Neural Computation, Vol. 12, pages 2719-2741, 2001

A Bayesian Committee Machine

Volker Tresp
Siemens AG, Corporate Technology
Dept. Information and Communications
Otto-Hahn-Ring 6, 81730 Munich, Germany
Volker.Tresp@mchp.siemens.de

Abstract

The Bayesian committee machine (BCM) is a novel approach to combining estimators which were trained on different data sets. Although the BCM can be applied to the combination of any kind of estimators the main foci are Gaussian process regression and related systems such as regularization networks and smoothing splines for which the degrees of freedom increase with the number of training data. Somewhat surprisingly, we find that the performance of the BCM improves if several test points are queried at the same time and is optimal if the number of test points is at least as large as the degrees of freedom of the estimator. The BCM also provides a new solution for online learning with potential applications to data mining. We apply the BCM to systems with fixed basis functions and discuss its relationship to Gaussian process regression. Finally, we also show how the ideas behind the BCM can be applied in a non-Bayesian setting to extend the input dependent combination of estimators.

1 Introduction

For reasons typically associated with their architectures and their learning algorithms, some learning systems are limited in their capability to handle large data sets and to perform online learning. Typical examples are kernel based approaches such as smoothing splines, regularization networks, kriging and Gaussian process regression where it is necessary to invert matrices of the size of the training data set and where the number of kernels grows proportionally to the number of training data. All four systems are closely related and are considered biologically relevant models for information processing (Poggio and Girosi, 1990). Gaussian process regression is an approach recently introduced into the machine learning community (Neal, 1996, Rasmussen, 1996, Williams, 1996, 1997, 1998a, Gibbs and MacKay, 1997, MacKay, 1998) and is considered suitable for applications with only up to a few hundred training samples. The reason is that the computational cost of the required matrix inversion scales as $\mathcal{O}(K^3)$ where K is the number of training data. In this paper we introduce an approximate solution to regression whose computational cost only increases linearly with the number of training patterns and which is applicable in particular to kernel based regression systems but also to other regression models. The idea is to split up the data set in M data sets and train M systems on the data sets. We then combine the predictions of the individual systems using a new weighting scheme in form of a Bayesian committee machine (BCM). This scheme is an extension to the input dependent averaging of estimators, a procedure which was introduced by Tresp and Taniguchi, 1995 and Taniguchi and Tresp, 1997. A particular application of our solution is online learning where data arrive sequentially and training must be performed sequentially as well. We present an implementation of our approach which is capable of online learning and which requires the storage of only one matrix of the size of the number of query points.

In Section 2 we develop the BCM in a general setting and in Section 3 we apply the BCM to kernel regression systems, in particular Gaussian process regression. In Section 4 we apply the BCM to regression with fixed basis functions. We show that in the special case that we have as least as many query points as there are linearly independent basis functions we obtain the optimal prediction. In Section 5 we explain the principle behind the BCM. We show that the the dimension and the effective degrees of freedom P_{eff}^{Data} of an estimator are important quantities for determining the optimal number of query points. In Section 6 we give a frequentist version of the BCM and show how it can be used to improve the input dependent combination of estimators by Tresp and Taniguchi (1995). In Section 7 we derive online Kalman filter versions of our approach and show how predictions at additional query points can be calculated. In Section 8 we demonstrate the effectiveness of our approach using several data sets. In Section 9 we present our conclusions.

2 Derivation of the BCM

2.1 Combining Bayesian Estimators

Let's assume an input variable x and a one dimensional real response variable $f(x)$. We assume that we can only measure a noisy version of f

$$g(x) = f(x) + \epsilon(x)$$

where $\epsilon(x)$ is independent Gaussian distributed noise with zero mean and variance $\sigma_\psi^2(x)$.

Let $X^m = \{x_1^m, \ldots, x_K^m\}$ denote the input vectors of the training data set of size K (superscript m for measurement) and let $g^m = (g_1 \ldots g_K)'$ be the vector of the corresponding target measurements. Let $D = \{X^m, g^m\}$ denote both inputs and targets.

Furthermore, let $X^q = \{x_1^q, \ldots, x_{N_Q}^q\}$ denote a set of N_Q query or test points (superscript q for query) and let $f^q = (f_1, \ldots, f_{N_Q})$ be the vector of the corresponding unknown response variables.

We assume now a setting where instead of training one estimator using all the data we split up the data into M data sets $D = \{D^1, \ldots, D^M\}$ (of typically approximately same size) and train M estimators separately on each training data set. Gaussian process regression which will be introduced in the next sections is one example where this procedure is useful. Correspondingly, we partition inputs $X^m = \{X_1^m, \ldots, X_M^m\}$ and targets $g^m = \{g_1^m, \ldots, g_M^m\}$. Let $\mathcal{D}^i = \{D^1, \ldots, D^i\}$ denote the data sets with indices smaller or equal to i with $i = 1, \ldots M$.

Let $P(f^q|D^i)$ be the posterior predictive probability density at the N_Q query points for the ith estimator.[1] Then we have in general[2]

$$P(f^q|\mathcal{D}^{i-1}, D^i) \propto P(f^q)P(\mathcal{D}^{i-1}|f^q)P(D^i|\mathcal{D}^{i-1}, f^q).$$

Now we would like to approximate

$$P(D^i|\mathcal{D}^{i-1}, f^q) \approx P(D^i|f^q). \tag{1}$$

This is not true in general unless $f^q \equiv f$: only conditioned on the complete function f are targets independent. The approximation might still be reasonable when, first, N_Q is large — since then f^q determines f everywhere — and when, second, the correlation between the targets in $\mathcal{D}^{i-1}$ and D^i is small, for example if inputs in those sets are spatially separated from each other.

Using the approximation and applying Bayes' formula, we obtain

$$P(f^q|\mathcal{D}^{i-1}, D^i) \approx const \times \frac{P(f^q|\mathcal{D}^{i-1})P(f^q|D^i)}{P(f^q)} \tag{2}$$

such that we can achieve an approximate predictive density

$$\hat{P}(f^q|D) = const \times \frac{\prod_{i=1}^{M} P(f^q|D^i)}{P(f^q)^{M-1}} \tag{3}$$

where *const* is a normalizing constant. The posterior predictive probability densities are simply multiplied. Note that since we multiply posterior probability densities, we have to divide by the priors $M-1$ times. This general formula can be applied to the combination of any Bayesian regression estimator.

2.2 The BCM

In case that the predictive densities are Gaussian (or can be approximately reasonably well by a Gaussian) Equation 2 takes on an especially simple form. Let's assume that the *a priori* predictive density at the N_Q query points is a Gaussian with zero mean and covariance Σ^{qq} and the posterior predictive density for each module is a Gaussian with mean $E(f^q|D^i)$ and covariance $cov(f^q|D^i)$. In this case we achieve (see Appendix 10.1, $\hat{E}$ and $\widehat{cov}$ are caclulated w.r.t. the approximate density $\hat{P}$)

$$\hat{E}(f^q|D) = C^{-1} \sum_{i=1}^{M} cov(f^q|D^i)^{-1} E(f^q|D^i) \tag{4}$$

with

$$C = \widehat{cov}(f^q|D)^{-1} = -(M-1)(\Sigma^{qq})^{-1} + \sum_{i=1}^{M} cov(f^q|D^i)^{-1}. \tag{5}$$

[1] As an example, for an estimator parameterized by a weight vector w, the posterior predictive probability density is $P(f^q|D^i) = \int P(f^q|w)P(w|D^i)dw$ (Bernardo and Smith, 1994).

[2] In this paper we assume that inputs are given and an expression such as $P(D|f^q)$ denotes the probability density of only the targets g^m conditioned both on f^q and inputs X^m.

We recognize that the prediction of each module i is weighted by the inverse covariance of its prediction. But note that we do not compute the covariance between the modules but the covariance between the N_Q query points! We have named this way of combining the predictions of the modules the Bayesian committee machine (BCM). Modules which are uncertain about their predictions are automatically weighted less than modules which are certain about their prediction.

In the next sections we will apply the BCM to two important special cases with Gaussian posterior predictive densities, first to Gaussian process regression and second to systems with fixed basis functions.

3 Gaussian Process Regression and the BCM

An important application of the BCM is kernel based regression in the form of regularization networks, smoothing splines, kriging and Gaussian processes regression. The reason is that first, the degrees of freedom in such systems increase with the number of training data points such that it is not suitable to update posterior parameter probabilities and that second, the approaches require the inversion of matrices of the dimension of the number of data points which is clearly unsuitable for large data sets. Since it is well known that all four approaches can give equivalent solutions we will focus on only one of them, namely the Gaussian process regression framework. In the next section we briefly review Gaussian process regression and in the following section we discuss the BCM in context with Gaussian process regression.

3.1 A Short Review of Gaussian Process Regression

In contrast to the ususal parameterized approach to regression, in Gaussian process regression we specify the prior model directly in function space. In particular we assume that *a priori* $f(x)$ is Gaussian distributed (in fact it would be an infinite-dimensional Gaussian) with zero mean and a covariance $cov(f(x_1), f(x_2)) = \sigma_{x_1,x_2}$. As before, we assume that we can only measure a noisy version of f

$$g(x) = f(x) + \epsilon(x)$$

where $\epsilon(x)$ is independent Gaussian distributed noise with zero mean and variance $\sigma_\psi^2(x)$.

Let $(\Sigma^{mm})_{ij} = \sigma_{x_i^m, x_j^m}$ be the covariance matrix of the measurements, $\Psi^{mm} = \sigma_\psi^2(x) I$ be the noise-variance matrix, $(\Sigma^{qq})_{ij} = \sigma_{x_i^q, x_j^q}$ be the covariance matrix of the query points and $(\Sigma^{qm})_{ij} = \sigma_{x_i^q, x_j^m}$ be the covariance matrix between training data and query data. I is the K-dimensional unit matrix.

Under these assumptions the conditional density of the response variables at the query points is Gaussian distributed with mean

$$E(f^q|D) = \Sigma^{qm}(\Psi^{mm} + \Sigma^{mm})^{-1} g^m, \tag{6}$$

and covariance

$$cov(f^q|D) = \Sigma^{qq} - \Sigma^{qm}\ (\Psi^{mm} + \Sigma^{mm})^{-1}\ (\Sigma^{qm})'. \tag{7}$$

Note, that for the $i-$th query point we obtain

$$E(f_i^q|D) = \sum_{j=1}^{K} \sigma_{x_i^q, x_j^m} \; v_j \tag{8}$$

where v_j is the $j-$th component of the vector $(\Psi^{mm} + \Sigma^{mm})^{-1} g^m$. The last equation describes the weighted superposition of kernel functions $b_j(x_i^q) = \sigma_{x_i^q, x_j^m}$ which are defined for each training data point and is equivalent to some solutions obtained for kriging, regularization networks and smoothing splines (Wahba, 1990, Poggio and Girosi, 1990, Williams, 1996, MacKay, 1998, Hastie and Tibshirani, 1990). A common assumptions is that $\sigma_{x_i,x_j} = A\exp(-1/(2\gamma^2)||x_i - x_j||^2)$ with $A, \gamma > 0$ such that we obtain Gaussian basis functions.

3.2 The BCM for Gaussian Process Regression

Since the computational cost of the matrix inversion in Equation 6 scales as $\mathcal{O}(K^3)$ where K is the number of training data, the cost becomes prohibitive if K is large. There have been a few attempts at finding more efficient approximate solutions. Gibbs and MacKay (1997) have applied an algorithm developed by Skilling to Gaussian process regression (Skilling, 1993). This iterative algorithm scales as $\mathcal{O}(K^2)$. In the regression-filter algorithm by Zhu and Rohwer (1996) and Zhu *et al.* (1998), the functional space is projected into a lower-dimensional basis-function space where the Kalman filter algorithm can be used for training the coefficients of the basis functions. The latter approach scales as $\mathcal{O}(KN^2)$ where N is the number of basis functions.

The big advantage of the application of the BCM (Equation 4) to Gaussian process regression (Equations 6 and 7) is that now instead of having to invert a K-dimensional matrix we only have to invert approximately (K/M)-dimensional and N_Q-dimensional matrices. If we set $M = K/\alpha$ where α is a constant, the BCM scales linearly in K. We found that $M = K/N_Q$ is a good choice to optimize the number of computations since then all matrices which have to be inverted have the same size. In this case the number of elements in each module is identical to N_Q, the number of query points. In Section 7 we will describe online versions of the algorithm which are well suited for data mining applications.

If N_Q is large it can be more efficient to work with another formulation of the BCM which is described in Appendix 10.2.

4 The BCM and Regression with Fixed Basis Functions

4.1 Regression with Fixed Basis Functions

In the previous section the BCM was applied to a kernel based system, i.e. Gaussian process regression. For kernel based systems the BCM can be very useful for online learning and when the training data set is large. For regression with fixed basis functions, on the other hand, sufficient statistics exist such that large data sets and online learning is not problematic. Still there are two good reasons why we want to study the BCM in the context of systems with fixed basis functions. First, the system under consideration might have a large number —or even an infinite number— of basis functions such that working with posterior weight densities is infeasible and second, the study of these systems provides insights into kernel based systems.

Consider the same situation as in Section 2, only that $f(x)$ is a superposition of N fixed basis functions, i.e.

$$f(x) = \sum_{i=1}^{N} w_i \phi_i(x)$$

where $w = (w_1, \ldots, w_N)'$ is the $N-$dimensional weight vector with a Gaussian weight prior with mean 0 and covariance Σ_w. Let $(\Phi)_{ki} = \phi_i(x_k^m)$ be the design matrix, i.e. the matrix of the activations of the basis functions at the training data. Furthermore, let $(\Phi^q)_{ki} = \phi_i(x_k^q)$ be the corresponding matrix for the query points. We obtain for the posterior weights a Gaussian density with mean

$$E(w|D) = \left(\Sigma_w^{-1} + \Phi'(\Psi^{mm})^{-1}\Phi\right)^{-1} \Phi'(\Psi^{mm})^{-1} g^m \tag{9}$$

and covariance

$$cov(w|D) = \left(\Sigma_w^{-1} + \Phi'(\Psi^{mm})^{-1}\Phi\right)^{-1}.$$

The posterior density of the query point is then also Gaussian with mean

$$E(f^q|D) = \Phi^q E(w|D) = \Phi^q \left(\Sigma_w^{-1} + \Phi'(\Psi^{mm})^{-1}\Phi\right)^{-1} \Phi'(\Psi^{mm})^{-1} g^m \tag{10}$$

and covariance

$$cov(f^q|D) = \Phi^q \left(\Sigma_w^{-1} + \Phi'(\Psi^{mm})^{-1}\Phi\right)^{-1} (\Phi^q)'.$$

By substituting these expressions into Equations 4 and 5 we obtain the BCM solution. Note, that here $\Sigma^{qq} = \Phi^q \Sigma_w (\Phi^q)'$. As a special case consider that Φ^q is a nonsingular square matrix, i.e. there are as many query points as there are parameters N. Then we obtain after some manipulations

$$\hat{E}(f^q|D) = E(f^q|D)$$

i.e. the BCM approximation is identical to the solution we would have obtained if we had done Bayesian parameter updates. Assuming the model assumptions are correct this is the Bayes optimal prediction! This of course should be no surprise since N appropriate data points uniquely define the weights w and therefore also f is uniquely defined and the approximation of Equation 1 becomes an equality.[3] We can see this also from Equation 10. If Φ^q is a square matrix with full rank, the optimal weight vector can be recovered from the expected value of f^q by matrix inversion.

[3]For nonlinear systems we can replace the design matrix Φ by the matrix of first derivatives of the map at the inputs w.r.t. the weights (Tresp and Taniguchi, 1995). The equations in this section are then valid within the linearized approximation.

4.2 Fixed Basis Functions and Gaussian Process Regression

Note that by using basic matrix manipulations we can write the expectation in Equation 10 also as

$$E(f^q|D) = \Phi^q \Sigma_w \Phi' (\Psi^{mm} + \Phi \Sigma_w \Phi')^{-1} g^m \tag{11}$$

and equivalently

$$cov(f^q|D) = \Phi^q \Sigma_w (\Phi^q)' - \Phi^q \Sigma_w \Phi' \left(\Psi^{mm} + \Phi \Sigma_w \Phi'\right)^{-1} \Phi \Sigma'_w (\Phi^q)'.$$

This is mathematically equivalent to Equation 6 and 7 if we substitute

$$\Sigma^{mm} = \Phi \Sigma_w \Phi' \quad \Sigma^{qm} = \Phi^q \Sigma_w \Phi' \quad \Sigma^{qq} = \Phi^q \Sigma_w (\Phi^q)'. \tag{12}$$

This means that the solution we have obtained for fixed basis functions is identical to the solution for Gaussian process regression with kernel functions

$$b_j(x_i^q) = \sigma_{x_i^q, x_j^m} = \phi(x_i^q)' \Sigma_w \phi(x_j^m) \tag{13}$$

where $\phi(x) = (\phi_1(x), \ldots, \phi_N(x))'$ is the activation of the fixed basis functions at x. From this it is obvious that the kernel functions are simply linear combinations of the fixed basis functions and "live in" the same functional space. The maximum dimension of this space is equal to the number of fixed basis functions (assuming these are linearly independent).[4] In the context of smoothing splines the relationship between fixed basis functions and kernel regression is explored by Wahba (1990) and in the context of a Bayesian setting by Neal (1996, 1997), Williams (1996, 1997, 1998a, 1998b) and MacKay (1998).

Note that according to Equation 13, the covariance function of the response variables and therefore also the kernel function is defined by inner products in feature space. In the work on support vector machines (Vapnik, 1995) this relationship is also explored. There the question is posed, given the kernel function $b_j(x)$, is there a fixed basis function expansion which corresponds to these kernel functions. One answer is given by Mercer's theorem which states that for symmetric and positive definite kernels a corresponding set of fixed basis function exists. Note also that if the kernel is known and if the number of basis functions is much larger than the number of training data points, the calculations in Equation 10 are much more expensive than the calculations in Equation 11, resp. 6.

The relationship between the support vector machine, kernel based regression and and basis function approaches was recently explored in papers by Poggio and Girosi (1998), Girosi (1998) and Smola, Schölkopf and Müller (1998). Note, that we can define the priors either in weight space with covariance Σ_w or in function space with covariance Σ^{mm}. This duality is also explored in Moody and Rögnvaldsson (1997) and in the regression-filter algorithm by Zhu and Rohwer (1996).

[4] An interesting case is when the number of fixed basis functions is infinite. Consider the case that we have an infinite number of uniformly distributed fixed radial Gaussian basis functions $\phi_i(x) = G(x; x_i, \sigma^2)$ and $\Sigma_w = I$ where I is the unit matrix. Then $b_j(x_i^q) = \int G(x_i^q; x, \sigma^2) G(x_j^m; x, \sigma^2) dx \propto G(x_i^q; x_j^m, 2\sigma^2)$ which is a Gaussian with twice the variance. We will use Gaussian kernel functions in the experiments. $G(x; c, \Sigma)$ is our notation for a normal density function with mean c and covariance Σ evaluated at x.

5 Why the BCM Works

The central question is under which conditions Equation 1 is a good approximation. One case is when the data sets are unconditionally independent, i.e. $P(D^i|\mathcal{D}^{i-1}) \approx P(D^i)$. This might be achieved by first clustering the data and by then assigning the data of each cluster to a separate module. A second case where the approximation is valid, is if the data sets are independent conditioned on f^q. In the previous section we have already seen an example where we even have the equality $P(D^i|\mathcal{D}^{i-1}, f^q) = P(D^i|f^q)$. We have shown that for a system with N fixed basis functions, N query points are sufficient such that the BCM provides the Bayes optimal prediction. Most kernel based systems, although, have infinite degrees of freedom which would require an infinite number of query points to make the previous equality valid. We have seen one exception in the previous section, i.e. a (degenerate) Gaussian process regression system which has the same number of degrees of freedom as the equivalent system with fixed basis functions.

In this section we will show that even for Gaussian process regression with infinte degrees of freedom, in typical cases only a limited number of degrees of freedom are really used such that even with a small number of query points, the BCM approximation is reasonable.

5.1 The Dimension and the *effective* Degrees of Freedom

In Gaussian process regression, the regression function lives in the space spanned by the kernel functions (Equations 6 and 8). Since the number of kernel functions is equal to the number of data points, the dimension of the subspace spanned by the kernel functions is upper bounded by the number of data points K. For two reasons, the dimension might be smaller than K. The first reason is that the data points are degenerate, for example if some input training data points appear several times in the data set. The second reason is that the kernel functions themselves are degenerate such that even for an arbitrary selection of input data, the dimension of the functional space represented by the kernel functions is upper bounded by dim^{max}. An example was given in section 4 where we gave an example of a finite dim^{max}: for the system of fixed basis functions and the equivalent Gaussian process regression system we have $dim^{max} = N$.

In addition to the dimension, we can also define the effective degrees of freedom kernel regression (Hastie and Tibshirani, 1990). A reasonable definition for Gaussian process regression is

$$P_{eff}^{Data} = trace\,(\Sigma^{mm}(\Psi^{mm} + \Sigma^{mm})^{-1})$$

which measures how many degrees of freedom are used by the given data. This is analogous to the definition of the effective number of parameters by Wahba (1983), Hastie and Tibshirani (1990), Moody (1992) and MacKay (1992).[5] Let $\{\lambda_0, \ldots \lambda_K\}$ be the eigenvalues of Σ^{mm}. Then we obtain

$$P_{eff}^{Data} = \sum_{i=1}^{K} \frac{\lambda_i}{\lambda_i + \sigma_\psi^2}.$$

[5]Note, that $\Sigma^{mm}(\Psi^{mm}+\Sigma^{mm})^{-1}$ maps training targets into predictions at the same locations. This expression is therefore equivalent to the so called hat matrix in linear regression. There analogously, the trace of the hat matrix is defined as the effective number of parameters.

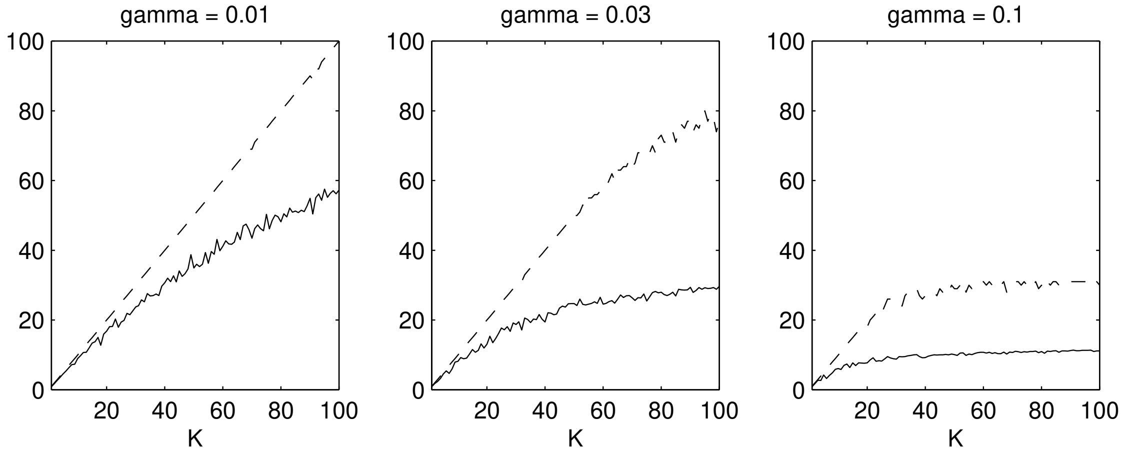

Figure 1: Shown are the effective degrees of freedom P_{eff}^{Data} (continuous) and the rank of Σ^{mm} (dashed) as a function of the number of training data K. The rank of Σ^{mm} is a lower bound of the dimension of the space spanned by the kernel functions. The rank is here calculated as the number of singular values that are larger than $tol = K \times norm(\Sigma^{mm}) \times eps$ where $eps = 2.2204 \times 10^{-16}$ and $norm(\Sigma^{mm})$ is the largest singular value of Σ^{mm} (default matlab settings). As in the experiments in Section 8, for two locations in input space x_i and x_j, we set $\sigma_{x_i,x_j} = \exp(-1/(2\gamma^2)||x_i - x_j||^2)$ where γ is a positive constant. The locations of the input training data were chosen randomly in the interval $[0, 1]$. For the noise variance we have $\sigma_\psi = 0.1$. Note that P_{eff}^{Data} is always smaller than K, especially for large K. P_{eff}^{Data} also decreases when we assume a strong correlation (i.e. a large γ). The rank is larger than P_{eff}^{Data} but also smaller than K in particular for large γ and K. For this particular covariance structure dim^{max} is theoretically infinite. As the plots show — if the input data are drawn always from the same distribution — for practical purposes dim^{max} might be considered to be finite if γ is large.

The effective degrees of freedom are therefore approximately equal to the number of eigenvalues which are greater than the noise variance. If σ_ψ^2 is small, the system uses all degrees of freedom to fit the data. For a large σ_ψ^2, $P_{eff}^{Data} \to 0$ and the data barely influence the solution (see Figure 1).

We can therefore conclude that only P_{eff}^{Data} query points are necessary to fix the relevant degrees of freedom of the system. We have to select the query points such that they fix these degrees of freedom.[6]

Independently, Ferrari-Trecate, Williams and Opper (1999) made similar observation. They found that for small amounts of data the coefficients of the eigenfunctions with small eigenvalues are not well determined by the data, and thus can effectively be ignored.

[6]An impractical solution would be to include all training data in the set of query data, i.e. $f^m \subseteq f^q$. Although this would guarantee the conditional independence of the measurements, the set of query points would obviously be too large."

5.2 Dependency Graph

Another way of looking at the issue is to use a dependency graph which will provide us with a more general view on the BCM. Consider again the case that a set of past observations D is partitioned into M smaller data sets $D = \{D^1, \ldots, D^M\}$. In a typical statistical model the data sets are independent given all the parameters in a model w. Furthermore a future prediction f^* is independent of the training data sets given the parameters. The corresponding dependency graph is shown in Figure 2 (left). Let's assume that there is another (typically vectorial) variable f^q which is independent of f^* and the data sets given w. In the previous discussion we always assumed that $f^* \in f^q$ but in Section 7 we will also be interested in the case that this is not the case. If we now consider a probabilistic model without w, i.e. if we marginalize out the parameters w, then f^*, f^q and all the data sets will in general become fully dependent. On the other hand, if we can assume that w is almost uniquely determined by f^q the approximation that given f^q the data sets and f^* are all mutually independent might still be reasonable (Figure 2, right).

Explicitly, we assume that, first,

$$P(D^i|f^q, f^*, D \setminus D^i) \approx P(D^i|f^q)$$

which means that if f^q is fixed, the different data sets are mutually independent and also independent of f^*. One case where this is true is if w is deterministically determined function by f^q. Secondly we assume that

$$P(f^*|D, f^q) \approx P(f^*|f^q).$$

This says that given f^q, D does not provide you with additional information about f^*. This is guaranteed for example if $f^* \in f^q$ as in Section 2 or if —as before— w is deterministically determined by f^q.[7] Under these two assumptions

$$\hat{P}(f^q|D) = \frac{1}{D} P(f^q) \prod_{i=1}^{M} P(D^i|f^q) \propto \frac{1}{P(f^q)^{M-1}} \prod_{i=1}^{M} P(f^q|D^i) \tag{14}$$

and

$$\hat{P}(f^*|D) = \int P(f^*|f^q) \hat{P}(f^q|D) \, df^q. \tag{15}$$

Equation 14 is the basis for the BCM (compare Equation 3). Equation 15 shows how a new prediction of f^* can be derived from $\hat{P}(f^q|D)$. The dependency structure can experimentally be determined for Gaussian process regression by analyzing the inverse covariance structure (Appendix 10.3).

6 The Idea Behind the BCM Applied to a Frequentist Setting

So far we have developed the BCM in a Bayesian setting. In this section we apply the basic idea behind the BCM to a frequentist setting. The basic difference is that here we have to form expectations over data rather than parameters.

[7] In Section 7.1 on online learning we will also consider the case where $f^* \notin f^q$.

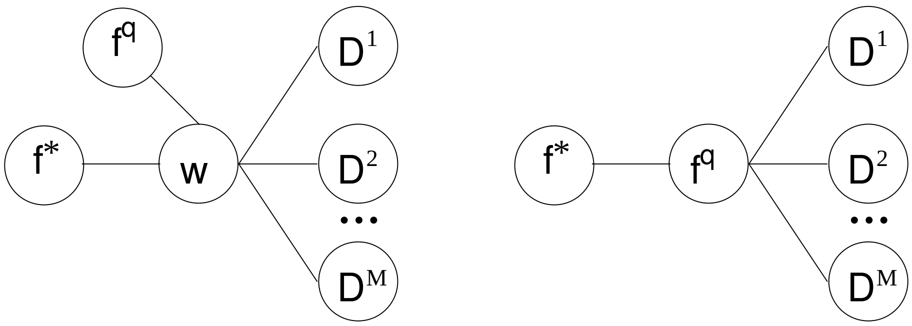

Figure 2: On the left we show the dependency structure of parameterized Bayesian learning. On the right we show the approximate dependency structure we assume for the BCM.

In Tresp and Taniguchi (1995) and Taniguchi and Tresp (1997) an input dependent weighting scheme was introduced which takes into account the input dependent certainty of an estimator. This approach is now extended to include the ideas developed in this paper. In contrast to the previous work, we allow that estimates at other query points can be used for improving the estimate at a given point x. As before, we use N_Q query points. Let there be M estimators and let $\hat{f}_i(x)$ denote the estimate of the $i-$th estimator at input x. Let $\hat{f}_i^q = (\hat{f}_i(x_1^q), \ldots, \hat{f}_i(x_{N_Q}^q))'$ denote the vector of responses of the $i-$th estimator to all query points and let $\hat{f}^q = ((\hat{f}_1^q)', \ldots, (\hat{f}_M^q)')'$ denote the vector of all responses of all estimators. We assume that all estimators are unbiased. Then our goal is to form a linear combination of the predictions at the query points

$$\hat{f}^q_{comb} = A\hat{f}^q$$

where A is a $N_Q \times (N_Q M)$ matrix of unknown coefficients such that the expected error

$$E[(t^q - A\hat{f}^q)'(t^q - A\hat{f}^q)]$$

is minimum where t^q is the (unknown) vector of true expectations at the query points. Using the bias-variance decomposition for linearly combined systems we obtain[8]

$$E[(t^q - A\hat{f}^q)'(t^q - A\hat{f}^q)]$$

$$= E\left[\left(E(A\hat{f}^q) - A\hat{f}^q\right)' \left(E(A\hat{f}^q) - A\hat{f}^q\right)\right] + E\left[\left(E(A\hat{f}^q) - t^q\right)' \left(E(A\hat{f}^q) - t^q\right)\right]$$

$$= trace(A\Omega A') + (AE[\hat{f}^q] - t^q)'(AE[\hat{f}^q] - t^q).$$

The term in the trace is the covariance of the combined estimator and the second term is the bias of the combined estimator and

$$(\Omega)_{ij} = cov\left((\hat{f}^q)_i, (\hat{f}^q)_j\right).$$

[8]In his section expectations are taken w.r.t. different data sets.

Note, that at this point we include the case that different estimators are correlated. We want to enforce the constraint that the combined system is unbiased. This can be enforced if we require that

$$A\mathcal{I} = I^{N_Q}$$

where I^{N_Q} is a N_Q-dimensional unit matrix and $\mathcal{I} = (I^{N_Q}, \ldots, I^{N_Q})'$ concatenates M unit matrices. Under these constraints it can easily be shown that $E(\hat{f}^q_{comb}) = AE[\hat{f}^q] = t^q$. Using Lagrange multipliers to enforce the constraint, we obtain for the optimal coefficients

$$A = (\mathcal{I}'\Omega^{-1}\mathcal{I})^{-1}\mathcal{I}'\Omega^{-1}.$$

This is the most general result allowing for both correlations between estimators and between query points. In our case we assume that the individual estimators were trained on different data sets which means that their respective prediction errors are uncorrelated. In this case

$$\Omega = \begin{pmatrix} cov_1^{qq} & 0 & \ldots & 0 \\ 0 & cov_2^{qq} & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & cov_M^{qq} \end{pmatrix}$$

where cov_i^{qq} is the covariance matrix of the query points for the $i-$th estimator. Then

$$\hat{f}^q_{comb} = \left(\sum_{j=1}^{M}(cov_j^{qq})^{-1}\right)^{-1}\left(\sum_{i=1}^{M}(cov_i^{qq})^{-1}\hat{f}_i^q\right). \tag{16}$$

Note, the similarity to Equation 4 if $(\Sigma^{qq})^{-1} \rightarrow 0$, i.e. if we have a very "weak" prior. Note again that the covariance matrices are w.r.t. responses in the same *estimator* and not between estimators. For linear systems with fixed basis functions, cov_j^{qq} can be calculated using well known formulas from linear regression (Sen and Srivastava, 1990). In Tresp and Taniguchi, 1995 and Taniguchi and Tresp, 1997 it is described how the covariance of the prediction of estimators can be calculated for nonlinear systems.

7 Extensions

7.1 Online BCM

In this section we derive an online version of the BCM. We assume that data arrive continuously in time. Let D^k denote the data points collected between time $t(k)$ and $t(k-1)$ and let $\mathcal{D}^k = \{D^1, \ldots, D^k\}$ denote the set of all data collected up to time $t(k)$.

We can derive an algorithm which directly adapts the posterior probabilities of f^Q. The iterations are based on the Kalman filter and yields for $k = 1, 2, \ldots$

$$A_k = A_{k-1} + K_k(E(f^q|D^k) - A_{k-1})$$

$$K_k = S_{k-1}(S_{k-1} + cov(f^q|D^k))^{-1}$$

$S_k = S_{k-1} - K_k(S_{k-1} + cov(f^q|D^k))K_k'$

with $S_0 = \Sigma^{qq}$ and $A_0 = (0, \ldots, 0)'$. At any time

$\hat{E}(f^q|\mathcal{D}^k) = \frac{1}{k}\Sigma^{qq}(\frac{1}{k}\Sigma^{qq} - S_k)^{-1}A_k.$

Note that only one matrix and one vector of the size of the number of query points need to be stored which makes this online version of the BCM applicable for data mining applications. If N_Q is large, the Kalman filter described in Appendix 10.2 might be more suitable.

7.2 Varying Query Points

In some cases, particularly in online learning, we might be interested in the responses at additional query points. The goal is then to infer from the estimated density at the query points f^q the density at other query points f^*. Using the approximation in Section 5, $\hat{P}(f^*|D)$ is Gaussian distributed with

$$\hat{E}(f^*|D) = \Sigma^{*q}(\Sigma^{qq})^{-1}\hat{E}(f^q|\mathcal{D}^k). \tag{17}$$

Note, that this equation describes the superposition of basis functions defined for every query point f^q and evaluated at f^* with weights $(\Sigma^{qq})^{-1}\hat{E}(f^q|\mathcal{D}^k)$. Note also, that these operations are relatively inexpensive. Furthermore the covariance becomes

$$\widehat{cov}(f^*|D) = \Sigma^{**} - \Sigma^{*q}(\Sigma^{qq})^{-1}(\Sigma^{*q})' + \Sigma^{*q}(\Sigma^{qq})^{-1}\ \widehat{cov}(f^q|D)\ (\Sigma^{*q}(\Sigma^{qq})^{-1})'.$$

The covariance of the prediction is small if the covariance $\widehat{cov}(f^q|D)$ has small elements and if f^* can be well determined from f^q.

8 Experiments

In the experiments we set for two locations in input space x_i and x_j: $\sigma_{x_i,x_j} = A\exp(-1/(2\gamma^2)||x_i - x_j||^2)$ where A, γ are positive constants. In the first experiments we used an artificial data set. The input space is $Dim = 5-$dimensional and the inputs were randomly chosen in $x \in [-1, 1]^{Dim}$. Targets were generated by adding independent Gaussian noise with a standard deviation of 0.1 to a map defined by 5 normalized Gaussian basis functions (see Appendix 10.4).

The width parameter γ and the 'noise to prior' - factor σ_ψ^2/A were optimized using a validation set of size 100 and were then left fixed for all experiments. Alternative approaches to estimate (or to integrate out) these hyperparameters are described by Gibbs and MacKay (1997), Neal (1996) and Williams and Rasmussen (1996).

Figure 3 (left) shows the CPU-time for the different algorithms as a function of the number of training samples. Clearly, the CPU-time for Gaussian process regression scales as K^3. The two versions of the BCM schemes scale linearly. Also note that for $K/M = 100 \approx K/N_Q$ (in the experiment, $N_Q = 128$) we obtain the least computational cost.

Figures 3 (right) shows the mean squared query-set error for simple Gaussian process regression and for the BCM. As can be seen, the BCM is not significantly worse than the Gaussian

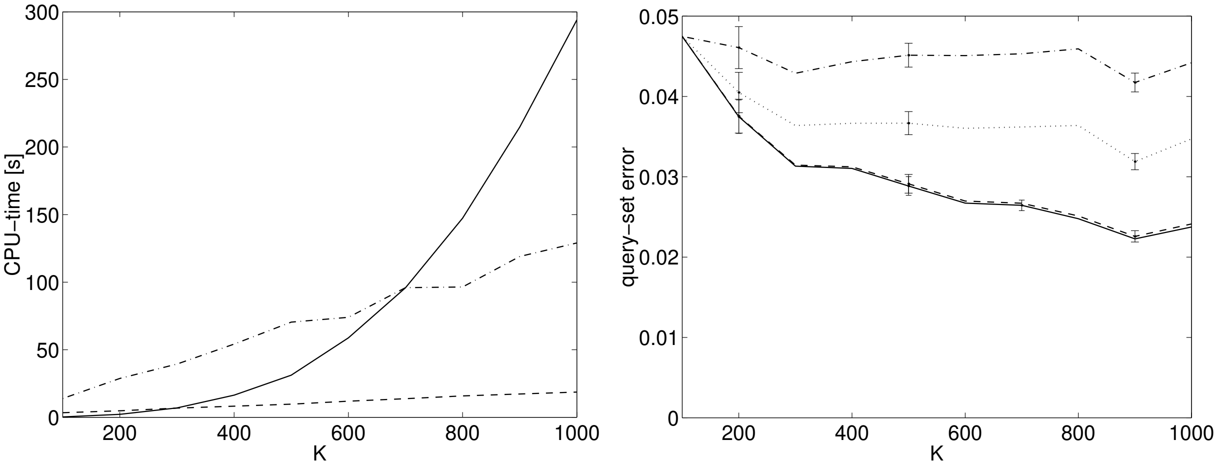

Figure 3: Left: CPU-time in seconds as a function of the training set size K for $M = 1$ (Gaussian process regression, continuous) and the BCM with module size $K/M = 10$ (dash-dotted) and module size $K/M = 100$ (dashed). In these experiments the number of query points was $N_Q = 128$.
Right: The mean squared query-set error as a function of the number of training data. The continuous line shows the performance of the simple Gaussian process regression system with only one module. For the other experiments the training data set is divided into $K/100$ modules of size 100. Shown are the average errors of the individual modules (dash-dotted), the error achieved by averaging the predictions of the modules (dotted), and the error of the BCM (dashed). Note that the error of the BCM is not significantly worse than the error of simple Gaussian process regression.

process regression system with only one module. Note, that the average error of the individual modules is considerably worse than the BCM solution and that simple averaging of the predictions of the modules is a reasonable approach but considerably worse than the BCM.

Figure 4 (left) shows the mean squared query-set error of the BCM as a function of the query set size N_Q. As confirmed by the theory, for small N_Q, the performance deteriorates. We obtain better results when we query more at the same time!

In the second set of experiments we tested the performance using a number of real and artificial data sets. Here we found it more useful to normalize the error. The relative error is defined as $mse_r^a = (mse^a - mse^f)/mse^f$ where mse^a is the mean squared query-set error of algorithm a and mse^f is the mean squared query-set error of Gaussian process regression ($M = 1$).

Table 1 and Table 2 summarize the results. Well demonstrated is the excellent performance of the BCM. The results show that P_{eff}^{Data} is indeed a good indication of how many query points are required: if the number of query points is larger than P_{eff}^{Data}, the performance of the BCM solution is hardly distinguishable from the performance of Gaussian process regression with only one module.

For the real world data (Table 1) we obtain excellent performance for the BUPA and the DIABETES data sets with a P_{eff}^{Data} of 16 and 64, respectively, and slightly worse performance for the WAVEFORM and the HOUSING data with a P_{eff}^{Data} of 223 and 138, respectively. A possible explanation for the slightly worse performance for the Boston housing data set might be the well known heteroscedasticity (unequal target noise variance) exhibited by this data set.

For the artificial data set (Table 2) it can be seen that a large P_{eff}^{Data} is obtained for a data set with no noise and a high dimensional data set with 50 inputs. In contrast a small P_{eff}^{Data} is obtained for a noisy data set and for a low dimensional data set.

In our third experiment (Figure 4, right) we tested the online BCM described in Section 7.1. Note, that the BCM with $K = 60000$ achieves results unobtainable with Gaussian process regression which is limited to a training data set of approximately $K = 1000$. Considering the excellent results obtained by the BCM for $K = 60000$ as displayed in the figure one might ask how close the BCM solution comes to a simple Gaussian process regression system for such a large data set. We expect that the latter should give better results since the width parameter γ can be optimized for the large training data set of size $K = 60000$. Typically, the optimal γ decreases with K (Hastie and Tibshirani, 1990). For the BCM, γ is chosen such that the effective degrees of freedom approximately matches the number of query points N_Q so the optimal γ should be more or less be independent of K. Experimentally, we found that scaling γ down with K also improves performance for the BCM but that γ cannot be scaled down as much as in case of the simple Gaussian regression model. But recall that for $K = 60000$, the simple Gaussian process regression system would require one year of CPU-time.

9 Conclusions

We have introduced the BCM and have applied it to Gaussian process regression and to systems with fixed basis functions. We found experimentally that the BCM provides excellent predictions on several data sets. The CPU-time of the BCM scales linearly in the number of training data

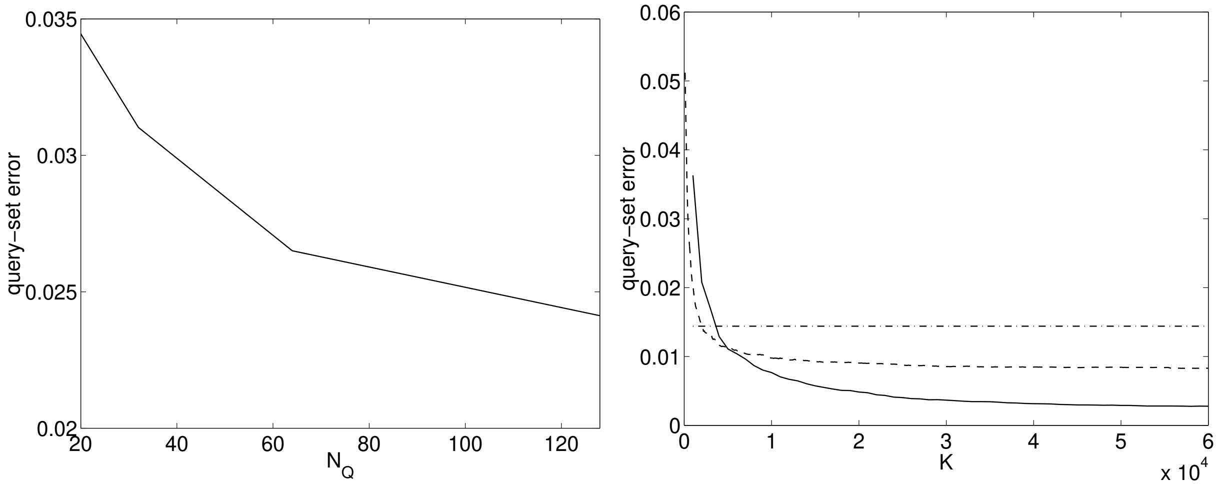

Figure 4: Left: The mean squared query-set error as a function of the number of query points N_Q for a module size $K/M = 100$ and with $K = 1000$. Note that for small N_Q, the performance deteriorates.
Right: The test of the online learning algorithms of Section 7.1 using the artificial data set. The continuous line is the mean squared query-set error for $N_Q = 1000$ and a module size of $K/M = 1000$ as a function of the number of training data K. The width parameter γ and the 'noise to prior' - factor σ_ψ^2/A were optimized for the large data set ($K = 60000$). The dash dotted line shows the error of Gaussian process regression (M=1) with $K = 1000$, approximately the largest training data size suitable for Gaussian process regression. Here, the width parameter γ and the 'noise to prior' - factor σ_ψ^2/A were optimized using a validation set of size $K = 1000$. Note the great improvement with the BCM which can employ the larger data sets. The dashed line is the performance of the BCM with $N_Q = 100$. Note, that the full benefit of the larger data set is only exploited by the larger N_Q. For $K = 60000$ Gaussian process regression would need an estimated one year of CPU-time using a typical workstation, whereas the BCM with $N_Q = 100$ only required a few minutes and with $N_Q = 1000$ a few hours.

Table 1: The table shows results from real world data. These data sets can be retrieved from the UCI data base at http://www.ics.uci.edu/ mlearn/MLRepository.html. All data sets were normalized to a mean of zero and a standard deviation of one. Shown is the input dimension Dim, the number of training data used K, and the mean square query-set error of various algorithms. Shown are averages over 20 experiments from which error bars could be derived. In each experiment, data were assigned into training and query data randomly. The width parameter γ and the 'noise to prior' - factor σ_ψ^2/A were optimized using a validation set of size 100. The first result (GPR) shows the mean squared query-set error of Gaussian process regression (M=1). The other rows show *relative* errors (see main text). BCM(i, j) shows the relative error with module size $K/M = i$ and query-set size $N_Q = j$. Also shown is the rank of Σ^{mm} (calculated as described in Figure 1) and the effective number of parameters.

	BUPA	DIABETES	WAVEFORM	HOUSING
Dim	6	8	21	13
K (train. size)	200	600	600	400
GPR	0.828 ± 0.028	0.681 ± 0.032	0.379 ± 0.019	0.1075 ± 0.0065
BCM(10, 50)	$(-2.6 \pm 1.8) \times 10^{-4}$	0.0095 ± 0.0030	0.0566 ± 0.0243	0.338 ± 0.132
BCM(100, 50)	$(1.6 \pm 1.8) \times 10^{-4}$	-0.0027 ± 0.0023	0.0295 ± 0.0213	0.117 ± 0.038
BCM(10, 100)	$(0.1 \pm 0.2) \times 10^{-4}$	-0.0001 ± 0.0015	0.0037 ± 0.0095	0.196 ± 0.053
BCM(100,100)	$(0.0 \pm 0.1) \times 10^{-4}$	-0.0011 ± 0.0010	0.0163 ± 0.0086	0.1138 ± 0.0568
$rank$	159	600	600	400
P_{eff}^{Data}	16	43	223	138

Table 2: Same as in Table 1 but with artificial data. The data are generated as described in the beginning of Section 8. As before, the width parameter γ and the 'noise to prior' - factor σ_ψ^2/A were optimized using a validation set of size 100. In columns from left to right, the experiments are characterized as: no noise $\sigma_\psi = 0$, large noise $\sigma_\psi = 0.5$, high input dimension $Dim = 50$ and low input dimension $Dim = 2$. The large relative error for the 'no noise' experiment must be seen in connection with the small absolute error.

	ART ($\sigma_\psi = 0$)	ART ($\sigma_\psi = .5$)	ART ($\sigma_\psi = 0.1$)	ART ($\sigma_\psi = 0.1$)
Dim	5	5	50	2
K	600	600	600	600
GPR	0.0185 ± 0.0009	0.0529 ± 0.0026	0.0360 ± 0.0017	0.0022 ± 0.00012
BCM(10, 50)	0.3848 ± 0.0442	0.0021 ± 0.0013	0.4135 ± 0.0860	0.0809 ± 0.0221
BCM(100, 50)	0.3066 ± 0.0282	0.0011 ± 0.0012	0.0994 ± 0.0541	0.0928 ± 0.0157
BCM(10, 100)	0.1574 ± 0.0135	0.0001 ± 0.0000	-0.0923 ± 0.0181	-0.0007 ± 0.0008
BCM(100, 100)	0.0595 ± 0.0114	0.0000 ± 0.0000	-0.0527 ± 0.0166	0.0002 ± 0.0011
$rank$	600	600	600	242
P_{eff}^{Data}	121	22	600	49

and the BMC can therefore be used in applications with large data sets as in data mining and in applications requiring online learning. May be the most surprising result is that the quality of the prediction of the BCM improves if several query points are calculated at the same time.

Acknowledgements

Fruitful discussions with Harald Steck, Michael Haft, Reimar Hofmann and Thomas Briegel and the comments of two anonymous reviewers are greatfully acknowledged.

10 Appendix

10.1 Product and Quotient of Gaussian Densities

For the product of two Gaussian densities we obtain

$$G(x;c_1,\Sigma_1)G(x;c_2,\Sigma_2) \propto G(x; \left(\Sigma_1^{-1}+\Sigma_2^{-1})^{-1}(\Sigma_1^{-1}c_1+\Sigma_2^{-1}c_2), (\Sigma_1^{-1}+\Sigma_2^{-1})^{-1}\right)$$

and for the quotient of two Gaussian densities we obtain

$$G(x;c_1,\Sigma_1)/G(x;c_2,\Sigma_2) \propto G\left(x; (\Sigma_1^{-1}-\Sigma_2^{-1})^{-1}(\Sigma_1^{-1}c_1-\Sigma_2^{-1}c_2), (\Sigma_1^{-1}-\Sigma_2^{-1})^{-1}\right).$$

$G(x;c,\Sigma)$ is our notation for a normal density function with mean c and covariance Σ evaluated at x.

Substituting Gaussian densities in Equation 3 we obtain

$$\hat{P}(f^q|D) = const \times \frac{1}{G[f^q;0,\Sigma^{qq}]^{M-1}} \prod_{i=1}^{M} G[f^q; E(f^q|D^i), cov(f^q|D^i)]$$

$$= const \times \frac{1}{G[f^q;0,\Sigma^{qq}/(M-1)]} \times$$

$$G\left[f^q; \left(\sum_{i=1}^{M} cov(f^q|D^i)^{-1}\right)^{-1} \left(\sum_{i=1}^{M} cov(f^q|D^i)^{-1}E(f^q|D^i)\right), \left(\sum_{i=1}^{M} cov(f^q|D^i)^{-1}\right)^{-1}\right]$$

$$= const \times G\Big[f^q; \left(-(M-1)(\Sigma^{qq})^{-1} + \sum_{i=1}^{M} cov(f^q|D^i)^{-1}\right)^{-1} \left(\sum_{i=1}^{M} cov(f^q|D^i)^{-1}E(f^q|D^i)\right),$$

$$\left(-(M-1)(\Sigma^{qq})^{-1} + \sum_{i=1}^{M} cov(f^q|D^i)^{-1}\right)^{-1}\Big]$$

such that Equations 4 and 5 follow.

10.2 Another Formulation of the BCM

Based on our Gaussian assumptions we can also calculate the probability density of measurements given the query points $P(g^m|f^q)$. This density is also Gaussian with mean

$$E(g^m|f^q) = (\Sigma^{qm})'(\Sigma^{qq})^{-1}f^q, \tag{18}$$

and covariance

$$cov(g^m|f^q) = \Psi^{mm} + \Sigma^{mm} - (\Sigma^{qm})'(\Sigma^{qq})^{-1}\Sigma^{qm}. \tag{19}$$

Now, the BCM approximation can also be written as

$$\hat{P}(f^q|D) \propto P(f^q)\prod_{i=1}^{M} P(g_i^m|f^q).$$

Then we obtain with $A_i = (\Sigma^{qm,i})'(\Sigma^{qq})^{-1}$

$$\hat{E}(f^q|D) = \frac{1}{C}\sum_{i=1}^{M} A_i' cov(g_i^m|f^q)^{-1} g_i^m \tag{20}$$

with

$$C = \widehat{cov}(f^q|D)^{-1} = (\Sigma^{qq})^{-1} + \sum_{i=1}^{M} A_i' cov(g_i^m|f^q)^{-1} A_i. \tag{21}$$

$\Sigma^{qm,i}$ is the the covariance matrix between training data for the i-th module and the query points. An online version can be obtained using the Kalman filter which yields the iterative set of equations, $k = 1, 2, \ldots$

$$\hat{E}(f^q|\mathcal{D}^k) = \hat{E}(f^q|\mathcal{D}^{k-1}) + K_k(g_k^m - A_k f^q)$$

$$K_k = (S_{k-1}A_k')(A_k S_{k-1} A_k' + cov(g_k^m|f^q))^{-1}$$

$$S_k = S_{k-1} - K_k(A_k S_{k-1} A_k' + cov(g_k^m|f^q))K_k'$$

with $S_0 = \Sigma^{qq}$. In the iterations, no matrix of size N_Q needs to be inverted.

10.3 Independencies in the Inverse Covariance Matrix

Figure 5 (left) shows the absolute values of the covariance matrix for $K = 300$ training data points (indices 1 to 300) and $N_Q = 100$ query points (indices 301 to 400). The covariance matrix is calculated as

$$\begin{pmatrix} \Sigma^{mm} + \Psi^{mm} & (\Sigma^{qm})' \\ \Sigma^{qm} & \Sigma^{qq} \end{pmatrix}.$$

It is apparent that there are strong correlations in all dimensions. Figure 5 (right) shows the absolute values of the inverse covariance matrix. Here, white pixels indicate strong dependencies. Apparent are the strong dependencies among the query points. Also apparent is the fact that there are strong dependencies between the training data and the query data but not in between the training data (given the query data) confirming that the approximation in Equation 1 is reasonable.

Readers not familiar with the relationship between the inverse covariance matrix and independencies are referred to the book by Whittaker (1990).

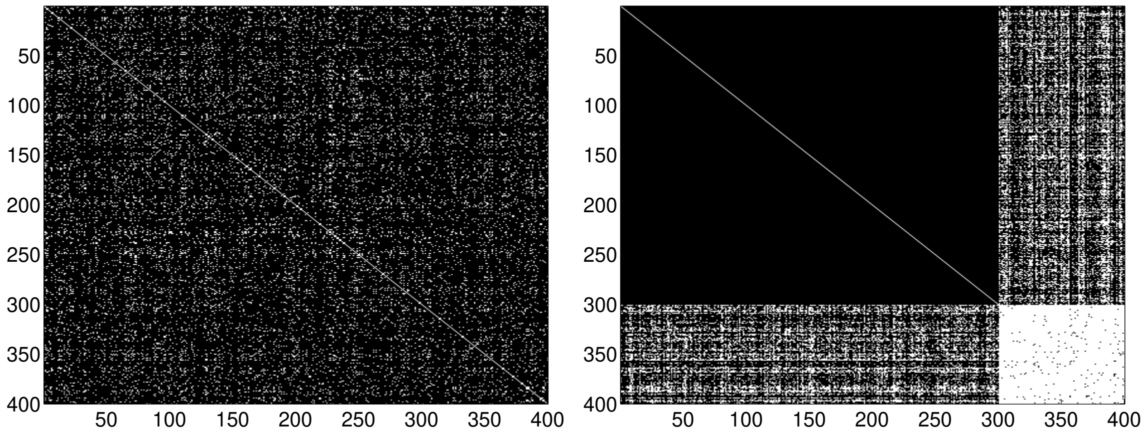

Figure 5: Left: Absolute values of the covariance matrix. White pixels are entries larger than 0.8. Right: Absolute values of the inverse covariance matrix. White pixels are larger than 1.

10.4 Artificial Data

Explicitly, the response is calculated as

$$f(x) = \frac{\sum_{i=1}^{5} a_i \exp\left(-\frac{||x-center_i||^2}{2\sigma_a^2}\right)}{\sum_{i=1}^{5} \exp\left(-\frac{||x-center_i||^2}{2\sigma_a^2}\right)}.$$

In most experiments

$$\sigma_a = 0.36 \quad a = (1.16, 0.63, 0.08, 0.35, -0.70)'$$

and $center_i$ is generated randomly according to a uniform density in the Dim-dimensional unit hypercube.

References

Bernardo, J., M., and Smith, A. F. M. (1994). *Bayesian Theory.* Wiley.

Ferrari-Trecate, G., Williams, C. K. I and 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,* MIT Press, Cambridge MA.

Gibbs, M. and MacKay, D. J. C. (1997). Efficient implementation of Gaussian processes. Available from http://wol.ra.phy.cam.ac.uk/mackay/homepage.html.

Girosi, F. (1998). An equivalence between sparse approximation and support vector machines. *Neural Computation*, vol. 10, No. 6.

Hastie, T. and Tibshirani R. J. (1990). *Generalized additive models.* Chapman & Hall.

MacKay, D. J. C. (1992). Bayesian model comparison and backprop nets. In J. E. Moody, S. J. Hanson, and R. P. Lippmann (Eds.), *Advances in neural information processing systems, 4.* Morgan Kaufmann, San Mateo.

MacKay, D. J. C. (1998). Introduction to Gaussian processes. In Bishop, C., M., (Ed.), *Neural Networks and Machine Learning*, NATO Asi Series. Series F, Computer and Systems Sciences, Vol 168.

Moody, J. E. (1992). The effective number of parameters: an analysis of generalization and regularization in nonlinear learning systems. In J. E. Moody, S. J. Hanson, and R. P. Lippmann (Eds.), *Advances in neural information processing systems, 4.* Morgan Kaufmann, San Mateo, pp. 847-854.

Moody, J. E., and Rögnvaldsson, T. S. (1997). Smoothing regularizers for projective basis function networks. In M. C. Mozer, M. I. Jordan, and T. Petsche (Eds.), *Advances in neural information processing systems, 9.* MIT Press, Cambridge MA.

Neal, R. M. (1996) *Bayesian learning for neural networks.* New York: Springer Verlag.

Neal, R. M. (1997) *Monte Carlo implementation of Gaussian process models for Bayesian regression and classification.* Tech. Rep. No. 9702, Department of Statistics, University of Toronto.

Poggio, T., and Girosi, F. (1990). Networks for approximation and learning. *Proceedings of the IEEE, 78,* pp. 1481-1497.

Poggio, T., and Girosi, F. (1998). A sparse representation for function approximation. *Neural Computation*, vol. 10, No. 6.

Rasmussen, C. E. (1996). *Evaluation of Gaussian processes and other methods for non-linear regression.* Unpublished Ph.D. dissertation, Department of Computer Science, University of Toronto. Available from http:// www.cs.utoronto.ca/ carl/.

Skilling, J. (1993). Bayesian numerical analysis. In *Physics and Probability*, W. T. Grandy and P. Milonni., eds., C. U. P.

Smola, A. J., Schölkopf, B. and Müller, K.-R. (1998). The connection between regularization operators and support vector kernels. *Neural Networks,* Vol. 11, pp. 637–649.

Sen, A. and Srivastava, M. (1990). *Regression Analysis.* Springer Verlag, New York.

Taniguchi, M. and Tresp, V. (1997). Averaging regularized estimators, *Neural Computation,* Vol. 9, Nr. 5.

Tresp, V. and Taniguchi, M. (1995). Combining estimators using non-constant weighting functions, in G. Tesauro, D. S. Touretzky and T. K. Leen, eds., *Advances in Neural Information Processing Systems 7,* MIT Press, Cambridge MA. pp. 419-426.

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

Wahba, G. (1983). Bayesian "confidence intervals" for the cross-validated smoothing spline. *J.*

Roy. Stat. Soc. Ser. B, Vol. 10, pp. 133-150.

Wahba, G. (1990). *Spline models for observational data.* Philadelphia: Society for Industrial and Applied Mathematics.

Whittaker, J. (1990). *Graphical Models in Applied Multivariate Statistics.* Wiley.

Williams, C. K. I. and Rasmussen, C. E. (1996). Gaussian processes for regression. In D. S: Touretzki, M. C. Mozer, and M. E. Hasselmo (Eds.), *Advances in Neural Processing systems, 8.* Cambridge, MA: MIT Press, pp. 514-520.

Williams, C. K. I. (1997). Computing with infinite networks. In M. C. Mozer, M. I. Jordan, and T. Petsche (Eds.), *Advances in neural information processing systems, 9.* Cambridge, MA: MIT Press.

Williams, C. K. I. (1998a). Computing with infinite neural networks. *Neural Computation, 10*, pp. 1203-1216.

Williams, C. K. I. (1998b). Prediction with Gaussian processes: from linear regression to linear prediction and beyond. In *Learning in Graphical Models*, Jordan, M. I., editor, Kluwer Academic, pp. 599-621.

Zhu, H., and Rohwer (1996). Bayesian regression filters and the issue of priors. *Neural Comp. Appl, 4*, 3, pp. 130-142.

Zhu, H., Williams, C. K. I., Rohwer, R., and Morciniec, M. (1998). Gaussian regression and optimal finite dimensional linear models. In Bishop, C., M., (Ed.), *Neural Networks and Machine Learning*, NATO Asi Series. Series F, Computer and Systems Sciences, Vol 168.