
Notes on Minka's Expectation Propagation for Gaussian process classification

Matthias Seeger
Institute for Adaptive and Neural Computation
University of Edinburgh
5 Forrest Hill, Edinburgh EH1 2QL
seeger@dai.ed.ac.uk

Abstract

This paper contains details and gives some minor extensions of Thomas Minka's *expectation propagation* algorithm, as applied to sparse Gaussian process classification.

1 Introduction

In [4], Thomas Minka introduces *expectation propagation (EP)*, a general scheme for approximate Bayesian inference, and applies it to Bayesian Gaussian process classification (GPC). The scheme can be applied to the problem of *sparse GPC*, resulting in a generalization of the sparse online GPC algorithm proposed by Csato and Oppor [1]. In this paper, we provide some notes concerning this application.

Please note that this paper is not self-contained, but relies heavily on the presentation in [4], especially on chapter 5.

1.1 The Gaussian process classification model

In this paper, we focus on binary classification problems, i.e. discrimination between classes labeled as $-1/+1$. As in [4], we will view GPC as “kernelization” of *Bayesian linear discrimination*. In the latter, we assume the relation $\mathbf{x} \rightarrow u \rightarrow y$, where $\mathbf{x} \in \mathbb{R}^m$ is the input point, $u \in \mathbb{R}$ is the latent output, and $y \in \{-1, +1\}$ is the class label (or the target). The latent output is modelled as *linear function* of \mathbf{x} , i.e. $u = \mathbf{w}^T \mathbf{x}$. In this paper, we employ a *probit noise* model, i.e. assume $P(y|u) = \Phi(yu)$, where Φ is the cumulative distribution function (c.d.f.) of the standard Gaussian distribution $N(0, 1)$. We will write $P(y|\mathbf{x}, \mathbf{w}) = P(y|u = \mathbf{w}^T \mathbf{x})$. We place a Gaussian prior $P(\mathbf{w}) = N(\mathbf{w}|\mathbf{0}, \mathcal{I})$ on the weight vector. Let $D = \{(\mathbf{x}_i, y_i) \mid i = 1, \dots, n\}$ be an i.i.d. training set. We denote the data matrix by $\mathcal{X} = (\mathbf{x}_1 \dots \mathbf{x}_n)$. If we apply our model to D , then exact Bayesian analysis requires us to compute the posterior distribution $P(\mathbf{w}|D) \propto P(D|\mathbf{w})P(\mathbf{w})$, where $P(D|\mathbf{w}) = \prod_i P(y_i|u_i)$, $u_i = \mathbf{w}^T \mathbf{x}_i$. Bayesian prediction amounts to compute the predictive distribution $P(y_*|\mathbf{x}_*, D) = \int P(y_*|\mathbf{x}_*, \mathbf{w})P(\mathbf{w}|D) d\mathbf{w}$. However, exact Bayesian analysis and prediction within this model is not analytically tractable due to the

non-Gaussian noise model. Minka’s EP is a particular technique for performing Bayesian analysis approximately.

A “kernelization” of Bayesian linear discrimination leads to Bayesian GPC. The extension is much the same as the step from linear to generalized linear discrimination, i.e. we replace the input point \mathbf{x} by a *feature vector* $\phi(\mathbf{x})$ living in a *feature space* \mathcal{F} . The model is now assumed to be linear in the feature space (which is a Hilbert space), i.e. the weight vector \mathbf{w} lives in \mathcal{F} . The catch is that for certain feature maps, the symmetric positive semidefinite *kernel function* $K(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$ can be computed without having to do explicit computations in \mathcal{F} .

We follow [4] in that we first develop all we need for Bayesian linear discrimination, then later “kernelize” the expression by replacing all input points by the corresponding feature vectors. If it turns out that the new expressions can be computed without requiring explicit manipulations in \mathcal{F} , the “kernelization” proves successful, leading to a nonlinear generalization of a linear method.

A final note on the scaling of u : in the linear model above, this scaling is fixed arbitrarily due to the prior $P(\mathbf{w}) = N(\mathbf{w}|\mathbf{0}, \mathcal{I})$, and if our goal was indeed to work with this “kernel-free” linear model, we would be better off using $P(\mathbf{w}) = N(\mathbf{w}|\mathbf{0}, \sigma_0^2 \mathcal{I})$, with σ_0^2 being a hyperparameter. However, in the process of “kernelization”, we can equivalently employ a *variance parameter* in the kernel, representing the prior variance of u . We settle for this latter convention.

1.2 Assumed-density filtering

Minka’s EP is a generalization of a particular Bayesian online learning algorithm, which he refers to as *assumed-density filtering (ADF)*. The idea is to propagate a *Gaussian* approximation $Q(\mathbf{w})$ to the true intractable posterior $P(\mathbf{w}|D)$, starting with $Q(\mathbf{w}) = P(\mathbf{w})$, and adding in new points from D once at a time. In order to incorporate a new point (\mathbf{x}_i, y_i) , we first compute $\hat{P}(\mathbf{w}) \propto P(y_i|\mathbf{x}_i, \mathbf{w})Q(\mathbf{w})$. We now choose the new approximating distribution $Q^{new}(\mathbf{w})$ to be the one Gaussian minimizing $D[\hat{P}(\mathbf{w})||\cdot]$, where D denotes the relative entropy. This is equivalent to simply *match* the moments of $Q^{new}(\mathbf{w})$ and $\hat{P}(\mathbf{w})$. Thus, in order to add in a new point, all we have to be able to do is to compute mean and covariance matrix of $\hat{P}(\mathbf{w})$, which can be done analytically if the normalization constant $\int P(y_i|\mathbf{x}_i, \mathbf{w})Q(\mathbf{w}) d\mathbf{w}$ can be computed analytically and is differentiable w.r.t. the parameters of $Q(\mathbf{w})$. This is the only requirement on the noise distribution $P(t|y)$, which is why ADF is considerably more general than other propagation schemes such as the Kalman filter.

As a Bayesian online scheme, ADF can only make use of each datapoint once. This leads to the unfortunate fact that the resulting approximation $Q(\mathbf{w})$, after having included all datapoints, is strongly dependent on the ordering in which points are included. Minka starts off from an alternative view on ADF which allows him to formulate the scheme in a way s.t. the online requirement becomes void. Within this view, all one has to do in order to consider a point again for “inclusion” à la ADF, is to remove its contribution to the approximation prior to this inclusion.

2 Sparse online Gaussian process classification

Minka’s EP for Bayesian linear discrimination iterates “deletion-inclusion” steps over all datapoints in the sample D , until convergence in the parameters of the ap-

proximating distribution $Q(\mathbf{w})$ is reached¹. This is sensible if your goal is to achieve order independence of the final approximation. However, it does not lead to a *sparse* approximation $Q(\mathbf{w})$. Recall that our final goal is a kernelization of the scheme we are dealing with. It turns out in general that kernelized versions of approximate Bayesian linear discrimination techniques employ as final discriminant function, or equivalently as expression for the mean of the (approximate) predictive distribution $P(u_*|\mathbf{x}_*, D)$, a linear combination of the kernel functions $K(\mathbf{x}_*, \mathbf{x}_i)$, $i = 1, \dots, n$. In the context of this paper, we refer to a method (or an approximation $Q(\mathbf{w})$) as *sparse* if its kernelization results in this linear combination being sparse, i.e. most of its coefficients being 0. The number k of (potentially) non-zero coefficients is a controllable parameter. In a sparse GPC technique, prediction time does not scale with n , but rather with k . A further requirement on a sparse GPC technique is that training time scales at most linearly in n . Note that for conventional GPC techniques, training time scales at least as $O(n^2)$ (worst case even $O(n^3)$), and prediction time scales at least as $O(n)$, thus sparse approximations are of large practical interest for very obvious reasons.

The first phase in Minka’s EP is in fact a pure “ADF phase”. As mentioned above, we start with $Q(\mathbf{w}) = P(\mathbf{w})$, and as long as we include points that have not been considered before, there is no need to remove their influence on $Q(\mathbf{w})$ prior to that. In fact, this pure ADF variant of Bayesian linear discrimination has been proposed before by Csató and Opper [1]. Obviously, if we stop ADF after having included k points only, we end up with a sparse discriminant. The final $Q(\mathbf{w})$ and the discriminant will strongly depend on *which* points we have selected. In line with the general online character of the algorithm, and chiefly because exhaustive search over all k subsets of D is clearly intractable, it seems sensible to adopt a greedy approach: in order to decide which point to include next, score all remaining points in D using some heuristic, then choose the point which attains the best score. Such heuristics have been proposed in [1], [3].

The final approximation $Q(\mathbf{w})$ for sparse ADF also depends on the order in which points have been included into the “active” k -subset. This order dependence can be alleviated in a straightforward way, simply by iterating “deletion-inclusion” EP steps, restricted to the active set, until convergence. Another idea, developed further below, is to consider simple “local” modifications to the active set which do not change its size k . For example, if pattern i is in the active set, pattern j is not, the “deletion-inclusion” step can easily be modified to delete i and include j . It is possible to generalize the pure “inclusion” scoring heuristics to apply to such “exchange” steps. There are $O(kn)$ possible “exchange” moves, as opposed to $O(n)$ “inclusion” moves, thus a full greedy search is more expensive roughly by a factor of k . Alternatively, one can consider randomized greedy searches. Note that Csató and Opper [1] propose a different kind of “exchange” moves.

2.1 Expectation Propagation

In this subsection, we reproduce the EP algorithm for Bayesian linear discrimination from chapter 5 of [4]. This is necessary because our setup differs slightly from the one used there, and because we will generalize it further below. However, the exposition here is by no means self-contained.

Attention: We follow the convention in [4] to absorb the class label y_i into the pattern \mathbf{x}_i . This is possible because the likelihood $P(D|\mathbf{w})$ is in fact a function of

¹Unfortunately, EP is not in general a globally convergent method, i.e. it may fail to converge, or may break down resulting in a non-distribution $Q(\mathbf{w})$ (e.g. having negative variances).

$\{y_i \mathbf{x}_i \mid i = 1, \dots, n\}$. Thus, within the remainder of this section, \mathbf{x}_i is a shorthand for $y_i \mathbf{x}_i$, \mathbf{x} a shorthand for $y\mathbf{x}$, etc. We also write $\mathcal{X} = (\mathbf{x}_1 \dots \mathbf{x}_n)$.

The true posterior $P(\mathbf{w}|D)$ can be written as

$$P(\mathbf{w}|D) = P(D)^{-1} P(\mathbf{w}) \prod_{i=1}^n t_i(\mathbf{w}), \quad t_i(\mathbf{w}) = \Phi(\mathbf{w}^T \mathbf{x}_i).$$

At any time between iterations, the current approximating distribution $Q^{new}(\mathbf{w})$ is analogously factored as

$$Q^{new}(\mathbf{w}) \propto P(\mathbf{w}) \prod_{i=1}^n \tilde{t}_i(\mathbf{w}), \quad \tilde{t}_i(\mathbf{w}) = \exp\left(-\frac{1}{2} v_i^{-1} (\mathbf{w}^T \mathbf{x}_i - m_i)^2\right). \quad (1)$$

The principal parameters of Q^{new} are the m_i, v_i^{-1} , and we write $\Lambda^{-1} = \text{diag}(v_i^{-1})_i$, $\mathbf{m} = (m_i)_i$. Note that we allow $v_i^{-1} = 0$.² In fact, we have $v_i^{-1} = 0$ for all patterns \mathbf{x}_i which are not currently in the active set. We denote the set of indexes of patterns in the active set by I , i.e. $v_i^{-1} = 0$ if $i \notin I$. Note that we do not use the parameters s_i of [4]. They are used in EP in order to provide an approximation to the marginal likelihood $P(D)$, which we do not require here³, thus for our purpose the s_i can be fixed to 1 permanently. From (1) it is easy to see that $Q^{new}(\mathbf{w}) = N(\mathbf{w} | \boldsymbol{\mu}_w^{new}, \mathcal{V}_w^{new})$ with

$$\mathcal{V}_w^{new} = (\mathcal{I} + \mathcal{X} \Lambda^{-1} \mathcal{X}^T)^{-1}, \quad \boldsymbol{\mu}_w^{new} = \mathcal{V}_w^{new} \mathcal{X} \Lambda^{-1} \mathbf{m}. \quad (2)$$

A ‘‘deletion-inclusion’’ step for pattern \mathbf{x}_i consists of the deletion of $\tilde{t}_i(\mathbf{w})$ from $Q^{new}(\mathbf{w})$, in order to arrive at $Q(\mathbf{w}) \propto P(\mathbf{w}) \prod_{j \neq i} \tilde{t}_j(\mathbf{w})$, followed by an ADF update including \mathbf{x}_i into $Q(\mathbf{w})$ to arrive at a new $Q^{new}(\mathbf{w})$. The second step amounts to replacing $\tilde{t}_i(\mathbf{w})$ by the normalized ratio $\propto Q^{new}(\mathbf{w})/Q(\mathbf{w})$. The intermediate $Q(\mathbf{w}) = Q^{\setminus i}(\mathbf{w})$ is a Gaussian with mean $\boldsymbol{\mu}_w = \boldsymbol{\mu}_w^{\setminus i}$ and covariance matrix $\mathcal{V}_w = \mathcal{V}_w^{\setminus i}$. Here, we follow the notation in [4] and [1], using the superscript $\setminus i$ to indicate the removal of pattern \mathbf{x}_i . Note that this superscript is frequently left away in formulae in [4], if it can be inferred from the context. As in [4], we define

$$\begin{aligned} \lambda_i &= \lambda_i^{\setminus i} = \mathbf{x}_i^T \mathcal{V}_w^{\setminus i} \mathbf{x}_i, \\ \mathcal{C} &= \mathcal{X}^T \mathcal{X} = (\mathbf{x}_i^T \mathbf{x}_j)_{i,j}, \\ h_i &= \mathbf{x}_i^T \boldsymbol{\mu}_w^{new}, \quad h_i^{\setminus i} = \mathbf{x}_i^T \boldsymbol{\mu}_w^{\setminus i}, \\ \mathcal{A} &= \mathcal{X}^T \mathcal{V}_w^{new} \mathcal{X} = (\mathcal{C}^{-1} + \Lambda^{-1})^{-1}. \end{aligned}$$

Let us first see how to remove $\tilde{t}_i(\mathbf{w})$ from $Q^{new}(\mathbf{w})$. If $i \notin I$, we have $Q(\mathbf{w}) = Q^{new}(\mathbf{w})$, thus we have to do nothing (i.e. $\lambda_i = [\mathcal{A}]_{i,i}$ and $h_i^{\setminus i} = h_i$). Otherwise observe from (2) that $\mathcal{V}_w^{-1} = (\mathcal{V}_w^{new})^{-1} - v_i^{-1} \mathbf{x}_i \mathbf{x}_i^T$, thus

$$\mathcal{V}_w = \mathcal{V}_w^{new} + \mathcal{V}_w^{new} \mathbf{x}_i (v_i - [\mathcal{A}]_{i,i})^{-1} \mathbf{x}_i^T \mathcal{V}_w^{new},$$

and we obtain $\lambda_i = \mathbf{x}_i^T \mathcal{V}_w \mathbf{x}_i = (v_i^{-1} + [\mathcal{A}]_{i,i}^{-1})^{-1}$. Furthermore,

$$(\mathcal{V}_w^{new})^{-1} \boldsymbol{\mu}_w^{new} = \mathcal{X} \Lambda^{-1} \mathbf{m} = \mathcal{V}_w^{-1} \boldsymbol{\mu}_w + v_i^{-1} m_i \mathbf{x}_i,$$

²Even $v_i^{-1} < 0$ for some patterns is allowed. However, the algorithm can break down if a significant numbers of the v_i^{-1} become negative.

³This approximation could be used for model selection, or even to define a new heuristic to score inclusion candidates. These possibilities are not explored in this paper.

thus $\boldsymbol{\mu}_w = \boldsymbol{\mu}_w^{new} + v_i^{-1}(h_i - m_i)\mathcal{V}_w \mathbf{x}_i$, and $h_i^{\setminus i} = h_i + \lambda_i v_i^{-1}(h_i - m_i)$.

The noise model $P(y|u) = \Phi(yu)$ we use here differs from the flip noise employed in [4]. The partition function (for ADF inclusion of pattern \mathbf{x}_i) is

$$\begin{aligned} Z_i &= \int \Phi(\mathbf{w}^T \mathbf{x}_i) Q(\mathbf{w}) d\mathbf{w} = \int \Phi(u_i) N(u_i | h_i^{\setminus i}, \lambda_i) du_i \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{u_i} N(v|0,1) N(u_i | h_i^{\setminus i}, \lambda_i) dv du_i \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^0 N(v - u_i, 1) N(u_i | h_i^{\setminus i}, \lambda_i) dv du_i \\ &= \int_{-\infty}^0 N(v - h_i^{\setminus i}, 1 + \lambda_i) dv = \Phi\left(\frac{h_i^{\setminus i}}{\sqrt{1 + \lambda_i}}\right). \end{aligned}$$

Given $\log Z_i$, we can use the formulae (5.11) to (5.13) in [4] to compute the ADF update. If

$$z_i = \frac{h_i^{\setminus i}}{\sqrt{1 + \lambda_i}}, \quad \alpha_i = \frac{\partial \log Z_i}{\partial h_i^{\setminus i}} = \frac{N(z_i | 0, 1)}{\Phi(z_i) \sqrt{1 + \lambda_i}}, \quad (3)$$

then the ADF update is given by

$$\begin{aligned} \boldsymbol{\mu}_w^{new} &= \boldsymbol{\mu}_w + \mathcal{V}_w \alpha_i \mathbf{x}_i, \quad h_i^{new} = h_i^{\setminus i} + \lambda_i \alpha_i, \\ \mathcal{V}_w^{new} &= \mathcal{V}_w - \mathcal{V}_w \mathbf{x}_i \left(\frac{\alpha_i (\alpha_i + h_i^{new})}{1 + \lambda_i} \right) (\mathcal{V}_w \mathbf{x}_i)^T. \end{aligned}$$

Lawrence and Herbrich [3] suggest a simple and very efficient heuristic to score inclusion candidates \mathbf{x}_i . Namely, they argue for the inclusion of the most ‘‘informative’’ pattern, the one which reduces the entropy of the posterior approximation $Q^{new}(\mathbf{w})$ most. This is equivalent to scoring \mathbf{x}_i by $\log |\mathcal{V}_w^{new}|$ *after* its inclusion, and this score can easily be computed as follows:

$$\begin{aligned} \log |\mathcal{V}_w^{new}| &= \log \left(1 - \frac{\alpha_i (\alpha_i + h_i^{new})}{1 + \lambda_i} \lambda_i \right) + c \\ &= \log \left(\frac{1 + \lambda_i (1 - \alpha_i (\alpha_i + h_i^{new}))}{1 + \lambda_i} \right) + c, \end{aligned} \quad (4)$$

where c does not depend on \mathbf{x}_i . Since also $h_i^{new} = h_i^{\setminus i} + \lambda_i \alpha_i$, the criterion (4) can be computed in $O(1)$. We select the pattern which minimizes this score.

The EP algorithm starts with the initialization $\Lambda^{-1} = \mathbf{0}$, $\mathbf{m} = (m_i)_i = \mathbf{0}$, i.e. with $\boldsymbol{\mu}_w^{new} = \mathbf{0}$ and $\mathcal{V}_w^{new} = \mathcal{I}$, furthermore $\mathbf{h} = (h_i)_i = \mathbf{0}$ and $\mathcal{A} = \mathcal{C}$.

A ‘‘deletion-insertion’’ step for pattern \mathbf{x}_i is done as follows:

1. Remove \tilde{t}_i from Q^{new} to obtain Q :

$$\begin{aligned} \lambda_i &= ([\mathcal{A}]_{i,i}^{-1} + v_i^{-1})^{-1}, \\ h_i^{\setminus i} &= h_i + \lambda_i v_i^{-1} (h_i - m_i) \end{aligned}$$

If $i \notin I$, this simplifies to $\lambda_i = [\mathcal{A}]_{i,i}$ and $h_i^{\setminus i} = h_i$.

2. Compute new posterior approximation Q^{new} (ADF step):

$$z_i = \frac{h_i^{\setminus i}}{\sqrt{1 + \lambda_i}}, \quad \alpha_i = \frac{N(z_i|0, 1)}{\Phi(z_i)\sqrt{1 + \lambda_i}},$$

$$h_i^{new} = h_i^{\setminus i} + \lambda_i \alpha_i.$$

At this point, we can compute the scoring criterion of Lawrence and Herbrich as given in (4). If we only want to score \mathbf{x}_i , we can stop here.

$$v_i^{-1} = \frac{\alpha_i(\alpha_i + h_i^{new})}{1 + \lambda_i(1 - \alpha_i(\alpha_i + h_i^{new}))}, \quad (5)$$

$$m_i = h_i^{new} + v_i \alpha_i = h_i^{\setminus i} + \frac{1 + \lambda_i}{\alpha_i + h_i^{new}}.$$

3. Recompute \mathcal{A} and $\mathbf{h} = (h_j)_j$:

$$\mathcal{A} = (\mathcal{C}^{-1} + \Lambda^{-1})^{-1},$$

$$\mathbf{h} = \mathcal{A} \Lambda^{-1} \mathbf{m}.$$

The recomputation of \mathcal{A} and \mathbf{h} in step 3 can be done efficiently in several different ways. For “deletion-inclusion” steps, we follow the recommendation in [4] (see below), while for the initial pure “inclusion” steps (i.e. $i \notin I$), the following incremental Cholesky technique may prove numerically more stable. Let $I_k = I$ be the current active set of size k , and denote $\mathcal{C}_k = [\mathcal{C}]_{I_k, I_k}$, $\mathcal{C}_{n,k} = [\mathcal{C}]_{\cdot, I_k}$, $\Lambda_k = [\Lambda]_{I_k, I_k}$. If $\mathcal{E}_k \in \mathbb{R}^{n,k}$ s.t. $[\mathcal{E}_k]_{I_k, \cdot} = \mathcal{I}_k$, $[\mathcal{E}_k]_{j, \cdot} = \mathbf{0}^T$ for $j \notin I_k$, then $\mathcal{A} = (\mathcal{C}^{-1} + \mathcal{E}_k \Lambda_k^{-1} \mathcal{E}_k^T)^{-1}$. Using the Woodbury formula (see [2], section 2.7), we have

$$\mathcal{A} = \mathcal{C} - \mathcal{C} \mathcal{E}_k (\Lambda_k + \mathcal{E}_k^T \mathcal{C} \mathcal{E}_k)^{-1} \mathcal{E}_k^T \mathcal{C}$$

$$= \mathcal{C} - \mathcal{C}_{n,k} \Lambda_k^{-1/2} \left(\mathcal{I}_k + \Lambda_k^{-1/2} \mathcal{C}_k \Lambda_k^{-1/2} \right)^{-1} \Lambda_k^{-1/2} \mathcal{C}_{n,k}^T. \quad (6)$$

Thus, if we have the Cholesky decomposition $\mathcal{I}_k + \Lambda_k^{-1/2} \mathcal{C}_k \Lambda_k^{-1/2} = \mathcal{L}_k \mathcal{L}_k^T$, then $\mathcal{A} = \mathcal{C} - \mathcal{M}_k^T \mathcal{M}_k$, where $\mathcal{M}_k = \mathcal{L}_k^{-1} \Lambda_k^{-1/2} \mathcal{C}_{n,k}^T$ is obtained by backsubstitution. Note that although we used the Woodbury formula as a rewriting tool, we never compute the inverse in (6) explicitly. Now suppose $i \notin I_k$, and $I_{k+1} = I_k \cup \{i\}$. Then, it is easy to see that \mathcal{L}_k and \mathcal{M}_k can be updated as follows. Let $\mathbf{f} = v_i^{-1/2} \Lambda_k^{-1/2} [\mathcal{C}]_{I_k, i}$, $f = 1 + v_i^{-1} [\mathcal{C}]_{i, i}$, the new elements in $\mathcal{I}_{k+1} + \Lambda_{k+1}^{-1/2} \mathcal{C}_{k+1} \Lambda_{k+1}^{-1/2}$. Then, \mathcal{L}_{k+1} is obtained from \mathcal{L}_k by adding the row $(l^T l)$, with $l = \mathcal{L}_k^{-1} \mathbf{f}$ and $l = \sqrt{f - l^T l}$. Furthermore, \mathcal{M}_{k+1} is obtained from \mathcal{M}_k by adding the row \mathbf{m}_{k+1}^T , where $\mathbf{m}_{k+1} = l^{-1} (v_i^{-1/2} [\mathcal{C}]_{\cdot, i} - \mathcal{M}_k^T l)$. And $\mathcal{M}_{k+1}^T \mathcal{M}_{k+1} = \mathcal{M}_k^T \mathcal{M}_k + \mathbf{m}_{k+1} \mathbf{m}_{k+1}^T$. Thus, $\mathcal{A}^{new} = \mathcal{A} - \mathbf{m}_{k+1} \mathbf{m}_{k+1}^T$. For the update of \mathbf{h} , let $\mathbf{b} = \Lambda^{-1} \mathbf{m}$ and $\mathbf{b}^{new} = \mathbf{b}^{old} + \beta \boldsymbol{\delta}_i$, where $\beta = (v_i^{new})^{-1} m_i^{new}$ and $\boldsymbol{\delta}_i = [\mathcal{I}_n]_{\cdot, i}$. Then,

$$\mathbf{h}^{new} = \mathbf{h} + \beta \mathcal{A} \boldsymbol{\delta}_i - (\mathbf{m}_{k+1}^T \mathbf{b}^{new}) \mathbf{m}_{k+1}.$$

Here, $\mathcal{A} \boldsymbol{\delta}_i = [\mathcal{C}]_{\cdot, i} - \mathcal{M}_k^T [\mathcal{M}_k]_{\cdot, i}$. Note that during this initial phase, we do not need to maintain and update $[\mathcal{A}]_{\cdot, I}$. The only part of \mathcal{A} which is explicitly required, is $\text{diag } \mathcal{A}$.

Unfortunately there seems to be no easy way to update the Cholesky decomposition for a “deletion-inclusion” step, i.e. $i \in I$. In this case, we follow the recommendation

in [4], writing $\mathcal{A}^{new} = (\mathcal{A}^{-1} + \Delta^{-1}\boldsymbol{\delta}_i\boldsymbol{\delta}_i^T)^{-1}$, where $\boldsymbol{\delta}_i = (\delta_{i,j})_j = [\mathcal{I}_n]_{\cdot,i}$, and $\Delta^{-1} = (v_i^{new})^{-1} - (v_i^{old})^{-1}$. Using the Woodbury formula, we have

$$\mathcal{A}^{new} = \mathcal{A} - (\Delta + a_{i,i})^{-1}\mathbf{a}_i\mathbf{a}_i^T, \quad (7)$$

where $\mathcal{A} = \mathcal{A}^{old} = (\mathbf{a}_1 \dots \mathbf{a}_n)$ and $\text{diag } \mathcal{A} = (a_{i,i})_i$. Note that one source of instability here is $(v_i^{new})^{-1} \approx (v_i^{old})^{-1}$. In this case, we recommend not to do the update for pattern \mathbf{x}_i at all, but rather to select another one. If v_j^{-1} remains the same for all $j \in I$, we have attained convergence on the active set. As for the update of \mathbf{h} , let $\mathbf{b} = \Lambda^{-1}\mathbf{m}$, and $\mathbf{b}^{new} = \mathbf{b}^{old} + \beta\boldsymbol{\delta}_i$. Using (7), we arrive at

$$\mathbf{h}^{new} = \mathbf{h} + \frac{\beta\Delta - \mathbf{a}_i^T\mathbf{b}^{old}}{\Delta + a_{i,i}}\mathbf{a}_i.$$

Finally note that at any time, we only need to know a part of \mathcal{A} , namely $\text{diag } \mathcal{A}$ and $[\mathcal{A}]_{\cdot,I}$, and to this end, we only need to evaluate $\text{diag } \mathcal{C}$ and $[\mathcal{C}]_{\cdot,I}$. Thus, each pure ‘inclusion’ step requires the evaluation of one new row of \mathcal{C} . It is also clear that the running time requirements per iteration are dominated by the updates of \mathcal{A} and \mathbf{h} , which cost $O(nk)$. Storage requirements are $O(nk)$.

2.2 Kernelization. Prediction

We have been careful to formulate the algorithm in the previous subsection without using variables living in the input space \mathbb{R}^m . Kernelization amounts to replace patterns \mathbf{x}_i by $y_i\phi(\mathbf{x}_i)$. In our case, the job is done by simply setting $\mathcal{C} = (y_i y_j K(\mathbf{x}_i, \mathbf{x}_j))_{i,j}$. If we define the diagonal matrix $\mathcal{Y} = \text{diag}(y_i)_i$ and the kernel matrix $\mathcal{K} = (K(\mathbf{x}_i, \mathbf{x}_j))_{i,j}$, we have $\mathcal{C} = \mathcal{Y}\mathcal{K}\mathcal{Y}$.

What about prediction? The true intractable predictive distribution $P(y_*|\mathbf{x}_*, D)$ can be approximated by plugging in our Gaussian posterior approximation $Q(\mathbf{w}|D)$ for the true posterior $P(\mathbf{w}|D)$. If $u_* = \mathbf{w}^T \mathbf{x}_*$, $\mathbf{w} \sim Q(\mathbf{w}|D) = N(\mathbf{w}|\boldsymbol{\mu}_w^{new}, \mathcal{V}_w^{new})$, then $u_* \sim N(u_*|(\boldsymbol{\mu}_w^{new})^T \mathbf{x}_*, \mathbf{x}_*^T \mathcal{V}_w^{new} \mathbf{x}_*)$. Now, using (2) and the notation introduced at the end of the previous subsection (for $I_k = I$), it is easy to see that $Q(u_*|\mathbf{x}_*, D) = N(u_*|\mu_*, \sigma_*^2)$, $\mu_* = \boldsymbol{\beta}^T \mathbf{k}(\mathbf{x}_*)$, $\sigma_*^2 = K(\mathbf{x}_*, \mathbf{x}_*) - (\Lambda_k^{-1/2} \mathbf{k}(\mathbf{x}_*))^T \mathcal{B}^{-1} (\Lambda_k^{-1/2} \mathbf{k}(\mathbf{x}_*))$, where $\mathbf{k}(\mathbf{x}_*) = (K(\mathbf{x}_*, \mathbf{x}_i))_{i \in I}$, and

$$\mathcal{B} = \mathcal{I}_k + \Lambda_k^{-1/2} [\mathcal{K}]_{I,I} \Lambda_k^{-1/2}, \quad \boldsymbol{\beta} = \Lambda_k^{-1/2} \mathcal{B}^{-1} \Lambda_k^{-1/2} [\mathcal{T} \mathbf{m}].$$

If we compute the Cholesky decomposition $\mathcal{B} = \mathcal{L}\mathcal{L}^T$, then $\boldsymbol{\beta}$ can be computed by backsubstitution. Furthermore, $\sigma_*^2 = K(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{a}^T \mathbf{a}$, $\mathbf{a} = \mathcal{L}^{-1} \Lambda_k^{-1/2} \mathbf{k}(\mathbf{x}_*)$. Finally, the approximate predictive distribution is

$$Q(y_*|\mathbf{x}_*, D) = \int P(y_*|u_*)Q(u_*|D) du_* = \Phi\left(\frac{y_*\mu_*}{\sqrt{1 + \sigma_*^2}}\right).$$

It is also clear that the approximate Bayes discriminant is simply $\text{sgn } \mu_* = \text{sgn}(\boldsymbol{\beta}^T \mathbf{k}(\mathbf{x}_*))$.

2.3 Exchange moves

Suppose we want to exchange a pattern \mathbf{x}_j , $j \in I$ against a pattern \mathbf{x}_i , $i \notin I$, i.e. simultaneously remove j from and add i to the active set. This can be done by a slight generalization of the ‘deletion-inclusion’ move of EP (see subsection 2.1). Candidates (j, i) can be stored in much the same way as an inclusion candidate.

Here, we follow Lawrence and Herbrich [3] and score (j, i) by the entropy of $Q^{new}(\mathbf{w})$ after the exchange move.

Note that now $Q(\mathbf{w}) = Q^{\setminus j}(\mathbf{w})$. We require the following definitions:

$$\lambda_i^{\setminus j} = \mathbf{x}_i^T \mathcal{V}_w^{\setminus j} \mathbf{x}_i, \quad h_i^{\setminus j} = \mathbf{x}_i^T \boldsymbol{\mu}_w^{\setminus j}.$$

Let $\mathcal{A} = (a_{i,j})_{i,j}$. Using (5.42) of [4], we obtain

$$\lambda_i^{\setminus j} = \frac{a_{i,i} + v_j^{-1}(a_{i,j}^2 - a_{i,i}a_{j,j})}{1 - v_j^{-1}a_{j,j}}. \quad (8)$$

And (5.44), (5.45) of [4] gives us

$$h_i^{\setminus j} = h_i + \frac{a_{i,j}v_j^{-1}(h_j - m_j)}{1 - v_j^{-1}a_{j,j}}. \quad (9)$$

The ADF update from $Q^{\setminus j}(\mathbf{w})$ to $Q^{new}(\mathbf{w})$ is given by

$$\boldsymbol{\mu}_w^{new} = \boldsymbol{\mu}_w^{\setminus j} + \mathcal{V}_w^{\setminus j} \alpha_i \mathbf{x}_i, \quad \mathcal{V}_w^{new} = \mathcal{V}_w^{\setminus j} + \mathcal{V}_w^{\setminus j} \mathbf{x}_i \left(\frac{\alpha_i(\alpha_i + h_i^{new})}{1 + \lambda_i^{\setminus j}} \right) \left(\mathcal{V}_w^{\setminus j} \mathbf{x}_i \right)^T.$$

Here, z_i and α_i are defined as in (3), however $h_i^{\setminus i}$, λ_i have to be replaced by $h_i^{\setminus j}$, $\lambda_i^{\setminus j}$. Furthermore, $h_i^{new} = h_i^{\setminus j} + \lambda_i^{\setminus j} \alpha_i$. Note that the dependence of z_i , α_i and h_i^{new} on j is not made explicit in the notation. The Lawrence/Herbrich score is computed as in (4), and the update of v_i^{-1} , m_i as in (5), where again $h_i^{\setminus i}$, λ_i have to be replaced by $h_i^{\setminus j}$, $\lambda_i^{\setminus j}$. We also have to reset $\tilde{t}_j(\mathbf{w}) \equiv 1$, i.e. $v_j^{-1} = 0$, $m_j = 0$.

\mathcal{A} can be updated efficiently using the Woodbury formula. Let $\Delta_i^{-1} = (v_i^{new})^{-1} - (v_i^{old})^{-1} = (v_i^{new})^{-1}$, $\Delta_j^{-1} = (v_j^{new})^{-1} - (v_j^{old})^{-1} = -(v_j^{old})^{-1}$, $\Delta = \text{diag}(\Delta_i \Delta_j)^T$. Then:

$$\mathcal{A}^{new} = \mathcal{A} - (\mathbf{a}_i \mathbf{a}_j) \Delta^{-1/2} \left(\mathcal{I}_2 + \Delta^{-1/2} [\mathcal{A}]_{\{ij\},\{ij\}} \Delta^{-1/2} \right)^{-1} \Delta^{-1/2} (\mathbf{a}_i \mathbf{a}_j)^T.$$

Furthermore, if $\mathbf{b} = \Lambda^{-1} \mathbf{m}$ as above, and $\mathbf{b}^{new} = \mathbf{b}^{old} + \beta_i \boldsymbol{\delta}_i + \beta_j \boldsymbol{\delta}_j$, $\boldsymbol{\beta} = (\beta_i \beta_j)^T$, then

$$\mathbf{h}^{new} = \mathbf{h} + (\mathbf{a}_i \mathbf{a}_j) \boldsymbol{\nu},$$

$$\boldsymbol{\nu} = \Delta^{-1/2} \left(\mathcal{I}_2 + \Delta^{-1/2} [\mathcal{A}]_{\{ij\},\{ij\}} \Delta^{-1/2} \right)^{-1} \left(\Delta^{1/2} \boldsymbol{\beta} + \Delta^{-1/2} (\mathbf{a}_i \mathbf{a}_j)^T \mathbf{b}^{old} \right).$$

Acknowledgments

The author gratefully acknowledges support through a research studentship from *Microsoft Research Ltd.*

References

- [1] Lehel Csató and Manfred Opper. Sparse online Gaussian processes. Technical report, NCRG, Aston University, 2001. To appear in *Neural Computation*.
- [2] Brian P. Flannery, William H. Press, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C*. Cambridge University Press, 2nd edition, 1992.
- [3] Neil Lawrence and Ralf Herbrich. A sparse Bayesian compression scheme - the Informative Vector machine. Presented at NIPS 2001 workshop on kernel methods, 2001.
- [4] Thomas Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, Massachusetts Institute of Technology, 2001.