Sparse On-Line Gaussian Processes

Lehel Csató
csatol@aston.ac.uk
Manfred Opper
opperm@aston.ac.uk
Neural Computing Research Group,
Department of Information Engineering, Aston University,
B4 7ET Birmingham, U. K.

We develop an approach for sparse representations of gaussian process (GP) models (which are Bayesian types of kernel machines) in order to overcome their limitations for large data sets. The method is based on a combination of a Bayesian on-line algorithm, together with a sequential construction of a relevant subsample of the data that fully specifies the prediction of the GP model. By using an appealing parameterization and projection techniques in a reproducing kernel Hilbert space, recursions for the effective parameters and a sparse gaussian approximation of the posterior process are obtained. This allows for both a propagation of predictions and Bayesian error measures. The significance and robustness of our approach are demonstrated on a variety of experiments.

1 Introduction

Gaussian processes (GP) (Bernardo & Smith, 1994; Williams & Rasmussen, 1996) provide promising Bayesian tools for modeling real-world statistical problems. Like other kernel-based methods, such as support vector machines (SVMs) (Vapnik, 1995), they combine a high flexibility of the model by working in often infinite-dimensional feature spaces with the simplicity that all operations are "kernelized"—performed in the lower-dimensional input space using positive definite kernels.

An important advantage of GPs over other non-Bayesian models is the explicit probabilistic formulation of the model. This allows the modeler to assess the uncertainty of the predictions by providing Bayesian confidence intervals (for regression) or posterior class probabilities (for classification). It also opens the possibility of treating a variety of other nonstandard data models (e.g., quantum inverse statistics, Lemm, Uhlig, & Weiguny, 2000; wind-fields, Evans, Cornford, & Nabney, 2000; Berliner, Wikle, & Cressie, 2000) using a kernel method.

Neural Computation **14, 641–668** (2002)

GPs are nonparametric in the sense that the "parameters" to be learned are functions f_x of a usually continuous input variable $x \in R^d$. The value f_x is used as a latent variable in a likelihood $P(y|f_{x,x})$ that denotes the probability of an observable output variable y given the input x. The a priori assumption on the statistics of f is that of a gaussian process: any finite collection of random variables f_i is jointly gaussian. Hence, one must specify the prior means and the prior covariance function of the variables f_x. The latter is called the kernel $K_0(x, x') = Cov(y, y')$ (Vapnik, 1995; Kimeldorf & Wahba, 1971). Thus, if a zero-mean GP is assumed, the kernel K_0 fully specifies the entire prior information about the model. Based on a set of input-output observations (x_n, y_n) $(n = 1, \ldots, N)$, the Bayesian approach computes the posterior distribution of the process f_x using the prior and the likelihood (Williams, 1999; Williams & Rasmussen, 1996; Gibbs & MacKay, 1999).

A straightforward application of this simple appealing idea is impeded by two major obstacles: nongaussianity of the posterior process and the size of the kernel matrix $K_0(x_i, x_j)$. A first obvious problem stems from the fact that the posterior process is usually nongaussian (except when the likelihood itself is gaussian in the f_x). Hence, in many important cases, its analytical form precludes an exact evaluation of the multidimensional integrals that occur in posterior averages. Nevertheless, various methods have been introduced to approximate these averages. A variety of such methods may be understood as approximations of the nongaussian posterior process by a gaussian one (Jaakkola & Haussler, 1999; Seeger, 2000); for instance, in Williams and Barber (1998), the posterior mean is replaced by the posterior maximum (MAP), and information about the fluctuations is derived by a quadratic expansion around this maximum. The computation of these approximations, which become exact for regression with gaussian noise, require the solution of a system of coupled nonlinear equations of the size equal to the number of data points. The second obstacle that prevents GPs from being applied to large data sets is that the matrix that couples these equations is typically not sparse.

Hence, the development of good sparse approximations is of major importance. Such approximations aim at performing the most time-consuming matrix operations (inversions or diagonalizations) only on a representative subset of the training data. In this way, the computational time is reduced from $\mathcal{O}(N^3)$ to $\mathcal{O}(Np^2)$, where N is the total size of the training data and p is the size of the representative set. The memory requirement is $\mathcal{O}(p^2)$ as opposed to $\mathcal{O}(N^2)$. So far, a variety of sparsity techniques (Smola & Schölkopf, 2000; Williams & Seeger, 2001) for batch training of GPs have been proposed. This article presents a new approach that combines the idea of a sparse representation with an on-line algorithm that allows for a speedup of the GP training by sweeping through a data set only once. A different sparse approximation, which also allows for an on-line processing, was recently introduced by Tresp (2000). It is based on combining predictions of models trained on smaller data subsets and needs an additional query set of inputs.

Central to our approach are exact expressions for the posterior means $\langle f_x \rangle_t$ and the posterior covariance $K_t(x, x')$ (subscripts denote the number of data points), which are derived in section 2. Although both quantities are continuous functions, they can be represented as finite linear (or respective bilinear) combinations of kernels $K_0(x, x_i)$ evaluated at the training inputs x_i (Csató, Fokoué, Opper, Schottky, & Winther, 2000). Using sequential projections of the posterior process on the manifold of gaussian processes, we obtain approximate recursions for the effective parameters of these representations. Since the size of representations grows with the number of training data, we use a second type of projection to extract a smaller subset of input data (reminiscent of the Support Vectors of Vapnik, 1995, or Relevance Vectors of Tipping, 2000). This subset builds up a sparse representation of the posterior process on which all predictions of the trained GP model rely. Our approach is related to that introduced in Wahba (1990). Although we use the same measure for projection, we do not fix the set of basis vectors from the beginning but decide on-line which inputs to keep.

2 On-Line Learning with Gaussian Processes

In Bayesian learning, all information about the parameters that we wish to infer is encoded in probability distributions (Bernardo & Smith, 1994). In the GP framework, the parameters are functions, and the GP priors specify a gaussian distribution over a function space. The posterior process is entirely specified by all its finite-dimensional marginals. Hence, let $\mathbf{f} = \{f(x_1), \ldots, f(x_M)\}$ be a set of function values such that $\mathbf{f}_{\mathcal{D}} \subseteq \mathbf{f}$, where $\mathbf{f}_{\mathcal{D}}$ is the set of $f(x_i) = f_i$ with x_i in the observed set of inputs, we compute the posterior distribution using the data likelihood together with the prior $p_0(\mathbf{f})$ as

$$p_{post}(\mathbf{f}) = \frac{P(\mathcal{D}|\mathbf{f})p_0(\mathbf{f})}{\langle P(\mathcal{D}|\mathbf{f}_{\mathcal{D}})\rangle_0}, \tag{2.1}$$

where $\langle P(\mathcal{D}|\mathbf{f}_{\mathcal{D}})\rangle_0$ is the average of the likelihood with respect to the prior GP (GP at time 0). This form of the posterior distribution can be used to express posterior expectations as typically high-dimensional integrals. For prediction, one is especially interested in expectations of functions of the process at inputs, which are not contained in the training set. At first glance, one might assume that every prediction on a novel input would require the computation of a new integral. Even if we had good methods for approximate integration, this would make predictions at new inputs a rather tedious task. Luckily, the following lemma shows that simple but important predictive quantities like the posterior mean and the posterior covariance of the process at arbitrary inputs can be expressed as a combination of a finite set of parameters that depend on the training data only. For arbitrary likelihoods we can show that:

Lemma 1 (parameterization). *The result of the Bayesian update, equation 2.1, using a GP prior with mean function $\langle f_x \rangle_0$ and kernel $K_0(x, x')$ and data $\mathcal{D} = \{(x_n, y_n) | \; n = 1, \ldots, N\}$, is a process with mean and kernel functions given by*

$$\begin{aligned} \langle f_x \rangle_{post} &= \langle f_x \rangle_0 + \sum_{i=1}^{N} K_0(x, x_i) q(i) \\ K_{post}(x, x') &= K_0(x, x') + \sum_{i,j=1}^{N} K_0(x, x_i) R(ij) K_0(x_j, x'). \end{aligned} \tag{2.2}$$

The parameters $q(i)$ and $R(ij)$ are given by

$$\begin{aligned} q(i) &= \frac{1}{Z} \int d\mathbf{f} p_0(\mathbf{f}) \; \frac{\partial P(\mathcal{D}|\mathbf{f})}{\partial f(x_i)} \quad \text{and} \\ R(ij) &= \frac{1}{Z} \int d\mathbf{f} p_0(\mathbf{f}) \; \frac{\partial^2 P(\mathcal{D}|\mathbf{f})}{\partial f(x_i) \partial f(x_j)} - q(i) q(j), \end{aligned} \tag{2.3}$$

where $\mathbf{f} = [f(x_1), \ldots, f(x_N)]^T$ and $Z = \int d\mathbf{f} p_0(\mathbf{f}) \; P(\mathcal{D}|\mathbf{f})$ is a normalizing constant.

The parameters $q(i)$ and $R(ij)$ have to be computed only once during the training of the model and are fixed when we make predictions. The parametric form of the posterior mean (assuming a zero mean for the prior) resembles the representations for the predictors in other kernel approaches (such as Support Vector machines) that are obtained by minimizing certain cost functions. While the latter representations are derived from the celebrated representer theorem of Kimeldorf and Wahba (1971), our result, equation 2.2, does, to our best knowledge, not follow from this but is derived from simple properties of gaussian distributions. To keep focused on the main flow, we defer the proof to appendix B.

Making an immediate use of this representation is usually not possible because the posterior process is in general not gaussian and the integrals cannot be computed exactly. Hence, we need approximations in order to keep the inference tractable (Csató et al., 2000). One popular method is to approximate the posterior by a gaussian process (Williams & Barber, 1998; Seeger, 2000). This may be formulated within a variational approach, where a certain dissimilarity measure between the true and the approximate distribution is minimized. The most popular choice is the Kullback-Leibler divergence between distributions, defined as

$$KL(p|q) = \int d\theta \; p(\theta) \ln \frac{p(\theta)}{q(\theta)}, \tag{2.4}$$

where θ denotes the set of arguments of the densities. If $\hat{p}$ denotes the approximating gaussian distribution, one usually tries to minimize $KL(\hat{p} \| p_{post})$

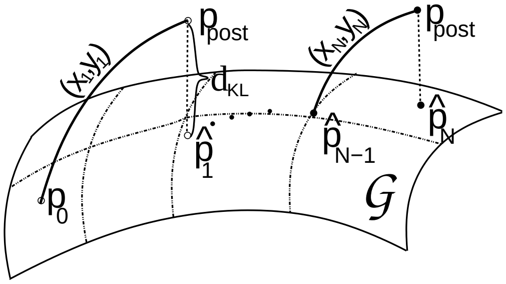

Figure 1: Visualization of the on-line approximation of the intractable posterior process. The resulting approximate process from previous iteration is used as the prior for the next one.

with respect to $\hat{p}$, which, in contrast to $KL(p_{post} \| \hat{p})$, requires only the computation of expectations over tractable distributions.

In this article, we use a different approach. To speed up the learning process in order to allow for the learning of large data sets, we aim at learning the data by a single sequential sweep through the examples. Let $\hat{p}_t$ denote the gaussian approximation after processing t examples; we use Bayes' rule,

$$p_{post}(\mathbf{f}) = \frac{P(y_{t+1}|\mathbf{f})\hat{p}_t(\mathbf{f})}{\langle P(y_{t+1}|\mathbf{f}_{\mathcal{D}})\rangle_t}, \tag{2.5}$$

to derive the updated posterior. Since p_{post} is no longer gaussian, we use a variational technique in order to project it to the closest gaussian process $\hat{p}_{t+1}$ (see Figure 1). Unlike the usual variational method, we now minimize the divergence $KL(p_{post} \| \hat{p})$. This is possible because in our on-line method, the posterior, equation 2.5, contains only the likelihood for a single example, and the corresponding nongaussian integral is one-dimensional, which can be performed analytically for many relevant cases. It is a simple exercise to show (Opper, 1998) that the projection results in the matching of the first two moments (mean and covariance) of p_{post} and the approximated gaussian posterior $\hat{p}_{t+1}$.

We expect that the use of the divergence $KL(p_{post} \| \hat{p})$ has several advantages over other variational methods (Gibbs & MacKay, 1999; Williams & Barber, 1998; Jaakkola & Haussler, 1999; Seeger, 2000). First, this choice avoids the numerical optimizations that are usually necessary for the divergence with inverted arguments. Second, this method is very robust, allowing for arbitrary choices of the single data likelihood. The likelihood can be noncontinuous and may even vanish over some range of values of the process. Finally, if one interprets the KL divergence as the expectation of the relative log loss of two distributions, our choice of divergence weights the losses with the correct distribution rather than the approximated one.

We expect that this may correspond to an improved quality of approximation.

In order to compute the on-line approximations of the mean and covariance kernel K_t, we apply lemma 1 sequentially with only one likelihood term $P(y_t|x_t)$ at an iteration step. Proceeding recursively, we arrive at

$$\begin{aligned}\langle f_x\rangle_{t+1} &= \langle f_x\rangle_t + q^{(t+1)}\, K_t(x, x_{t+1})\\ K_{t+1}(x, x') &= K_t(x, x') + r^{(t+1)}\, K_t(x, x_{t+1})K_t(x_{t+1}, x'),\end{aligned} \tag{2.6}$$

where the scalars $q^{(t+1)}$ and $r^{(t+1)}$ follow from lemma 1 (see appendix B for details):

$$\begin{aligned}q^{(t+1)} &= \frac{\partial}{\partial\langle f_{t+1}\rangle_t}\ln\langle P(y_{t+1}|f_{t+1})\rangle_t\\ r^{(t+1)} &= \frac{\partial^2}{\partial\langle f_{t+1}\rangle_t}\ln\langle P(y_{t+1}|f_{t+1})\rangle_t.\end{aligned} \tag{2.7}$$

The averages in equation 2.7 are with respect to the gaussian process at time t and the derivatives taken with respect to $\langle f_{t+1}\rangle_t = \langle f(x_{t+1})\rangle_t$. Note again that these averages require only a one-dimensional integration over the process at the input x_{t+1}. Unfolding the recursion steps in the update rules, equation 2.6, we arrive at the parameterization for the approximate posterior GP at time t as a function of the initial kernel and the likelihoods ("natural parameterization"):

$$\begin{aligned}\langle f_x\rangle_t &= \sum_{i=1}^{t} K_0(x, x_i)\alpha_t(i) = \boldsymbol{\alpha}_t^T\mathbf{k}_x\\ K_t(x, x') &= K_0(x, x') + \sum_{i,j=1}^{t} K_0(x, x_i)C_t(ij)K_0(x_j, x')\\ &= K_0(x, x') + \mathbf{k}_x^T\mathbf{C}_t\mathbf{k}_{x'},\end{aligned} \tag{2.8}$$

with coefficients $\alpha_t(i)$ and $C_t(ij)$ not depending on x and x' (for details, see appendix C). For simplicity, the values $\alpha_t(i)$ are grouped into the vector $\boldsymbol{\alpha}_t = [\alpha_t(1), \ldots, \alpha_t(t)]^T$, $\mathbf{C}_t = \{C_t(ij)\}_{i,j=1,t}$ and we also used vectorial (typeset in bold) notations for $\mathbf{k}_x = [K_0(x_1, x), \ldots, K_0(x_t, x)]^T$.

The recursion for the GP parameters in equation 2.8 is found from the recursion equation 2.6 and the parameterization lemma:

$$\begin{aligned}\boldsymbol{\alpha}_{t+1} &= T_{t+1}\left(\boldsymbol{\alpha}_t\right) + q^{(t+1)}\mathbf{s}_{t+1}\\ \mathbf{C}_{t+1} &= U_{t+1}\left(\mathbf{C}_t\right) + r^{(t+1)}\mathbf{s}_{t+1}\mathbf{s}_{t+1}^T\\ \mathbf{s}_{t+1} &= T_{t+1}\left(\mathbf{C}_t\mathbf{k}_{t+1}\right) + \mathbf{e}_{t+1},\end{aligned} \tag{2.9}$$

where $\mathbf{k}_{t+1} = \mathbf{k}_{x_{t+1}}$ and $\mathbf{e}_{t+1}$ the $t+1$th unit vector and $\mathbf{s}_{t+1}$ is introduced for clarity. We also introduced the operators T_{t+1} and U_{t+1}. They extend a t-dimensional vector and matrix to a $t+1$-dimensional one by appending zeros at the end of the vector and to the last row and column of the matrix, respectively.

Since $\mathbf{e}_{t+1}$ is the $t+1$th unit vector, we see that the dimension of the vector $\boldsymbol{\alpha}$ and the size of matrix $\mathbf{C}$ increase with each likelihood point added.

Equations 2.6 and 2.7 show some resemblance to the well-known extended Kalman filter. This is to be expected, because the latter approach can also be understood as a sequential propagation of an approximate gaussian distribution. However, the main difference between the two methods is in the way the likelihood model is incorporated. While the extended Kalman filter (see Bottou, 1998, for a general framework) is based on a linearization of the likelihood, our approach uses a more robust smoothing of the likelihood instead.

The drawback of using equation 2.9 in practice is the quadratic increase of the number of parameters with the number of training examples. This is a feature common to most other methods of inference with gaussian processes. A modification of the learning rule that controls the number of parameters is the main contribution of this article and is detailed in the following.

3 Sparseness in Gaussian Processes

Sparseness can be introduced within the GP framework by using suitable approximations on the representation equation 2.8. Our goal is to perform an update without increasing the number of parameters $\boldsymbol{\alpha}$ and $\mathbf{C}$ when, according to a certain criterion, the error due to the approximation is not too large. This could be achieved exactly if the new input x_{t+1} would be such that the relation

$$K_0(x, x_{t+1}) = \sum_{i=1}^{t} \hat{e}_{t+1}(i) K_0(x, x_i) \tag{3.1}$$

holds for all x. In such a case, we would have a representation for the updated process in the form equation 2.8 using only the first t inputs, but with "renormalized" parameters $\hat{\boldsymbol{\alpha}}$ and $\hat{\mathbf{C}}$. A glance at equation 2.9 shows that the only change would be the replacement of the vector $\mathbf{s}_{t+1}$ by

$$\hat{\mathbf{s}}_{t+1} = \mathbf{C}_t \mathbf{k}_{t+1} + \hat{\mathbf{e}}_{t+1}. \tag{3.2}$$

Note that $\hat{\mathbf{e}}_{t+1}$ is a vector of dimensionality t. Unfortunately, for most kernels and inputs x_{t+1}, equation 3.1 cannot be fulfilled for *all* x. Nevertheless, as an approximation, we could try an update of the form 3.2, where $\hat{\mathbf{e}}_{t+1}$ is determined by minimizing the error measure,

$$\left\| K_0(\cdot, x_{t+1}) - \sum_{i=1}^{t} \hat{e}_{t+1}(i) K_0(\cdot, x_i) \right\|^2, \tag{3.3}$$

where $\|\cdot\|$ is a suitably defined norm in a space of functions of inputs x (related optimization criteria in a function space are presented by Vijayakumar & Ogawa, 1999). Equation 3.3 becomes especially simple when the norm is based on the inner product of the reproducing kernel Hilbert space (RKHS) generated by the kernel K_0. In this case, for any two functions g and h that are represented as $g(x) = \sum_i c_i K_0(x, u_i)$ and $h(x) = \sum_i d_i K_0(x, v_i)$, for some arbitrary set of u_i's and v_i's, the RKHS inner product is defined as (Wahba, 1990)

$$\big(g(\cdot), h(\cdot)\big)_{RKHS} = \sum_{ij} c_i d_j K_0(u_i, v_i) \tag{3.4}$$

with norm

$$\|g\|^2_{RKHS} = \big(g(\cdot), g(\cdot)\big)_{RKHS} = \sum_{ij} c_i c_j K_0(u_i, u_j). \tag{3.5}$$

Hence, in this case, equation 3.3 is

$$K_0(x_{t+1}, x_{t+1}) + \sum_{i,j=1}^{t} \hat{e}_{t+1}(i)\hat{e}_{t+1}(j) K_0(x_i, x_j)$$
$$-2 \sum_{i=1}^{t} \hat{e}_{t+1}(i) K_0(x_{t+1}, x_i), \tag{3.6}$$

and simple minimization of equation 3.6 yields (Smola & Schölkopf, 2000)

$$\hat{\mathbf{e}}_{t+1} = \mathbf{K}_t^{(-1)} \mathbf{k}_{t+1}, \tag{3.7}$$

where $\mathbf{K}_t = \{K_0(x_i, x_j)\}_{i,j=1,t}$ is the Gram matrix. The expression

$$\widehat{K}_0(x, x_{t+1}) = \sum_{i=1}^{t} \hat{e}_{t+1}(i) K_0(x, x_i) \tag{3.8}$$

is simply the orthogonal projection (in the sense of the inner product, equation 3.4) of the function $K_0(x, x_{t+1})$ on the linear span of the functions $K_0(x, x_i)$. The approximate update using equation 3.2 will be performed

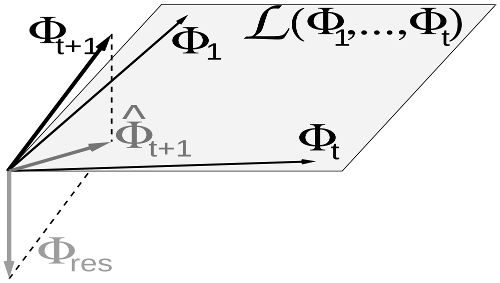

Figure 2: Visualization of the projection step. The new feature vector ϕ_{t+1} is projected to the subspace spanned by $\{\phi_1, \ldots, \phi_t\}$, resulting in the projection $\hat{\phi}_{t+1}$ and the orthogonal residual (the "left-out" quantity) ϕ_{res}. It is important that ϕ_{res} has $t+1$ components; it needs the extended basis including ϕ_{t+1}.

only when a certain measure for the approximation error (to be discussed later) is not exceeded. The set of inputs, for which the exact update is performed and the number of parameters is increased, will be called *basis vector set* ($\mathcal{BV}$ set), an element will be $\mathcal{BV}$. Proceeding sequentially, some of the inputs are left out, and others are included in $\mathcal{BV}$ set. However, due to the projection 3.8, the inputs left out from $\mathcal{BV}$ set will still contribute to the final GP configuration—the one used for prediction and to measure the posterior uncertainties. But the latter inputs will not be stored and do not lead to an increase of the size of the parameter set.

This procedure leads to the new representation for the posterior GP only in terms of the $\mathcal{BV}$ set and the corresponding parameters $\boldsymbol{\alpha}$ and $\mathbf{C}$:

$$\begin{aligned}\langle f_x \rangle &= \sum_{i\in\mathcal{BV}} K_0(x, x_i)\alpha(i)\\ K(x, x') &= K_0(x, x') + \sum_{i,j\in\mathcal{BV}} K_0(x, x_i)C(ij)K_0(x_j, x').\end{aligned} \tag{3.9}$$

An alternative derivation of these results can be obtained from the representation of the Mercer kernel (Wahba, 1990; Vapnik, 1995),

$$K_0(x, x') = \phi(x)^T \phi(x'), \tag{3.10}$$

in terms of the possibly infinite-dimensional "feature vector" $\phi(x)$ (Wahba, 1990). Minimizing equation 3.6 and using equation 3.2 for an update is equivalent to replacing the feature vector ϕ_{t+1} corresponding to the new input by its orthogonal projection,

$$\hat{\phi}_{t+1} = \sum_i \hat{e}_{t+1}(i)\phi_i, \tag{3.11}$$

onto the space of the old feature vectors (as in Figure 2). Note, however, that this derivation may be somewhat misleading by suggesting that the mapping induced by the feature vectors plays a special role for our approximation. This would be confusing because the representation equation, 3.10, is not unique. Our first derivation based on the RKHS norm shows, however, that our approximation uses only geometrical properties that are induced by the kernel K_0.

3.1 Projection-Induced Error. We need a rule to decide if the current input will be included in the $\mathcal{BV}$ set. We base the decision on a measure of change on the sample averaged posterior mean of the GP due to the sparse approximation.

Assuming a learning scenario where only the basis vectors are memorized, we measure the change of the posterior mean due to the approximation by

$$\Delta\langle f_x\rangle_{t+1} = \langle f_x\rangle_{t+1} - \langle f_x\rangle_{\widehat{t+1}},$$

where $\langle f_x\rangle_{\widehat{t+1}}$ is the posterior mean with respect to the approximated process. Summing up the absolute values of the changes for the elements in the $\mathcal{BV}$ set and the new input leads to

$$\begin{aligned}\varepsilon_{t+1} &= \sum_{i=1}^{t+1} |\Delta\langle f_i\rangle_{t+1}| = |q^{t+1}| \sum_{i=1}^{t+1} \left|K_0(x_i, x_{t+1}) - \widehat{K}_0(x_i, x_{t+1})\right| \\ &= |q^{(t+1)}| \left\|K_0(\cdot, x_{t+1}) - \widehat{K}_0(\cdot, x_{t+1})\right\|^2_{RKHS}, \qquad (3.12)\end{aligned}$$

where the second line follows from the orthogonal projection together with the definition of the inner product in the RKHS (see equation 3.5)

It is an important observation that, also due to orthogonal projection, the error is concentrated only on the last data point since $\widehat{K}_0(\cdot, x_{t+1}) = K_0(\cdot, x_{t+1})$ *at* the old data points x_i, $i = 1, \ldots, t$. Rewriting equation 3.12 using the coefficients for $\widehat{K}_0(\cdot, x_{t+1})$ from equation 3.7, the error is

$$\varepsilon_{t+1} = |q^{(t+1)}| \left(k^*_{t+1} - \mathbf{k}^T_{t+1}\mathbf{K}_t^{-1}\mathbf{k}_{t+1}\right) = |q^{(t+1)}|\gamma_{t+1}, \qquad (3.13)$$

where $k^*_{t+1} = K_0(x_{t+1}, x_{t+1})$ and $q^{(t+1)}$ is given from equation 2.7. The error measure ε_{t+1} is a product of two terms. If the new input would be included in $\mathcal{BV}$, the corresponding coefficient α_{t+1} in the posterior mean would be equal to $q^{(t+1)}$, which is the likelihood-dependent part. The second term,

$$\gamma_{t+1} = k^*_{t+1} - \mathbf{k}^T_{t+1}\mathbf{K}_t^{-1}\mathbf{k}_{t+1}, \qquad (3.14)$$

gives the geometrical part, which is the squared norm of the "residual vector" from the projection in the RKHS (shown in Figure 2), or, equivalently, the "novelty" of the current input. If we use the RBF kernels, then the error equation, 3.13, is similar to the one used in deciding if new centers have to be included in the resource-allocating network (McLachlan & Lowe, 1996; Platt, 1991).

To compute the geometrical component of the error ε_{t+1}, a matrix inversion is needed at each step. The costly matrix inversion can be avoided by keeping track of the inverse Gram matrix $\mathbf{Q}_t = \mathbf{K}_t^{-1}$. The updates for the matrix can also be expressed with the variables γ_{t+1} and $\hat{\mathbf{e}}_{t+1}$ (for details, see appendix D), and these updates will be important when deleting a $\mathcal{BV}$:

$$\mathbf{Q}_{t+1} = U_{t+1}(\mathbf{Q}_t) + \gamma_{t+1}^{-1}(T_{t+1}(\hat{\mathbf{e}}_{t+1}) - \mathbf{e}_{t+1})(T_{t+1}(\hat{\mathbf{e}}_{t+1}) - \mathbf{e}_{t+1})^T, \quad (3.15)$$

where U_{t+1} and T_{t+1} are the extension operators for a matrix and a vector, respectively (introduced in equation 2.9).

3.2 Deleting a Basis Vector. Our algorithm may run into problems when there is no possibility of including new inputs into $\mathcal{BV}$ without deleting one of the old basis vectors because we are at the limit of our resources. This gives the motivation to implement pruning: whenever a new example is found important, one should get rid of the $\mathcal{BV}$ with the smallest error and replace it by the new input vector. First, we discuss the elimination of a $\mathcal{BV}$ and then the criterion based on which we choose the $\mathcal{BV}$ to be removed.

To remove a basis vector from the $\mathcal{BV}$ set, we first assume that the respective $\mathcal{BV}$ has just been added; thus, the previous update step was done with $\mathbf{e}_{t+1}$, the $t + 1$th unit vector. With this assumption, we identify the elements $q^{(t+1)}$, $r^{(t+1)}$, and $\mathbf{s}_{t+1}$ from equation 2.9; compute $\hat{\mathbf{e}}_{t+1}$ (this computation is also replaced by an identification from equation 3.15); and use equation 3.11 for an update without including the new point in the $\mathcal{BV}$ set.

If we assume $t+1$ basis vectors, $\boldsymbol{\alpha}_{t+1}$ has $t+1$ elements, and the matrices $\mathbf{C}_{t+1}$ and $\mathbf{Q}_{t+1}$ are $(t+1) \times (t+1)$. Further assuming that we want to delete the last added element, the decomposition is as illustrated in Figure 3. Computing the "previous" model parameters and then using the nonincreasing update leads to the deletion equations (see appendix E for details):

$$\begin{aligned}
\hat{\boldsymbol{\alpha}} &= \boldsymbol{\alpha}^{(t)} - \alpha^* \frac{\mathbf{Q}^*}{q^*} \\
\hat{\mathbf{C}} &= \mathbf{C}^{(t)} + c^* \frac{\mathbf{Q}^*\mathbf{Q}^{*T}}{q^{*2}} - \frac{1}{q^*}\left[\mathbf{Q}^*\mathbf{C}^{*T} + \mathbf{C}^*\mathbf{Q}^{*T}\right] \qquad (3.16) \\
\hat{\mathbf{Q}} &= \mathbf{Q}^{(t)} - \frac{\mathbf{Q}^*\mathbf{Q}^{*T}}{q*},
\end{aligned}$$

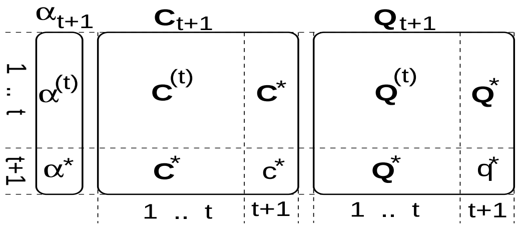

Figure 3: Grouping of the GP parameters for the update equation, 3.16.

where $\hat{\alpha}$, $\hat{\mathbf{C}}$, and $\hat{\mathbf{Q}}$ are the parameters after the deletion of the last basis vector and $\mathbf{C}^{(t)}, \mathbf{Q}^{(t)}, \boldsymbol{\alpha}^{(t)}, \mathbf{Q}^*, \mathbf{C}^*, q^*$, and c^* are taken from GP parameters before deletion. A graphical illustration of each element is provided in Figure 3.

Of particular interest is the identification of the parameters $q^{(t+1)}$ and γ_{t+1} since their product gives the score of the basis vector that is being deleted. This leads to the score

$$\varepsilon_{t+1} = \frac{\alpha^*}{q^*} = \frac{\alpha_{t+1}(t+1)}{Q_{t+1}(t+1, t+1)}. \tag{3.17}$$

Thus, we have the score for the last input point. Neglecting dependence of the GP posterior on the ordering of the data, equation 3.17 gives us a score measure for each element i in the $\mathcal{BV}$ set,

$$\varepsilon_i = \frac{|\alpha_{t+1}(i)|}{Q_{t+1}(i, i)}, \tag{3.18}$$

by rearranging the order in $\mathcal{BV}$ with element i at the last position. To summarize, if a deletion is needed, then the basis vector with minimal score (from equation 3.18) will be deleted. The scores are computationally cheap (linear).

3.3 The Sparse GP Algorithm. The following numerical experiments are based on the version of the algorithm that assumes a given maximal size for the $\mathcal{BV}$ set.

We start by initializing the $\mathcal{BV}$ set with an empty set, the maximal number of elements in the $\mathcal{BV}$ set with d, the prior kernel K_0, and a tolerance parameter ϵ_{tol}. This latter will be used to prevent the Gram matrix from being singular and is used in step 2. The GP parameters $\boldsymbol{\alpha}$, $\mathbf{C}$, and the inverse Gram matrix $\mathbf{Q}$ are set to empty values.

For each data element (y_{t+1}, x_{t+1}), we will iterate the following steps:

1. Compute $q^{(t+1)}$, $r^{(t+1)}$, k^*_{t+1}, $\mathbf{k}_{t+1}$, $\hat{\mathbf{e}}_{t+1}$, and γ_{t+1}.
2. If $\gamma_{t+1} < \epsilon_{tol}$, then perform a reduced update with $\hat{\mathbf{s}}_{t+1}$ from equation 3.2 without extending the size of the parameters $\boldsymbol{\alpha}$ and $\mathbf{C}$. Advance to the next data.

3. (else) Perform the update equation 2.9 using the unit vector $\mathbf{e}_{t+1}$. Add the current input to the $\mathcal{BV}$ set, and compute the inverse of the extended Gram matrix using equation 3.15.
4. If the size of the $\mathcal{BV}$ set is larger than d, then compute the scores ε_i for all $\mathcal{BV}$s from equation 3.18, find the basis vector with the minimum score, and delete it using equations 3.16.

The computational time for a single step is quadratic in d, the maximal number of $\mathcal{BV}$s allowed. Having an iteration over all data, the computational time is $\mathcal{O}(Nd^2)$. This is a significant improvement from the $\mathcal{O}(N^3)$ scaling of the nonsparse GP algorithm. Since we propose a "subspace" algorithm, we have the same computing time as in Wahba (1990) and the Nyström approximation for kernel matrices in Williams and Seeger (2001).

An important feature of the sparse GP algorithm is that we provide an approximation to the posterior kernel of the process providing predictive variance for a new inputs, as shown by the error bars in Figure 4. Another aspect of the algorithm is that the basis vectors are selected during runtime from the data. A final remark is that the iterative computation of the inverse Gram matrix $\mathbf{Q}$ prevents the numerical instabilities caused by inverting a singular Gram matrix: γ_{t+1} is zero if the new element x_{t+1} would make the Gram matrix singular. The comparison of γ_{t+1} with the preset tolerance value ϵ_{td} prevents this.

4 Experimental Results

In all experiments, we used spherical RBF kernels

$$K(x, x') = \exp\left(\frac{\|x - x'\|^2}{2d\sigma_K^2}\right), \tag{4.1}$$

where σ_K is the width of the kernel and d is the input dimension.

4.1 Regression. In the regression model, we assume a multidimensional input $x \in \mathbb{R}^m$ and an output y with the likelihood

$$P(y|x) = \frac{1}{\sqrt{2\pi}\sigma_0} \exp\left\{-\frac{\|y - f_x\|^2}{2\sigma_0^2}\right\}. \tag{4.2}$$

Since the likelihood is gaussian, the use of a gaussian posterior in the on-line algorithm is exact. Hence, only the sparsity will introduce an approximation into our procedure. For a given number of examples, the parameterization 2.8 of the posterior in terms of $\boldsymbol{\alpha}$ and $\mathbf{C}$ leads to a predictive distribution of

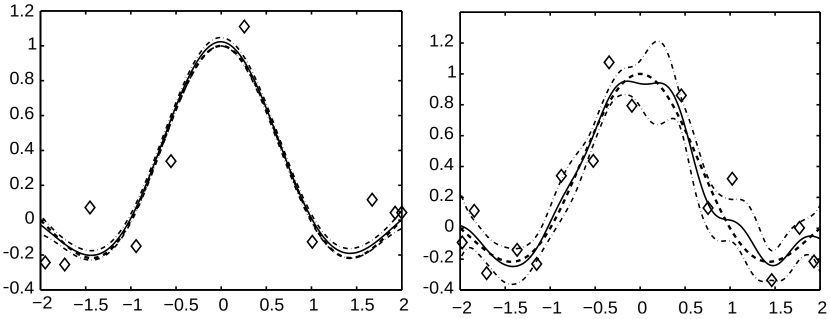

Figure 4: Results of the GP regression with 1000 noisy training data points with noise $\sigma_0^2 = 0.02$. The figures show the results for RBF kernels with two different widths. (Left) The good fit of the GP mean function (continuous lines) to the true function (dashed lines) is also consistent with the tight Bayesian error bars (dash-dotted lines) around the means. (Right) The error bars are broader, reflecting the larger uncertainty. The $\mathcal{BV}$ set is marked with rhombs and we kept 10 and 15 basis vectors. We used $\sigma_K^2 = 1$ for the left, and $\sigma_K^2 = 0.1$ for the right subfigure, respectively.

y for an input x,

$$p(y|x, \boldsymbol{\alpha}, \mathbf{C}) = \left(\frac{1}{2\pi\sigma_x^2}\right)^2 \exp\left\{-\frac{\|y - \boldsymbol{\alpha}^T\mathbf{k}_x\|^2}{2\sigma_x^2}\right\}, \tag{4.3}$$

with $\sigma_x^2 = \sigma_0^2 + \mathbf{k}_x^T C_t \mathbf{k}_x + k_x^*$. The on-line update rules, equation 2.9, for $\boldsymbol{\alpha}$ and $\mathbf{C}$ in terms of the parameters $q^{(t+1)}$ and $r^{(t+1)}$ are

$$q^{(t+1)} = (y - \boldsymbol{\alpha}_t^T\mathbf{k}_x)/\sigma_x^2 \qquad r^{(t+1)} = -\frac{1}{\sigma_x^2}. \tag{4.4}$$

To illustrate the performance, we begin with the toy example $y = sin(x)/x + \eta$, where η is a zero-mean gaussian noise with variance $\sigma_0^2 = 0.02$. The results for the posterior means and the Bayesian error bars together with the basis vectors are shown in Figure 4. The large error bars obtained for the "misspecified" kernel (with small width $\sigma_K^2 = 0.1$) demonstrate the advantage of propagating not only the means but also the uncertainties in the algorithm.

The second data set is the Friedman data set 1 (Friedman, 1991), an artificial data set frequently used to assess the performance of regression algorithms. For this example, we demonstrate the effect of the approximation introduced by the sparseness. The upper solid line in Figure 5 shows the development of the test error with increasing numbers of examples without sparseness, that is, when all data are included sequentially. The dots are the

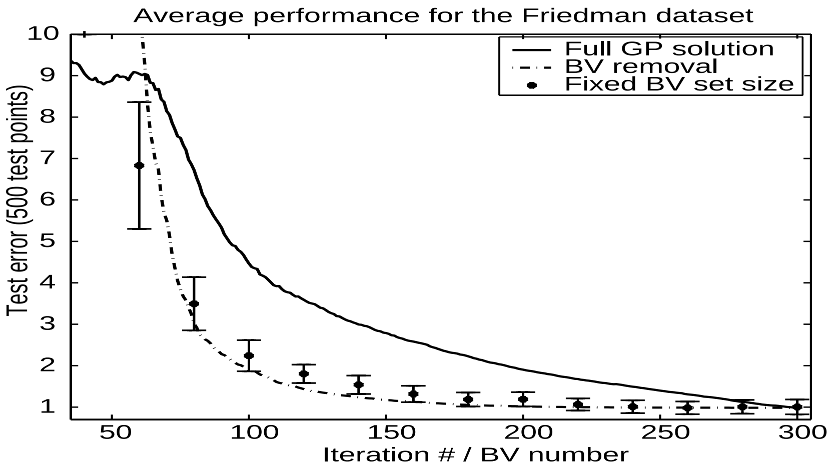

Figure 5: Results for the Friedman data using the full GP regression (continuous line) and the proposed sparse GP algorithm with a fixed $\mathcal{BV}$ size (dots with error bars). The dash-dotted line is obtained by sequentially reducing the size of the $\mathcal{BV}$ set. The lines show the average performance over 50 runs. The full GP solution uses only the specified number of data, whereas the other two curves are obtained by iterating over the full data set ($\sigma_K^2 = 1$ was used with 300 training and 500 test data).

test errors obtained by running the sparse algorithm using different sizes of the $\mathcal{BV}$ set. We see that almost two-thirds of the original training set can be excluded from the $\mathcal{BV}$ set without a significant loss of predictive performance. Finally, we have tested the effect of the greediness of the algorithm by adding or removing examples in different ways. The dependence on the data of the sparse GP is shown with the error bars around the dots, and the dependence of the result on the different orders is well within these error bars. The dash-dotted line is obtained by first running the on-line algorithm without sparseness on the full data set and then building sets $\mathcal{BV}$ of decreasing sizes by removing the least significant examples one after the other. Remarkably, the performance is rather stable against these variations in the plateau region of (almost) constant test error.

4.2 Classification. For classification, we use the probit model (Neal, 1997) where a binary value $y \in \{-1, 1\}$ is assigned to an input $x \in \mathbb{R}^m$ with the data likelihood

$$P(y|f_x) = \mathrm{Erf}\left(\frac{y\, f_x}{\sigma_0}\right). \tag{4.5}$$

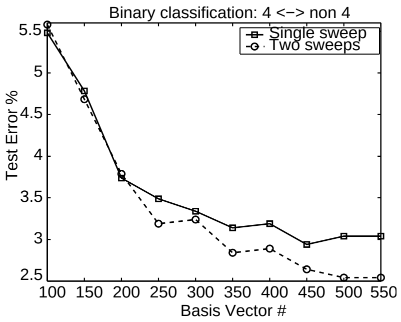

Figure 6: Results for the binary (left) and multiclass (right) classification. The multiclass case is a combination of the 10 individual classifiers. The example x is assigned to the class with highest $P(C_i|x)$. We compare different sizes of the $\mathcal{BV}$ set and the effect of reusing data a second time (signified by the circles).

Erf(x) is the cumulative gaussian distribution,[1] with σ_0 the noise variance. The predictive distribution for a new example x is

$$p(y|x, \boldsymbol{\alpha}, \mathbf{C}) = \langle P(y|f_x)\rangle_t = \text{Erf}\left(\frac{y\ \langle f_x\rangle}{\sigma_x}\right), \tag{4.6}$$

where $\langle f_x\rangle$ the mean of the GP at x given by equation 2.8 and $\sigma_x^2 = \sigma_0^2 + k_x^* + \mathbf{k}_x^T C \mathbf{k}_x$. Based on equation 2.9, for a given input-output pair (x, y), the update coefficients $q^{(t+1)}$ and $r^{(t+1)}$ are computed (for details, see Csató et al., 2000):

$$q^{(t+1)} = \frac{y}{\sigma_x}\frac{\text{Erf}'}{\text{Erf}} \qquad r^{(t+1)} = \frac{1}{\sigma_x^2}\left\{\frac{\text{Erf}''}{\text{Erf}} - \left(\frac{\text{Erf}'}{\text{Erf}}\right)^2\right\}, \tag{4.7}$$

with Erf(z) evaluated at $z = \frac{y\ \boldsymbol{\alpha}_t^T \mathbf{k}_x}{\sigma_x}$ and Erf$'$ and Erf$''$ are the first and second derivatives at z.

We have tested the sparse GP algorithm on the U. S. Postal Service (USPS) data set[2] of gray-scale handwritten digit images (of size 16 × 16) with 7291 training patterns and 2007 test patterns. In the first experiment, we studied the problem of classifying the digit 4 against all other digits. Figure 6 (left) plots the test errors of the algorithm for different $\mathcal{BV}$ set sizes and fixed values of hyperparameter $\sigma_K^2 = 1$.

[1] $\text{Erf}(x) = \int_{-\infty}^{x} dt \exp(-t^2/2)/\sqrt{2\pi}$.

[2] Available from `http://www.kernel-machines.org/data/`.

The USPS data set has been used previously to test the performance of other kernel-based classification algorithms that are based on a sparse representations. We mention the kernel PCA method of (Schölkopf et al., 1999) or the Nyström method of Williams and Seeger (2001). They obtained slightly better results than our on-line algorithm. When the basis of the Nyström approach is reduced to 512, the mean error is $\approx 1.7\%$ (Williams & Seeger, 2001) and the PCA reduced-set method of Schölkopf et al. (1999) leads to an error rate of $\approx 5\%$. This may be due to the fact that the sequential replacement of the posterior by a gaussian is an approximation for the classification problem. Hence, some of the information contained in an example is lost even when the $\mathcal{BV}$ set would contain all data. As shown in in Figure 6, we observe a slight improvement when the algorithm sweeps several times through the data. However, it should be noted that the use of the algorithm (in its present form) on data that it has already seen is a mere heuristic and can no longer be justified from a probabilistic point of view. A change of the update rule based on a recycling of examples will be investigated elsewhere.

We have also tested our method on the more realistic problem of classifying all 10 digits simultaneously. Our ability to compute Bayesian predictive probabilities is absolutely essential in this case. We have trained 10 classifiers on the 10 binary classification problems of separating a single digit from the rest. A new input was assigned to the class with the highest predictive probability given by equation 4.5. Figure 6 summarizes the results for the multiclass case for different $\mathcal{BV}$ set sizes and gaussian kernels (with the external noise variance $\sigma_0^2 = 0$). In this case, the recycling of examples was of less significance. The gap between our on-line result and the batch performance reported in Schölkopf et al. (1999) is also smaller; this might be due to the Bayesian nature of the GPs that avoids the overfitting.

To reduce the computational cost, we used the same set for all individual classifiers (only a single inverse of the Gram matrix was needed, and also the storage cost is smaller). This made the implementation of deleting a basis vector for the multiclass case less straightforward. For each input and each basis vector, there are 10 individual scores. We implemented a minimax deletion rule: whenever a deletion was needed, the basis vector having the smallest maximum value among the 10 classifier problems was deleted; that is, the index l of the deleted input was

$$l = \arg\min_{i \in \mathcal{BV}} \max_{c \in \overline{0,9}} \varepsilon_i^c. \tag{4.8}$$

Figure 7 shows the evolution of the test error when the sparse GP algorithm was initially trained with 1500 $\mathcal{BV}$s and (without any retraining) the "least scoring" basis vectors are deleted. As in the regression case (see Figure 5), we observe a long plateau of almost constant test error when up to 70% of the $\mathcal{BV}$s are removed.

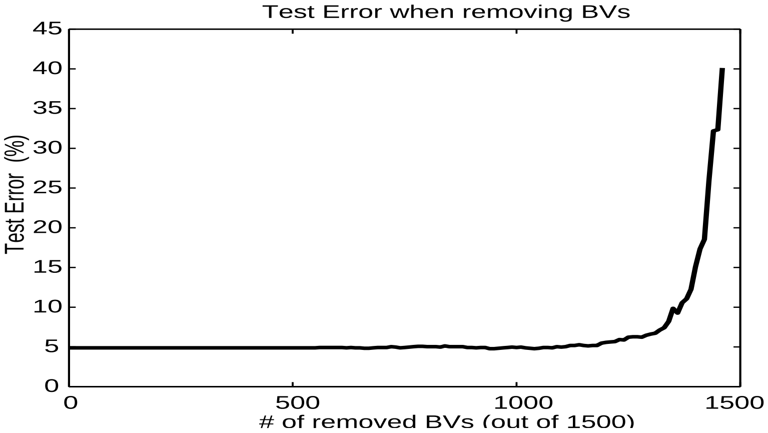

Figure 7: Performance of the combined classifier trained with an initial $\mathcal{BV}$ size of 1500 and a sequential removal of basis vectors.

5 Conclusion and Further Investigations

We have presented a greedy algorithm that allows computing a sparse gaussian approximation for the posterior of GP models with general (factorizing) likelihoods based on a single sweep through the data set. So far, we have applied the method to regression and classification tasks and obtained a performance close to batch methods.

The strength of the method lies in the fact that arbitrary, even noncontinuous likelihoods, which may be zero in certain regions, can be treated by our method. Such likelihoods may cause problems for other approximations based on local linearizations (advanced Kalman filters) or on the averaging of the log-likelihood (variational gaussian approximation). Our method merely requires the explicit computation of a gaussian smoothed likelihood and is thus well suited for cases where (local) likelihood functions can be modeled empirically as mixtures of gaussians. If such expressions are available, the necessary one-dimensional integrals can be done analytically, and the on-line updates require just matrix multiplications and function evaluations. A model of this structure for which we already have obtained promising preliminary results is the one used to predict wind fields from ambiguous satellite measurements based on a GP prior for the wind fields and a likelihood model for the measurement process.

A further development of the method requires the solution of various theoretical problems at which we are now working. An important problem is to assess the quality of our approximations. There are two sources of errors: one coming from the gaussian on-line approximation and another stemming from the additional sparsity. In both cases, it is easy to obtain explicit expressions for single-step errors, but it is not obvious how to combine these in order to estimate the cumulative deviation between the true posterior and our approximation. It may be interesting to concentrate on the regression problem first because in this case, there is no approximation involved in computing the posterior.

A different question is the (frequentist) statistical quality of the algorithm. Our on-line gaussian approximation (without sparseness) was found to be asymptotically efficient (in the sense of Fisher) in the finite-dimensional (parametric) case (Opper, 1996, 1998). This result does not trivially extend to the present infinite-dimensional GP case, and further studies are necessary. These may be based on the idea of an effective, finite dimensionality for the set of well-estimated parameters (Trecate, Williams, & Opper, 1999). Such work should also give an estimate for the sufficient number of basis vectors and explain the existence of the long plateaus (see Figures 7 and 5) with practically constant test errors.

Besides a deeper understanding of the algorithm, we find it also important to improve our method. Our sparse approximation was found to preserve the posterior means on previous data points when projecting on a representation that leaves out the current example. A further improvement might be achieved if information on the posterior variance would also be used (e.g., by taking the Kullback-Leibler loss rather than the RKHS norm) in optimizing the projection. This may, however, result in more complex and time-consuming updates.

Our experiments show that in some cases, the performance of the on-line algorithm is inferior to a batch method. We expect that our algorithm can be adapted to a recycling of data (e.g., along the lines of Minka, 2000) such that a convergence to a sparse representation of the TAP mean-field method (Opper & Winther, 1999) is achieved.

A further drawback that will be addressed in future work is the lack of an (on-line) adaptation of the kernel hyperparameters. Rather than setting them by hand, an approximate propagation of posterior distributions for the hyperparameters would be desirable.

Finally, there may be cases of probabilistic models where the restriction to unimodal posteriors as given by the gaussian approximation is too severe. Although we know that for gaussian regression and classification with logistic or probit functions the posterior is unimodal, for more complicated models, an on-line propagation of a mixture of GPs should be considered.

Acknowledgments

We thank Bernhard Schottky for initiating the gaussian process parameterization lemma. The work was supported by EPSRC grant no. GR/M81608.

Appendix A: Properties of Zero-Mean Gaussians

The following property of the gaussian probability density function (pdf) is often used in this article; here we state it in a form of a theorem:

Theorem 1. *Let $x \in \mathbb{R}^m$ and $p(x)$ zero-mean gaussian pdf with covariance $\boldsymbol{\Sigma} = \{\Sigma_{ij}\}$ (i, j from 1 to m). If $g\colon \mathbb{R}^m \to \mathbb{R}$ is a differentiable function not growing faster than a polynomial and with partial derivatives,*

$$\partial_j g(x) = \frac{\partial}{\partial x_j} g(x) \ ,$$

then

$$\int_{\mathbb{R}^m} dx p(x)\ x_i g(x) = \sum_{j=1}^{m} \Sigma_{ij} \int_{\mathbb{R}^m} dx p(x)\ \partial_j g(x). \tag{A.1}$$

In the following we will assume definite integration over $\mathbb{R}^m$ whenever the integral appears. Alternatively, using the vector notation, the above identity reads:

$$\int dx p(x)\ x g(x) = \boldsymbol{\Sigma} \int dx p(x)\ \nabla g(x). \tag{A.2}$$

For a general gaussian pdf with mean $\boldsymbol{\mu}$, above A.2 transforms to

$$\int dx p(x)\ x g(x) = \boldsymbol{\mu} \int dx p(x)\ g(x) + \boldsymbol{\Sigma} \int dx p(x)\ \nabla g(x). \tag{A.3}$$

Proof. The proof uses the partial integration rule:

$$\int dx p(x) \nabla g(x) = - \int dx g(x) \nabla p(x),$$

where we have used the fast decay of the gaussian function to dismiss one of the terms. Using the derivative of a gaussian pdf, we have

$$\int dx p(x) \nabla g(x) = \int dx\ g(x) \Sigma^{-1} x p(x).$$

Multiplying both sides with $\boldsymbol{\Sigma}$ leads to equation A.2, completing the proof. For the nonzero mean, the deductions are also straightforward.

Appendix B: Proof of the Parameterization Lemma

Using Bayes' rule, the posterior process has the form

$$\hat{p}(\mathbf{f}) = \frac{p_0(\mathbf{f})\ P(\mathcal{D}|\mathbf{f})}{\int d\mathbf{f} p_0(\mathbf{f})\ P(\mathcal{D}|\mathbf{f})},$$

where $\mathbf{f}$ is a set of realizations for the random process indexed by arbitrary points from $\mathbb{R}^m$, the inputs for the GPs.

We compute first the mean function of the posterior process:

$$\begin{aligned}\langle f_x\rangle_{post} &= \int d\mathbf{f}\hat{p}(\mathbf{f})\ f_x = \frac{\int d\mathbf{f} p_0(\mathbf{f})\ f_x\ P(\mathcal{D}|\mathbf{f})}{\int d\mathbf{f} p_0(\mathbf{f})\ P(\mathcal{D}|\mathbf{f})}\\ &= \frac{1}{Z}\int df_x \prod_{i=1}^{N} df_i\ p_0(f_x, f_1, \ldots, f_N)\ f_x\ P(\mathcal{D}|f_1, \ldots, f_N),\end{aligned} \tag{B.1}$$

where the denominator was denoted by Z and we used index notation for the realizations of the process also (thus, $f(x) = f_x$ and $f(x_i) = f_i$). Observe that regardless of the number of the random variables of the process considered, the dimension of the integral we need to consider is only $N + 1$; all other random variables will integrate out (as in equation B.1). We thus have an $N + 1$-dimensional integral in the numerator, and Z is an N-dimensional integral. If we group the variables related to the data as $\mathbf{f}_{\mathcal{D}} = [f_1, \ldots, f_N]^T$ and apply theorem 1 (see equation A.1), replacing x_i by f_x and $g(x)$ by $P(\mathcal{D}|\mathbf{f}_{\mathcal{D}})$, we have

$$\begin{aligned}\langle f_x\rangle_{post} = \frac{1}{Z}\Bigg(&\langle f_x\rangle_0 \int df_x d\mathbf{f}_{\mathcal{D}} p_0(f_x, \mathbf{f}_{\mathcal{D}})\ P(\mathcal{D}|\mathbf{f}_{\mathcal{D}})\\ &+ \sum_{i=1}^{N} K_0(x, x_i) \int df_x d\mathbf{f}_{\mathcal{D}} p_0(f_x, \mathbf{f}_{\mathcal{D}})\ \partial_i P(\mathcal{D}|\mathbf{f}_{\mathcal{D}})\Bigg),\end{aligned} \tag{B.2}$$

where K_0 is the kernel function generating the covariance matrix ($\mathbf{\Sigma}$ in theorem 1). The variable f_x in the integrals disappears since it is contained only in p_0. Substituting back Z leads to

$$\langle f_x\rangle_{post} = \langle f_x\rangle_0 + \sum_{i=1}^{N} K_0(x, x_i) q_i, \tag{B.3}$$

where q_i is read off from equation B.2,

$$q_i = \frac{\int d\mathbf{f}_{\mathcal{D}} p_0(\mathbf{f}_{\mathcal{D}})\ \partial_i P(\mathcal{D}|\mathbf{f}_{\mathcal{D}})}{\int d\mathbf{f}_{\mathcal{D}} p_0(\mathbf{f}_{\mathcal{D}})\ P(\mathcal{D}|\mathbf{f}_{\mathcal{D}})}, \tag{B.4}$$

and the coefficients q_i depend only on the data and are independent from x, at which the posterior mean is evaluated.

We can simplify the expression for q_i by performing a change of variables in the numerator, $f_i' = f_i - \langle f_i \rangle_0$, where $\langle f_i \rangle_0$ is the prior mean at x_i, and keeping all other variables unchanged $f_j' = f_j, j \neq i$, leading to the numerator

$$\int d\mathbf{f}_{\mathcal{D}} p_0(\mathbf{f}'_{\mathcal{D}}) \partial_i P(\mathcal{D}|f_1', \ldots, f_i' + \langle f_i \rangle_0, \ldots, f_N'),$$

and the differentiation is with respect to f_i'. We then change the partial differentiation with respect to f_i' with the partial differentiation with respect to $\langle f_i \rangle_0$ and exchange the differentiation and integral operators (they apply to a distinct set of variables), leading to

$$\frac{\partial}{\partial \langle f_i \rangle_0} \int d\mathbf{f}'_{\mathcal{D}} p_0(\mathbf{f}'_{\mathcal{D}}) P(\mathcal{D}|f_1', \ldots, f_i' + \langle f_i \rangle_0, \ldots, f_N').$$

We then perform the inverse change of variables inside the integral and substitute back into the expression for q_i:

$$\begin{aligned} q_i &= \frac{\frac{\partial}{\partial \langle f_i \rangle_0} \int d\mathbf{f}_{\mathcal{D}} p_0(\mathbf{f}_{\mathcal{D}}) P(\mathcal{D}|\mathbf{f}_{\mathcal{D}})}{\int d\mathbf{f}_{\mathcal{D}} p_0(\mathbf{f}_{\mathcal{D}}) P(\mathcal{D}|\mathbf{f}_{\mathcal{D}})} \\ &= \frac{\partial}{\partial \langle f_i \rangle_0} \ln \int d\mathbf{f}_{\mathcal{D}} p_0(\mathbf{f}_{\mathcal{D}}) P(\mathcal{D}|\mathbf{f}_{\mathcal{D}}). \end{aligned} \tag{B.5}$$

Writing the expression for the posterior kernel,

$$K_{post}(x, x') = \langle f_x f_{x'} \rangle_{post} - \langle f_x \rangle_{post} \langle f_{x'} \rangle_{post}, \tag{B.6}$$

and applying theorem 1 twice leads to

$$K_{post}(x, x') = K_0(x, x') + \sum_{i=1}^{N} \sum_{j=1}^{N} K_0(x, x_i) \left(D_{ij} - q_i q_j \right) K_0(x_j, x'), \tag{B.7}$$

where D_{ij} is

$$D_{ij} = \frac{1}{Z} \int d\mathbf{f}_{\mathcal{D}} p_0(\mathbf{f}_{\mathcal{D}}) \frac{\partial^2}{\partial f_j \partial f_i} P(\mathcal{D}|f_{\mathcal{D}}). \tag{B.8}$$

Identifying $R_{ij} = D_{ij} - q_i q_j$ leads to the required parameterization in equation 2.3 from lemma 1. Simplification of $R_{ij} = D_{ij} - q_i q_j$ is made by changing the arguments of the partial derivative and using the logarithm of the expectation (repeating steps B.4 and B.5 from q_i), leading to

$$R_{ij} = \frac{\partial^2}{\partial \langle f_i \rangle_0 \partial \langle f_j \rangle_0} \ln \int d\mathbf{f}_{\mathcal{D}} p_0(\mathbf{f}_{\mathcal{D}}) P(\mathcal{D}|\mathbf{f}_{\mathcal{D}}), \tag{B.9}$$

and using a single datum in the likelihood leads to the scalar coefficients $q^{(t+1)}$ and $r^{(t+1)}$ from equation 2.7.

Appendix C: On-Line Learning in GP Framework

We prove equation 2.8 by induction. We will show that for every time step, we can express the mean and kernel functions with coefficients $\boldsymbol{\alpha}$ and $\mathbf{C}$ given by the recursion (also equation 2.9):

$$\begin{aligned}\boldsymbol{\alpha}_{t+1} &= T_{t+1}\left(\boldsymbol{\alpha}_t\right) + q^{(t+1)}\mathbf{s}_{t+1} && \text{(C.1)}\\ \mathbf{C}_{t+1} &= U_{t+1}\left(\mathbf{C}_t\right) + r^{(t+1)}\mathbf{s}_{t+1}\mathbf{s}_{t+1}^T && \text{(C.2)}\\ \mathbf{s}_{t+1} &= T_{t+1}\left(\mathbf{C}_t\mathbf{k}_{t+1}\right) + \mathbf{e}_{t+1},\end{aligned}$$

where $\boldsymbol{\alpha}$ and $\mathbf{C}$ depend only on the data points x_i and kernel function K_0 but do not depend on the values x and x' (from equation 2.8) at which the mean and kernel functions are computed.

Proceeding by induction and using the induction hypothesis $\boldsymbol{\alpha}_0 = \mathbf{C}_0 = 0$ for time $t = 1$, we have $\alpha_1(1) = q^{(1)}$ and $C_1(1,1) = r^{(1)}$. The mean function at time $t = 1$ is $\langle f_x\rangle = \alpha_1(1)K_0(x_1, x)$ (from lemma 1 for a single datum; see equation 2.6). Similarly, the modified kernel is $K_1(x, x') = K_0(x, x') + K(x, x_1)C_1(1,1)K_0(x_1, x')$ with $\boldsymbol{\alpha}$ and $\mathbf{C}$ independent of x and x', proving the induction hypothesis.

We assume that at time moment t, we have the parameters $\boldsymbol{\alpha}_t$ and $\mathbf{C}_t$ independent of the points x and x'. These parameters specify a prior GP for which we apply the on-line learning:

$$\begin{aligned}\langle f_x\rangle_{t+1} &= \sum_{i=1}^{t} K_0(x_i, x)\alpha_t(i) + q^{(t+1)}\sum_{i,j=1}^{t} K_0(x, x_i)C_t(i,j)K_0(x_j, x_{t+1})\\ &\quad + q^{(t+1)}K_0(x, x_{t+1}) && \text{(C.3)}\\ &= \sum_{i=1}^{t+1} K_0(x, x_i)\alpha_{t+1}(i),\end{aligned}$$

and by pairwise identification we have equation C.1 or C.2 from the main body. The parameters $\boldsymbol{\alpha}_{t+1}$ do not depend on the particular value of x. Writing down the update equation for the kernels,

$$K_{t+1}(x, x') = K_t(x, x') + r^{(t+1)}K_t(x, x_{t+1})K_t(x_{t+1}, x'),$$

leads to equation C.2 in a straightforward manner with $C_{t+1}(i, j)$ independent of x and x', completing the induction.

Appendix D: Iterative Computation of the Inverse Gram Matrix

In the sparse approximation equation 3.7, we need the inverse Gram matrix of the $\mathcal{BV}$ set: $\mathbf{K}_{\mathcal{BV}} = \{K_0(x_i, x_j)\}$ is needed. In the following, the elements of the $\mathcal{BV}$ set are indexed from 1 to t. Using the matrix inversion formula,[3] the addition of a new element can be carried out sequentially. This is a well-known fact, exploited also in the Kalman filter algorithm. We consider the new element at the end (last row and column) of matrix $\mathbf{K}_{t+1}$. Matrix $\mathbf{K}_{t+1}$ is decomposed:

$$\mathbf{K}_{t+1} = \begin{bmatrix} \mathbf{K}_t & \mathbf{k}_{t+1} \\ \mathbf{k}_{t+1}^T & k_{t+1}^* \end{bmatrix}. \tag{D.1}$$

Assuming $\mathbf{K}_t^{-1}$ known and applying the matrix inversion lemma for $\mathbf{K}_{t+1}$,

$$\begin{aligned} \mathbf{K}_{t+1}^{-1} &= \begin{bmatrix} \mathbf{K}_t & \mathbf{k}_{t+1} \\ \mathbf{k}_{t+1}^T & k_{t+1}^* \end{bmatrix}^{-1} \\ &= \begin{bmatrix} \mathbf{K}_t^{-1} + \mathbf{K}_t^{-1}\mathbf{k}_{t+1}\mathbf{k}_{t+1}^T\mathbf{K}_t^{-1}\gamma_{t+1}^{-1} & -\mathbf{K}_t^{-1}\mathbf{k}_{t+1}\gamma_{t+1}^{-1} \\ -\mathbf{k}_{t+1}^T\mathbf{K}_t^{-1}\gamma_{t+1}^{-1} & \gamma_{t+1}^{-1} \end{bmatrix}, \end{aligned} \tag{D.2}$$

where $\gamma_{t+1} = k_{t+1}^* - \mathbf{k}_{t+1}^T\mathbf{K}_t^{-1}\mathbf{k}_{t+1}$ is the geometric term from equation 3.14. Using notations $\mathbf{K}_t^{-1}\mathbf{k}_{t+1} = \hat{\mathbf{e}}_{t+1}$ from equation 3.7, $\mathbf{K}_t^{-1} = \mathbf{Q}_t$, and $\mathbf{K}_{t+1}^{-1} = \mathbf{Q}_{t+1}$, we have a recursion equation,

$$\mathbf{Q}_{t+1} = \begin{bmatrix} \mathbf{Q}_t + \gamma_{t+1}^{-1}\hat{\mathbf{e}}_{t+1}\hat{\mathbf{e}}_{t+1}^T & -\gamma_{t+1}^{-1}\hat{\mathbf{e}}_{t+1} \\ -\gamma_{t+1}^{-1}\hat{\mathbf{e}}_{t+1}^T & \gamma_{t+1}^{-1} \end{bmatrix}, \tag{D.3}$$

and in a more compact matrix notation,

$$\mathbf{Q}_{t+1} = \mathbf{Q}_t + \gamma_{t+1}^{-1}(\hat{\mathbf{e}}_{t+1} - \mathbf{e}_{t+1})(\hat{\mathbf{e}}_{t+1} - \mathbf{e}_{t+1})^T, \tag{D.4}$$

where $\mathbf{e}_{t+1}$ is the $t+1$th unit vector. With this recursion equation, all matrix inversion is eliminated (this result is general for block matrices; such implementation, together with an interpretation of the parameters has been also made in (Cauwenberghs & Poggio, 2001). Using the score (see equation 3.17) and including in $\mathcal{BV}$ only inputs with nonzero scores, the Gram matrix is guaranteed to be nonsingular; $\gamma_{t+1} > 0$ guarantees nonsingularity of the extended Gram matrix.

For numerical stability, we can use the Cholesky decomposition of the inverse Gram matrix $\mathbf{Q}$. Using the lower triangular matrix $\mathbf{R}$ with the cor-

[3] A useful guide to formulas for matrix inversions and block matrix manipulation can be found at Sam Roweis's home page: `http://www.gatsby.ucl.ac.uk/~roweis/notes.html`.

responding indices and the identity $\mathbf{Q} = \mathbf{R}^T\mathbf{R}$, we have the update for the Cholesky decomposition,

$$\mathbf{R}_{t+1} = \begin{pmatrix} \mathbf{R}_t & 0 \\ -\gamma_{t+1}^{-1/2}\hat{\mathbf{e}}_{t+1}^T & \gamma_{t+1}^{-1/2} \end{pmatrix}, \tag{D.5}$$

which is a computationally very inexpensive operation, without additional operations, provided that the quantities γ_{t+1} and $\mathbf{e}_{t+1}$ are already computed.

Appendix E: Deleting a $\mathcal{BV}$

Adding a basis vector is made with the equations:

$$\begin{aligned} \mathbf{s}_{t+1} &= T_{t+1}\left(\mathbf{C}_t\mathbf{k}_{t+1}\right) + \mathbf{e}_{t+1} \\ \boldsymbol{\alpha}_{t+1} &= T_{t+1}\left(\boldsymbol{\alpha}_t\right) + q^{(t+1)}\mathbf{s}_{t+1} && \text{(E.1)} \\ \mathbf{C}_{t+1} &= U_{t+1}\left(\mathbf{C}_t\right) + r^{(t+1)}\mathbf{s}_{t+1}\mathbf{s}_{t+1}^T && \text{(E.2)} \\ \mathbf{Q}_{t+1} &= U_{t+1}(\mathbf{Q}_t) + \gamma_{t+1}^{-1}(T_{t+1}(\hat{\mathbf{e}}_{t+1}) - \mathbf{e}_{t+1})(T_{t+1}(\hat{\mathbf{e}}_{t+1}) - \mathbf{e}_{t+1})^T && \text{(E.3)} \end{aligned}$$

where $\boldsymbol{\alpha}$ and $\mathbf{C}$ are the GP parameters, $\mathbf{Q}$ is the inverse Gram matrix, γ_{t+1} and $\hat{\mathbf{e}}_{t+1}$ the geometric terms of the new basis vector, $\mathbf{k}_{t+1} = [K_0(x_1, x_{t+1}), \ldots, K_0(x_t, x_{t+1})]^T$, and $\mathbf{e}_{t+1}$ is the $t+1$th unity vector.

The optimal decrease of $\mathcal{BV}$s needs an answer to two questions. The first question is how to delete a basis vector from the set of basis vectors with minimal loss of information. If the method is given, then we have to find the $\mathcal{BV}$ to remove. The first problem is solved by inverting the learning equations, E.1 through E.3. Assuming $\boldsymbol{\alpha}_{t+1}$, $\mathbf{C}_{t+1}$, and $\mathbf{Q}_{t+1}$ known and using pairwise correspondence considering the $t+1$th element of $\boldsymbol{\alpha}_{t+1}$, we can identify $q^{(t+1)} = \alpha_{t+1}(t+1) \overset{def}{=} \alpha^*_{t+1}$ (the notations are illustrated in Figure 3). Using similar correspondences for the matrix $\mathbf{C}_{t+1}$, the following identifications can be done:

$$\begin{aligned} r^{(t+1)} &= \mathbf{C}_{t+1}(t+1, t+1) \overset{def}{=} c^*_{t+1} && \text{(E.4)} \\ \mathbf{C}_t\mathbf{k}_{t+1} &= \mathbf{C}_{t+1}(1..t, t+1) \overset{def}{=} \frac{\mathbf{C}^*_{t+1}}{c^*_{t+1}} \end{aligned}$$

with c^*_{t+1} and $\mathbf{C}^*_{t+1}$ sketched in Figure 3. Substituting back into equations E.1 and E.2, the old values for GP parameters are:

$$\boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{t+1}^{(t)} - \alpha^*_{t+1}\frac{\mathbf{C}^*_{t+1}}{c^*_{t+1}} \tag{E.5}$$

$$\mathbf{C}_t = \mathbf{C}_{t+1}^{(t)} - \frac{\mathbf{C}^*_{t+1}\mathbf{C}^{*T}_{t+1}}{c^*_{t+1}}, \tag{E.6}$$

where $\boldsymbol{\alpha}_{t+1}^{(t)} = T_{t+1}^{-1}(\boldsymbol{\alpha}_{t+1})$ and T_{t+1}^{-1} is the inverse operator that takes the first t elements of a $t+1$-dimensional vector. We define $\mathbf{C}_{t+1}^{(t)} = U_{t+1}^{-1}(\mathbf{C}_{t+1})$ similarly.

Proceeding similarly, using elements of matrix $\mathbf{Q}_{t+1}$, the correspondence with equation E.3 is as follows:

$$\begin{aligned} \gamma_{t+1} &= \frac{1}{Q_{t+1}(t+1, t+1)} \overset{def}{=} \frac{1}{q_{t+1}^{*}} \qquad \text{(E.7)} \\ \hat{\mathbf{e}}_{t+1} &= -\frac{\mathbf{Q}_{t+1}(1..t, t+1)}{q_{t+1}^{*}} \overset{def}{=} -\frac{\mathbf{Q}_{t+1}^{*}}{q_{t+1}^{*}}, \end{aligned}$$

with the reduced set matrix $\hat{\mathbf{Q}}_{t+1}$:

$$\hat{\mathbf{Q}}_{t+1} = \mathbf{Q}_{t+1}^{(t)} - \frac{\mathbf{Q}_{t+1}^{*}\mathbf{Q}_{t+1}^{*T}}{q_{t+1}^{*}}. \qquad \text{(E.8)}$$

The matrix $\mathbf{Q}$ does not need any further modification; however, for $\boldsymbol{\alpha}$ and $\mathbf{C}$, a sparse update is needed. Replacing γ_{t+1}, $\hat{\mathbf{e}}_{t+1}$ together with the "old" parameters $\boldsymbol{\alpha}_t$ and $\mathbf{C}_t$, we have the "optimally reduced" GP parameters as

$$\begin{aligned} \hat{\boldsymbol{\alpha}}_t &= \boldsymbol{\alpha}_{t+1}^{(t)} + \alpha_{t+1}^{*}\frac{\mathbf{Q}_{t+1}^{*}}{q_{t+1}^{*}} \qquad \text{(E.9)} \\ \hat{\mathbf{C}}_t &= \mathbf{C}_{t+1}^{(t)} + c_{t+1}^{*}\frac{\mathbf{Q}_{t+1}^{*}\mathbf{Q}_{t+1}^{*T}}{q_{t+1}^{*2}} - \frac{1}{q_{t+1}^{*}}\left[\mathbf{Q}_{t+1}^{*}\mathbf{C}_{t+1}^{*T} + \mathbf{C}_{t+1}^{*}\mathbf{Q}_{t+1}^{*T}\right]. \qquad \text{(E.10)} \end{aligned}$$

References

Berliner, L. M., Wikle, L., & Cressie, N. (2000). Long-lead prediction of Pacific SST via Bayesian dynamic modelling. *Journal of Climate, 13*, 3953–3968.

Bernardo, J. M., & Smith, A. F. (1994). *Bayesian theory.* New York: Wiley.

Bottou, L. (1998). Online learning and stochastic approximations. In D. Saad (Ed.), *On-line learning in neural networks* (pp. 9–42). Cambridge: Cambridge University Press.

Cauwenberghs, G., & Poggio, T. (2001). Incremental and decremental support vector machine learning. In T. K. Leen, T. G. Diettrich, & V. Tresp (Eds.), *Advances in neural information processing systems, 13*. Cambridge, MA: MIT Press.

Csató, L., Fokoué, E., Opper, M., Schottky, B., & Winther, O. (2000). Efficient approaches to gaussian process classification. In S. A. Solla, T. K. Leen, & K.-R. Müller (Eds.), *Advances in neural information processing systems.* Cambridge, MA: MIT Press.

Evans, D. J., Cornford, D., & Nabney, I. T. (2000). Structured neural network modelling for multi-valued functions for wind vector retrieval from satellite scatterometer measurements. *Neurocomputing, 30*, 23–30.

Friedman, J. H. (1991). Multivariate adaptive regression splines. *Annals of Statistics, 19*, 1–141.

Gibbs, M., & MacKay, D. J. (1999). *Efficient implementation of gaussian processes* (Tech. Rep.) Cambridge: Department of Physics, Cavendish Laboratory, Cambridge University. Available at: http://wol.ra.phy.cam.ac.uk/mackay/abstracts/gpros.html.

Jaakkola, T., & Haussler, D. (1999). Probabilistic kernel regression. In *Online Proceedings of 7th Int. Workshop on AI and Statistics*. Available at: http://uncertainty99.microsoft.com/proceedings.htm.

Kimeldorf, G., & Wahba, G. (1971). Some results on Tchebycheffian spline functions. *J. Math. Anal. Applic., 33*, 82–95.

Lemm, J. C., Uhlig, J., & Weiguny, A. (2000). A Bayesian approach to inverse quantum statistics. *Phys. Rev. Lett., 84*, 2068–2071.

McLachlan, A., & Lowe, D. (1996). Tracking of non-stationary time-series using resource allocating RBF networks. In R. Trappl (Ed.), *Cybernetics and Systems '96* (pp. 1066–1071). Proceedings of the 13th European Meeting on Cybernetics and Systems Research. Vienna: Austrian Society for Cybernetic Studies.

Minka, T. P. (2000). *Expectation propagation for approximate Bayesian inference*. Unpublished doctoral dissertation, MIT. Available at: vismod.www.media.mit.edu/~tpminka/.

Neal, R. M. (1997). Regression and classification using gaussian process priors (with discussion). In J. M. Bernerdo, J. O. Berger, A. P. Dawid, & A. F. M. Smith (Eds.), *Bayesian statistics* (Vol. 6). New York: Oxford University Press. Available at: ftp://ftp.cs .utoronto .ca/pub /radford/mc-gp.ps.Z.

Opper, M. (1996). Online versus offline learning from random examples: General results. *Phys. Rev. Lett., 77*(22), 4671–4674.

Opper, M. (1998). A Bayesian approach to online learning. In D. Saad (Ed.), *On-line learning in neural networks* (pp. 363–378). Cambridge: Cambridge University Press.

Opper, M., & Winther, O. (1999). Gaussian processes and SVM: Mean field results and leave-one-out estimator. In A. Smola, P. Bartlett, B. Schölkopf, & C. Schuurmans (Eds.), *Advances in large margin classifiers* (pp. 43–65). Cambridge, MA: MIT Press.

Platt, J. (1991). A resource-allocating network for function interpolation. *Neural Computation, 3*, 213–225.

Schölkopf, B., Mika, S., Burges, C. J., Knirsch, P., Müller, K.-R., Rätsch, G., & Smola, A. J. (1999). Input space vs. feature space in kernel-based methods. *IEEE Transactions on Neural Networks, 10*(5), 1000–1017.

Seeger, M. (2000). Bayesian model selection for support vector machines, gaussian processes and other kernel classifiers. In S. A. Solla, T. K. Leen, & K.-R. Müller (Eds.), *Advances in neural information processing systems, 12.* Cambridge, MA: MIT Press.

Smola, A. J., & Schölkopf, B. (2000). Sparse greedy matrix approxmation for machine learning. In *International Conference on Machine Learning*. San Mateo, CA: Morgan Kaufmann. Available at: www.kernel-machines.org/papers/upload_4467_kfa_long.ps.gz.

Tipping, M. (2000). The relevance vector machine. In S. A. Solla, T. K. Leen, & K.-R. Müller (Eds.), *Advances in neural information processing systems, 12.* Cambridge, MA: MIT Press.

Trecate, G. F., Williams, C. K. I., & Opper, M. (1999). Finite-dimensional approximation of gaussian processes. In M. S. Kearns, S. A. Solla, & D. A. Cohn (Eds.), *Advances in neural information processing systems, 11.* Cambridge, MA: MIT Press.

Tresp, V. (2000). A Bayesian committee machine. *Neural Computation, 12*(11), 2719–2741.

Vapnik, V. N. (1995). *The nature of statistical learning theory*. New York: Springer-Verlag.

Vijayakumar, S., & Ogawa, H. (1999). RKHS based functional analysis for exact incremental learning. *Neurocomputing, 29*(1–3), 85–113.

Wahba, G. (1990). *Splines models for observational data*. Philadelphia: SIAM.

Williams, C. K. I. (1999). Prediction with gaussian processes. In M. I. Jordan (Ed.), *Learning in graphical models*. Cambridge, MA: MIT Press.

Williams, C. K. I., & Barber, D. (1998). Bayesian classification with gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 20*(12), 1342–1351.

Williams, C. K. I., & Rasmussen, C. E. (1996). Gaussian processes for regression. In D. S. Touretzky, M. C. Mozer, & M. E. Hasselmo (Eds.), *Advances in neural information processing systems, 8.* Cambridge, MA: MIT Press.

Williams, C. K. I., & Seeger, M. (2001). Using the Nyström method to speed up kernel machines. In T. K. Leen, T. G. Diettrich, & V. Tresp (Eds.), *Advances in neural information processing systems, 13*. Cambridge, MA: MIT Press.

Received February 23, 2001; accepted May 24, 2001.