aligned} p(\tilde{y}|\tilde{\mathbf{x}}, \mathcal{D}, \Theta) &= \int p(\tilde{y}|\tilde{\mathbf{g}}^{\tilde{y}}, \Theta) p(\tilde{\mathbf{g}}^{\tilde{y}}|\mathcal{D}, \tilde{\mathbf{x}}, \Theta)\, d\tilde{\mathbf{g}}^{\tilde{y}} \\ &= \int \prod_{l \neq \tilde{y}} H(\tilde{g}_l^{\tilde{y}}) p(\tilde{\mathbf{g}}^{\tilde{y}}|\mathcal{D}, \tilde{\mathbf{x}}, \Theta)\, d\tilde{\mathbf{g}}^{\tilde{y}}, \end{aligned} \tag{50}$$

where $\tilde{\mathbf{g}}^{\tilde{y}} = [\tilde{g}^{\tilde{y},1}, \ldots, \tilde{g}^{\tilde{y},\tilde{y}-1}, \tilde{g}^{\tilde{y},\tilde{y}+1}, \ldots, \tilde{g}^{\tilde{y},J}]$, and

$$\begin{aligned} p(\tilde{\mathbf{g}}^{\tilde{y}}|\mathcal{D}, \tilde{\mathbf{x}}, \Theta) &= \int p(\mathbf{g}, \tilde{\mathbf{g}}^{\tilde{y}}|\mathcal{D}, \tilde{\mathbf{x}}, \Theta)\, d\mathbf{g} \\ &= \int p(\tilde{\mathbf{g}}^{\tilde{y}}|\tilde{\mathbf{x}}, \mathbf{g}, \Theta) p(\mathbf{g}|\mathcal{D}, \Theta) d\mathbf{g}. \end{aligned} \tag{51}$$

Using the Gaussian EP approximation to $p(\mathbf{g}|\mathcal{D}, \Theta)$, the terms in (51) are all Gaussian and, so, $p(\tilde{\mathbf{g}}^{\tilde{y}}|\mathcal{D}, \tilde{\mathbf{x}}, \Theta)$ is a $(J-1)$-dimensional Gaussian. Then, the value of (50) is a volume where all variables are positive in the $(J-1)$-dimensional Gaussian. This can be calculated numerically [29] or approximated by EP.

We can consider an approximation scheme to prediction which produces only strict class labels rather than class probabilities. We compute $\tilde{g}^{m,l}$ for every possible pair of classes $m = 1, 2, \ldots, J$, and $l \neq m$:

$$\tilde{g}^{m,l} = \sum_i \sum_{j \neq y_i} \alpha_i^j (\delta(y_i, m) - \delta(y_i, l) - \delta(m, j) + \delta(j, l)) c(\mathbf{x}_i, \tilde{\mathbf{x}}), \tag{52}$$

where α_i^j is α_i in (20) corresponding to $g_i^{y_i,j}$ obtained from EP.

Then, we choose one which most nearly satisfies $\tilde{g}^{m,l} > 0$ for all $l \neq m$ as follows:

$$\tilde{y} = \arg\max_m \tilde{g}^m, \tag{53}$$

TABLE 7
Classification Error Rates of Various Methods for Real-World Data Sets

	New Thyroid	Auto-Mpg	Boston Housing
1-NN	0.0561±0.0177	0.2881±0.0257	0.3260±0.0197
k-NN	0.0561±0.0177	0.2885±0.0231	0.3162±0.0141
LDA	0.0656±0.0234	0.2908±0.0320	0.2609±0.0216
SVM-CV(soft)	0.0556±0.0216	0.2121±0.0183	0.2153±0.0227
GPC-EP(s,soft)	0.0325±0.0145	0.2094±0.0147	**0.2075±0.0200**
GPC-EP(m,soft)	**0.0277±0.0147**	**0.1917±0.0188**	0.2135±0.0158

where $\tilde{g}^m = \sum_{l \neq m} H(\tilde{g}^{m,l})$. In case none of $\tilde{g}^m$ is $J-1$, we cannot be sure which one is the most probable prediction. Also, there can be ties where more than one $\tilde{g}^m$ is maximum.

We also propose a simpler approximation scheme that uses the property that latent function values are $g_i^{j,l} = f_i^j - f_i^l$. We use only latent functions $g_i^{j^*,l}$ for a fixed j^*. Let us consider the case that we only use the latent function values $\tilde{g}^{1,l}$ for class 1 ($j^* = 1$). If all $\tilde{g}^{1,l}$ for $l \neq 1$ are positive, we assign the test data point $\tilde{\mathbf{x}}$ to class 1. Otherwise, we assign it to the class whose correponding latent function value is minimum. The classification scheme can be written as follows:

$$\tilde{y} = \begin{cases} 1 & \text{if } \tilde{g}^{1,l} > 0, \text{ for all } l \neq 1; \\ \arg\min_{l \neq 1} \tilde{g}^{1,l} & \text{otherwise.} \end{cases} \tag{54}$$

Both of those two approximation schemes work well in practice and are simpler than the numerical integral (51), but lose the ability to obtain probabilities for the labels.

4.3 Experimental Results

We applied the multiclass EM-EP GPC to three real-world data sets. The detailed information for the data sets is in Table 6. New Thyroid, Auto-Mpg, and Boston Housing data sets were obtained from the UCI Machine Learning Repository. In the Auto-Mpg data set, the eighth attribute, origin, was used as the class attribute (three classes), and in Boston Housing data set class 1, 2, 3 includes data points where $M \leq 15.4$, $15.4 < M \leq 23.7$, and $M > 23.7$, respectively (M is the 14th attribute, median value of owner-occupied homes in \$1,000s).[18] Table 7 shows the classification error rates of various methods. The experiment protocol including the initial value setting and 10-fold averaging procedure were the same as with the case of binary-class classification (Section 3.5.2). For prediction in GPC, we used the approximation scheme (54) which produces a strict class label. Results for the Laplace method or variational method are not given since no public code for multiclass versions of these methods was found. We did not show the experiment results for the hard decision boundary case, because it is clear from the binary-class experiments

18. This discretization actually creates a three-class ordinal variable, so ordinal regression methods may be more appropriate [30].

that a soft decision boundary almost always outperforms the hard decision boundary case. For the SVM experiments, we used the same software as in the binay-class experiments. The software uses the DAGSVM method [31] for multiclass classification. For LDA, we used $J-1$ discriminant features when we have J classes.

Table 7 shows means of the classification error rates and standard errors on the means. In all three cases, GPC-EP(s,soft) is better than or as good as SVM-CV(s,soft). In the New Thyroid and Auto-Mpg data sets, GPC-EP(m,soft) is better than GPC-EP(s,soft) and in the New Thyroid data set, the classification error of GPC-EP(m,soft) is less than half the nearest competitor.

More information about the experiment results of the multiclass GPCs can be found on the Computer Society Digital Library at http://computer.org/tpami/archives.htm.

5 Conclusion

Based on the work of [11], [12], we presented the EM-EP algorithm for hyperparameter learning in Gaussian process classifiers. Experiments on synthetic and real-world data sets showed the usefulness of hyperparameters related to lengthscales and latent noise. GPC with EM-EP showed better performance than SVM with cross-validation on all the data sets used in the experiments. We derived a new EP method and an EM-EP algorithm for multiclass GPCs and showed that multiclass GPCs had lower test error rate than SVMs with cross-validation on the problems we tested.

Apart from competitive performance, Gaussian Process Classifiers also have some other advantages over nonprobabilistic kernel methods because they are fully statistical models. We can use the evidence for model selection and kernel hyperparameter optimization. Also, given new data, we can get a class probability rather than a hard decision. Even though we did not use it in this paper, prior information can be used to inform learning of the hyperparameters (for example, if some input features are thought to be more relevant or the noise is thought to be high).

The main problem with GPCs, including the EM-EP algorithm presented in the paper, is that it requires $O(n^3)$ computation of matrix inversions during learning. However, sparse approaches for GPC with the EP algorithm have recently been developed [18], [20], [28], [32] that could be applied here. If our multiclass extension is combined with sparse versions of EM-EP and parameterized kernels, it could provide a powerful general classification system. Bayesian versions of SVMs which have sparse solutions have also been proposed in [33], [34], [35], [36], [37]. Although we did not address the issue of reducing computational complexity in this paper, this is clearly an important topic which has received a lot of attention.

To summarize, the main contributions of this paper are the following:

- We have provided a detailed derivation of the EM-EP algorithm for GPCs based on the work in [11], [12], [17] and our own previous work [16], [25] (Appendix).
- We have provided a theorem on the property of the EM-EP algorithm (Section 3.4).
- We have carried out extensive empirical comparisons of EM-EP to other classification methods (NN,LDA), SVM classifiers, and other GPC algorithms (Table 4 and Table 5). EP-based learning of the kernel seems to perform very well compared to other methods.
- We have derived a novel formulation for multiclass classification suitable for EM-EP and tested it empirically (Section 4).

We hope that these contributions will encourage others to explore and further develop the highly flexible Gaussian process models for learning and pattern recognition.

Appendix

M-Step in the EM-EP Algorithm

We take a lower bound for the log evidence by Jensen's inequality as follows:

$$\begin{aligned}\log p(\mathbf{y}|X,\Theta) &= \log \int p(\mathbf{y}|\mathbf{f},\epsilon)p(\mathbf{f}|X,\Theta_{\text{cov}})\, d\mathbf{f} \\ &\geq \int q(\mathbf{f}) \log \frac{p(\mathbf{y}|\mathbf{f},\epsilon)p(\mathbf{f}|X,\Theta_{\text{cov}})}{q(\mathbf{f})}\, d\mathbf{f} = F.\end{aligned} \tag{55}$$

The lower bound F can be written as

$$\begin{aligned}F = &\int q(\mathbf{f}) \log p(\mathbf{y}|\mathbf{f},\epsilon)\, d\mathbf{f} + \int q(\mathbf{f}) \log p(\mathbf{f}|X,\Theta_{\text{cov}})\, d\mathbf{f} \\ &- \int q(\mathbf{f}) \log q(\mathbf{f})\, d\mathbf{f}.\end{aligned}$$

We use F_ϵ, $F_{\Theta_{\text{cov}}}$, and $H(q)$, respectively, to denote the three integrals that make up F in (56). Since $H(q)$ is independent of the hyperparamenter set Θ, and ϵ is independent of Θ_{cov}, we optimize F for Θ, by optimizing $F_{\Theta_{\text{cov}}}$ and F_ϵ for Θ_{cov} and ϵ, respectively.

By expanding $F_{\Theta_{\text{cov}}}$, we get

$$\begin{aligned}F_{\Theta_{\text{cov}}} &= E_q[\log p(\mathbf{f}|X,\Theta_{\text{cov}})] = E_q\left[-\frac{1}{2}\log|2\pi\mathbf{C}| - \frac{1}{2}\mathbf{f}^\top\mathbf{C}^{-1}\mathbf{f}\right] \\ &= -\frac{1}{2}\log|2\pi\mathbf{C}| - \frac{1}{2}E_q[\mathbf{f}^\top\mathbf{C}^{-1}\mathbf{f}] \\ &= -\frac{1}{2}\log|2\pi\mathbf{C}| - \frac{1}{2}E_q[\mathbf{f}]^\top\mathbf{C}^{-1}E_q[\mathbf{f}] - \frac{1}{2}tr(\mathbf{C}^{-1}Cov[\mathbf{f}]).\end{aligned} \tag{56}$$

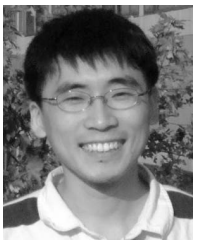
Differentiating $F_{\Theta_{\text{cov}}}$ for θ, using the E-step result (i.e., (27)), we obtain

$$\begin{aligned}\frac{\partial F_{\Theta_{\text{cov}}}}{\partial\theta} &= -\frac{1}{2}tr\left(\mathbf{C}^{-1}\frac{\partial\mathbf{C}}{\partial\theta}\right) + \frac{1}{2}E_q[\mathbf{f}]^\top\mathbf{C}^{-1}\frac{\partial\mathbf{C}}{\partial\theta}\mathbf{C}^{-1}E_q[\mathbf{f}] \\ &\quad + \frac{1}{2}tr\left(\mathbf{C}^{-1}\frac{\partial\mathbf{C}}{\partial\theta}\mathbf{C}^{-1}Cov[\mathbf{f}]\right) \\ &= -\frac{1}{2}tr\left(\mathbf{C}^{-1}\frac{\partial\mathbf{C}}{\partial\theta}\right) + \frac{1}{2}\mathbf{h}^\top\mathbf{C}^{-1}\frac{\partial\mathbf{C}}{\partial\theta}\mathbf{C}^{-1}\mathbf{h} \\ &\quad + \frac{1}{2}tr\left(\mathbf{C}^{-1}\frac{\partial\mathbf{C}}{\partial\theta}\mathbf{C}^{-1}\mathbf{A}\right).\end{aligned} \tag{57}$$

Using $\mathbf{h} = \mathbf{C}\boldsymbol{\alpha}$, we get (31).

REFERENCES

[1] V. Vapnik, *The Nature of Statistical Learning Theory.* Springer, 1995.
[2] R. Herbrich, T. Graepel, and C. Campbell, "Bayes Point Machines," *J. Machine Learning Research,* vol. 1, pp. 245-279, 2001.
[3] C.K.I. Williams and D. Barber, "Bayesian Classification with Gaussian Processes," *IEEE Trans. Pattern Anlysis and Machine Intelligence,* vol. 20, no. 12, pp. 1342-1351, Dec. 1998.
[4] M. Gibbs and D.J.C. MacKay, "Variational Gaussian Process Classifiers," *IEEE Trans. Neural Networks,* vol. 11, no. 6, p. 1458, Nov. 2000.
[5] R. Neal, "Regression and Classification Using Gaussian Process Priors," *Bayesian Statistics 6,* pp. 475-501, 1997.
[6] A. O'Hagan, "On Curve Fitting and Optimal Design for Regression," *J. Royal Statistical Soc.,* vol. 40, pp. 1-42, 1978.
[7] C.K.I. Williams and C.E. Rasmussen, "Gaussian Processes for Regression," *Proc. Neural Information Processing Systems Conf. (NIPS-8),* 1995.
[8] M. Gibbs and D.J. C. MacKay, "Efficient Implementation of Gaussian Processes," draft manuscript (http://citeseer.nj.nec.com/6489.html) 1997.
[9] R. Neal, "Bayesian Learning for Neural Networks," *Lecture Notes in Statistics,* no. 118, 1996.
[10] C.E. Rasmussen, "Evaluation of Gaussian Processes and other Methods for Non-Linear Regression," PhD Thesis, Univ. of Toronto, 1996.
[11] M. Opper and O. Winther, "Gaussian Processes for Classification: Mean Field Algorithms," *Neural Computation,* vol. 12, no. 11, pp. 2655-2684, Nov 2000.
[12] T. Minka, "A Family of Algorithms for Approximate Bayesian Inference," PhD thesis, MIT, Jan. 2001, http://research.microsoft.com/~minka/papers/ep/.
[13] M. Kuss and C. Rasmussen, "Assessing Approximate Inference for Binary Gaussian Process Classification," *J. Machine Learning Research,* vol. 6, pp. 1679-1704, 2005.
[14] L. Csato, E. Fokoue, M. Opper, B. Schottky, and O. Winther, "Efficient Approaches to Gaussian Process Classification," *Proc. Neural Information Processing Systems Conf. (NIPS),* vol. 13, 2000.
[15] B. Krishnapuram, A. Hartemink, L. Carin, and M. Figueiredo, "A Bayesian Approach to Joint Feature Selection and Classifier Design," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 26, no. 9, pp. 1105-1111, Sept. 2004.
[16] H.-C. Kim and Z. Ghahramani, "The EM-EP Algorithm for Gaussian Process Classification," *Proc. Workshop Probabilistic Graphical Models for Classification (ECML),* 2003.
[17] M. Seeger, "Notes on Minka's Expectation Propagation for Gaussian Process Classification," technical report, 2002.
[18] L. Csato and M. Opper, "Sparse Representation for Gaussian Process Models," *Proc. Neural Information Processing Systems Conf. (NIPS),* vol. 13, 2000.
[19] L. Csato, M. Opper, and O. Winther, "TAP Gibbs Free Energy, Belief Propagation and Sparsity," *Proc. Neural Information Processing Systems Conf. (NIPS),* vol. 14, 2001.
[20] M. Seeger, N. Lawrence, and R. Herbrich, "Sparse Representation for Gaussian Process Models," *Proc. Neural Information Processing Systems Conf. (NIPS),* vol. 15, 2002.
[21] T. Minka and J. Lafferty, "Expectation-Propagation for the Generative Aspect Model," *Proc. 18th Conf. Uncertainty in Artificial Intelligence (UAI),* pp. 352-359, 2002.
[22] Y. Qi and T. Minka, "Expectation Propagation for Signal Detection in Flat-Fading Channels," technical report, MIT, 2003.
[23] T. Minka and Y. Qi, "Tree-Structured Approximations by Expectation Propagation," *Proc. Neural Information Processing Systems Conf. (NIPS),* vol. 16, 2003.
[24] T. Heskes and O. Zoeter, "Expectation Propagation for Approximate Inference in Dynamic Bayesian Networks," *Proc. 16th Conf. Uncertainty in Artificial Intelligence (UAI),* pp. 216-223, 2002.
[25] H.-C. Kim, " Bayesian and Ensemble Kernel Classifiers," PhD thesis, POSTECH, Jan. 2005, http://home.postech.ac.kr/~grass/publication/.
[26] G.H. Golub and C.F.V. Loan, *Matrix Computation.* Johns Hopkins Press, 1996.
[27] M.N. Gibbs, "Bayesian Gaussian Processes for Regression and Classification," PhD thesis, Univ. of Cambridge, 1997.
[28] M. Seeger and M.I. Jordan, "Sparse Gaussian Process Classification with Multiple Classes," Technical Report TR 661, Dept. of Statistics, Univ. of California at Berkeley, 2004.
[29] A. Genz, "Numerical Computation of Multivariate Normal Probabilities," *J. Computer Graph Statistics,* vol. 1, pp. 141-149, 1992.
[30] W. Chu and Z. Ghahramani, "Gaussian Processes for Ordinal Regression," *J. Machine Learning Research,* vol. 6, pp. 1019-1041, 2005.
[31] J. Platt, N. Cristianini, and J. Shawe-Taylor, "Large Margin DAGs for Multiclass Classification," *Proc. Neural Information Processing Systems Conf. (NIPS),* pp. 547-553, vol. 12, 2000.
[32] E. Snelson and Z. Ghahramani, "Sparse Parametric Gaussian Processes," *Proc. Neural Information Processing Systems Conf. (NIPS),* vol. 18, 2005.
[33] M. Seeger, "Bayesian Model Selection for Support Vector Machines, Gaussian Processes and Other Kernel Classifiers," *Proc. Neural Information Processing Systems Conf. (NIPS),* vol. 12, pp. 603-609, 2000.
[34] J. Kwok, "Moderating the Outputs of Support Vector Machine Classifiers," *IEEE Trans. Neural Networks,* vol. 10, no. 5, pp. 1018-1031, 1999.
[35] J. Kwok, "The Evidence Framework Applied to Support Vector Machines," *IEEE Trans. Neural Networks,* vol. 11, no. 5, pp. 1162-1173, 2000.
[36] P. Sollich, "Bayesian Methods for Support Vector Machines: Evidence and Predictive Class Probabilities," *Machine Learning,* vol. 46, pp. 21-52, 2002.
[37] W. Chu, "Bayesian Approach to Support Vector Machines," PhD thesis, Nat'l Univ. of Singapore, Jan. 2003.



Hyun-Chul Kim received the BS and BEng degress in 1999, the MEng degree in 2001, and the PhD degree in 2005 from from the POSTECH (Pohang University of Science and Technology). In 2002, he was a visting research student in the Gatsby Computational Neuroscience Unit, University College London. He is now a researcher at the POSTECH Information Technology Laboratories. His current research interests include machine learning, Bayesian statistics, and pattern recognition. He has recently worked on Gaussian processes, graphical models, and financial engineering.

Zoubin Ghahramani received the BA and BSEng degrees from the University of Pennsylvania in 1990, and the PhD degree in 1995 from MIT, advised by Michael I Jordan. He was a postdoctoral fellow in the Artificial Intelligence Lab at the University of Toronto, working with Geoffrey Hinton from 1995-1998. From 1998 to 2005, he was a faculty member at the Gatsby Computational Neuroscience Unit, University College London. He is now a professor of information engineering at the University of Cambridge. He also has an appointment as an associate research professor in the Machine Learning Department at Carnegie Mellon University and is adjunct faculty in the Gatsby Unit, University College London. His current research interests include Bayesian approaches to machine learning, artificial intelligence, statistics, information retrieval, bioinformatics, and computational motor control. He has recently worked on Gaussian processes, nonparametric Bayesian methods, clustering, approximate inference algorithms, graphical models, Monte Carlo methods, and semisupervised learning. He serves on the editorial boards of the *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *Machine Learning*, the *Journal on Machine Learning Research*, the *Journal on Artificial Intelligence Research*, and *Bayesian Analysis*. He is a member of the IEEE and the IEEE Computer Society.