Efficient Implementation of Gaussian Processes

Mark Gibbs
Cavendish Laboratory
Cambridge CB3 0HE
United Kingdom

David J.C. MacKay
Cavendish Laboratory
Cambridge CB3 0HE
United Kingdom

May 28, 1997

Abstract

Neural networks and Bayesian inference provide a useful framework within which to solve regression problems. However their parameterization means that the Bayesian analysis of neural networks can be difficult. In this paper, we investigate a method for regression using Gaussian process priors which allows exact Bayesian analysis using matrix manipulations. We discuss the workings of the method in detail. We will also detail a range of mathematical and numerical techniques that are useful in applying Gaussian processes to general problems including efficient approximate matrix inversion methods developed by Skilling.

1 Introduction

Neural networks and Bayesian inference have provided a useful framework within which to solve regression problems (MacKay 1992a) (MacKay 1992b). However due to the parameterization of a neural network, implementations of the Bayesian analysis of a neural network require either maximum aposteriori approximations (MacKay 1992b) or the evaluation of integrals using Monte Carlo methods (Neal 1993).

In this paper, we will review a framework within which to solve regression problems using parameterized Gaussian processes (Cressie 1993; Williams 1995; Williams and Rasmussen 1996). This framework allows us to evaluate the integrals which occur in the associated Bayesian analysis exactly using matrix manipulations. Firstly we shall discuss the nature of the regression problem and how the Gaussian process framework can be used to solve it. We will also derive the principal formulae for Gaussian processes and describe how to estimate the hyperparameters of a Gaussian process from the data. We will then describe numerical techniques which can be used to implement the framework and discuss the performance of these techniques for varying amounts of data. Finally we shall give two examples to demonstrate some of the features of Gaussian processes.

Software implementing the framework described in this paper is freely available on the World Wide Web at www.wol.ra.phy.cam.ac.uk/mng10/GP/GP.html.

2 Bayesian Modelling

Let us briefly summarize the problem we wish to solve. We have some noisy data $\mathcal{D}$ which consists of N pairs of L-dimensional input vectors and scalar outputs $\{\mathbf{x}_n, t_n\}_{n=1}^{N}$. We wish to use the data to make predictions at new points, i.e, we wish to find the predictive distribution of t_{N+1} at a point $\mathbf{x}_{N+1} \notin \mathcal{D}$.

We can define a prior over the space of possible functions to model the data $P(f|\alpha)$ where α is some set of hyperparameters. We can also define a prior over the noise $P(\nu|\beta)$ where ν is some appropriate noise vector and β is a set of hyperparameters. We can then write down the probability of the data given these hyperparameters

$$P(\mathbf{t}_N|\{\mathbf{x}_n\},\alpha,\beta) = \int df d\nu \; P(\mathbf{t}_N|\{\mathbf{x}_n\},f,\nu)P(f|\alpha)P(\nu|\beta) \tag{1}$$

where $\mathbf{t}_N = (t_1, t_2, \cdots, t_N)$. Now if we define the vector $\mathbf{t}_{N+1} = (t_1, \cdots, t_N, t_{N+1})$ then we can write down the conditional distribution of t_{N+1}.

$$P(t_{N+1}|\mathcal{D},\alpha,\beta) = \frac{P(\mathbf{t}_{N+1}|\{\mathbf{x}_n\},\alpha,\beta)}{P(\mathbf{t}_N|\{\mathbf{x}_n\},\alpha,\beta)} \tag{2}$$

and hence we can use this conditional distribution to make predictions about t_{N+1}. Please note the notational difference between t_{N+1} (the prediction at $\mathbf{x}_{N+1}$) and the $\mathbf{t}_{N+1}$ (the vector constructed from $\mathbf{t}$ and t_{N+1}).

In general the integration in equation 1 is complicated. For example in neural networks with one or more hidden layers and with a finite number of neurons, the form of $f(\mathbf{x})$ is such that placing simple priors over the weights creates complicated priors over $f(\mathbf{x})$. The standard approach to solving such problems has been either to make approximations in order to evaluate equation 1 (MacKay (1992a)'s Evidence framework) or to evaluate the integral numerically using Monte Carlo methods (Neal 1993). We shall now consider a new approach based on Gaussian process priors which gives us an exact analytic form for equation 1 and allows us to perform exact Bayesian analysis using matrix manipulations.

2.1 The Gaussian Process Model

A Gaussian process is a collection of variables $\mathbf{t} = (t(\mathbf{x}_1), t(\mathbf{x}_2), \cdots)$ which have a joint distribution

$$P(\mathbf{t}|\mathbf{C},\{\mathbf{x}_n\}) = \frac{1}{Z}\exp\left(-\frac{1}{2}(\mathbf{t}-\mu)^T\mathbf{C}^{-1}(\mathbf{t}-\mu)\right) \tag{3}$$

for any $\{\mathbf{x}_n\}$ where $C_{mn} = C(\mathbf{x}_m, \mathbf{x}_n; \Theta)$ is a parameterized covariance function with hyperparameters Θ and μ is the mean vector. The parameters of a Gaussian process are referred to as hyperparameters due to their close relationship to the hyperparameters of a neural network (see Section 2.4.2). Let us define our data vector $\mathbf{t}_N$ to be a Gaussian process as in equation 3 with a covariance matrix $\mathbf{C}_N$ and a mean vector $\mu = \mathbf{0}$. By doing this we have bypassed the step of expressing individual priors on the noise and the modelling function by combining both priors into the covariance matrix $\mathbf{C}_N$. We shall deal with the exact form of $\mathbf{C}_N$ in the next section.

Using equation 3, the conditional Gaussian distribution over t_{N+1} can be written as

$$\begin{aligned} P(t_{N+1}|\mathcal{D}, C(\mathbf{x}_n,\mathbf{x}_m;\Theta), \mathbf{x}_{N+1}, \Theta) &= \frac{P(\mathbf{t}_{N+1}|C(\mathbf{x}_n,\mathbf{x}_m;\Theta),\Theta,\mathbf{x}_{N+1},\{\mathbf{x}_n\})}{P(\mathbf{t}_N|C(\mathbf{x}_n,\mathbf{x}_m;\Theta),\Theta,\{\mathbf{x}_n\})} && (4)\\ &= \frac{Z_N}{Z_{N+1}}\exp\left[-\frac{1}{2}\left(\mathbf{t}_{N+1}^T\mathbf{C}_{N+1}^{-1}\mathbf{t}_{N+1} - \mathbf{t}_N^T\mathbf{C}_N^{-1}\mathbf{t}_N\right)\right] && (5)\end{aligned}$$

where Z_N and Z_{N+1} are appropriate normalising constants. Figure 1 shows the relationship between $\mathbf{C}_{N+1}$ and $\mathbf{C}_N$. Collecting together those terms that are functions of t_{N+1} in equation 5, we can derive the Gaussian distribution of t_{N+1} at $\mathbf{x}_{N+1}$.

$$P(t_{N+1}|\mathcal{D}, C(\mathbf{x}_n,\mathbf{x}_m;\Theta),\Theta,\mathbf{x}_{N+1}) = \frac{1}{Z}\exp\left(-\frac{\left(t_{N+1}-\hat{t}_{N+1}\right)^2}{2\sigma^2_{\hat{t}_{N+1}}}\right) \tag{6}$$

where

$$\begin{aligned} \hat{t}_{N+1} &= \mathbf{k}_{N+1}^T\mathbf{C}_N^{-1}\mathbf{t}_N && (7)\\ \sigma^2_{\hat{t}_{N+1}} &= \kappa - \mathbf{k}_{N+1}^T\mathbf{C}_N^{-1}\mathbf{k}_{N+1} && (8)\end{aligned}$$

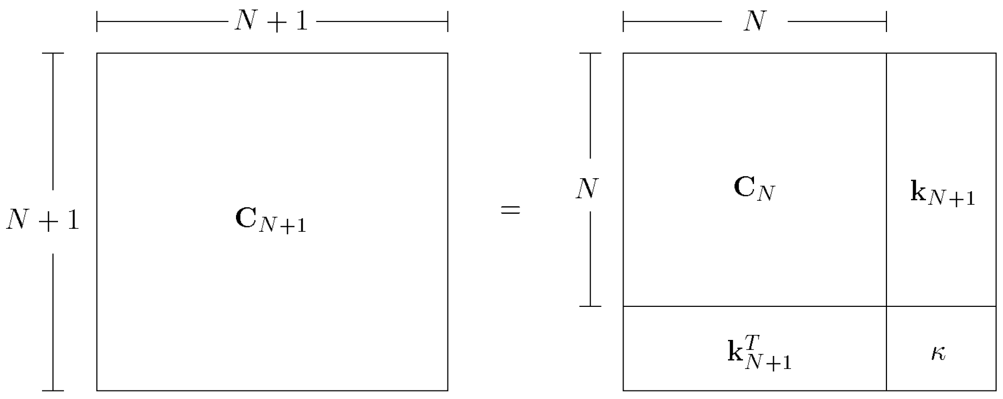


Figure 1: **Covariance Matrices :** The larger covariance matrix $\mathbf{C}_{N+1}$ is constructed from the smaller matrix $\mathbf{C}_N$, the vector $\mathbf{k}_{N+1} = (C(\mathbf{x}_1, \mathbf{x}_{N+1}; \Theta), \cdots, C(\mathbf{x}_N, \mathbf{x}_{N+1}; \Theta))$ and the scalar $\kappa = C(\mathbf{x}_{N+1}, \mathbf{x}_{N+1}; \Theta)$. This partitioned form is used later for calculating the inverse of $\mathbf{C}_{N+1}$ given $\mathbf{C}_N^{-1}$ (see Section 4.1).

with $\mathbf{k}_{N+1} = (C(\mathbf{x}_1, \mathbf{x}_{N+1}; \Theta), \cdots, C(\mathbf{x}_N, \mathbf{x}_{N+1}; \Theta))$ and $\kappa = C(\mathbf{x}_{N+1}, \mathbf{x}_{N+1}; \Theta)$. The predictive mean at the new point is given by $\hat{t}_{N+1}$ and $\sigma_{\hat{t}_{N+1}}$ are the error bars on this prediction. Notice that we do not need to invert $\mathbf{C}_{N+1}$ in order to make predictions at $\mathbf{x}_{N+1}$. Only $\mathbf{C}_N$ needs to be inverted.

2.2 The Covariance Function

We must now ask ourselves what should the form of the covariance function be and how the form of that function will affect the type of interpolant that we shall produce. The only constraint on our choice of covariance function is that it must generate a non-negative definite covariance matrix for any set of points $\{\mathbf{x}_n\}_{n=1}^N$. We choose the covariance function

$$C(\mathbf{x}_m, \mathbf{x}_n; \Theta) = \mathcal{C}_f(\mathbf{x}_m, \mathbf{x}_n; \Theta) + \delta_{mn}\mathcal{N}(\mathbf{x}_m; \Theta) \tag{9}$$

where $\mathcal{C}_f$ is associated with the form of the interpolant and $\mathcal{N}$ is associated with the noise model. One possible form for $\mathcal{C}_f$ is

$$\mathcal{C}_f(\mathbf{x}_m, \mathbf{x}_n; \Theta) = \theta_1 \exp\left\{ -\frac{1}{2}\sum_{l=1}^{L} \frac{\left(x_m^{(l)} - x_n^{(l)}\right)^2}{r_l^2} \right\} + \theta_2 \tag{10}$$

where $x_n^{(l)}$ is the l^{th} component of $\mathbf{x}_n$, an L dimensional vector, and $\theta_1, \theta_2, r_l \in \Theta$. The above expression embodies the property that points which are close in input space are strongly correlated and hence give rise to similar values of t. The definition of 'close' is expressed in terms of a set of length scales $\{r_l\}$. There is a length scale corresponding to each input which characterizes the distance in that particular direction over which t is expected to vary significantly. A very large length scale means that the predictions made using the Gaussian process would have little or no bearing on the corresponding input. Such an input could be said to be insignificant. This interpretation is closely related to Automatic Relevance Determination (MacKay 1994), (Neal 1996). The θ_1 hyperparameter gives the overall vertical scale relative to the mean of the Gaussian process in output space. The θ_2 hyperparameter gives the vertical uncertainty. This reflects how far we expect the true mean of the data to fluctuate from the mean of the Gaussian process.

The second term in the covariance function $\delta_{mn}\mathcal{N}$ represents the noise model. We expect the noise to be random and so do not expect any correlations between the noise on individual outputs. Hence the second term only contributes to the diagonal elements of the covariance matrix. The noise variance $\mathcal{N}(\mathbf{x}_n;\Theta)$ can have the form

$$\mathcal{N}(\mathbf{x}_n;\Theta) = \begin{cases} \theta_3 & \text{for input-independent noise} \\ \exp\left(\sum_{j=1}^{J}\beta_j\phi_j(\mathbf{x}_n)\right) & \text{for input-dependent noise} \end{cases} \tag{11}$$

where $\theta_3, \beta_j \in \Theta$. For noise that is not a function of the inputs we use a single hyperparameter θ_3 to describe its variance. However if we have input-dependent noise we can describe its variance using a set of basis functions $\{\phi_j(\mathbf{x})\}$ with appropriate coefficients β_j.

Equation 10 is not the only form of $\mathcal{C}_f$ that is valid. For example let us consider modelling some function that is periodic with known period λ_l in the l^{th} input direction. A covariance function that embodies such a periodicity is

$$\mathcal{C}_f(\mathbf{x}_n,\mathbf{x}_m;\Theta) = \theta_1 \exp\left[-\frac{1}{2}\sum_l \left(\frac{\sin\left(\frac{\pi}{\lambda_l}(x_m^{(l)} - x_n^{(l)})\right)}{r_l}\right)^2\right] \tag{12}$$

Functions described using this covariance function not only have strong correlations between points that are close together in input space but also have strong correlations between points that are separated from each other by a distance $\lambda_l \in \Theta$ in the l^{th} input direction.

Non-stationary covariance functions are a more complicated issue and will not be discussed in this paper.

2.3 Determining the Hyperparameters of a Gaussian process

We have a parameterized model for our data $\mathcal{D} = \{\mathbf{x}_n, t_n\}_{n=1}^N$. Now we need to find a consistent way in which to deal with the undetermined hyperparameters Θ. Ideally we would like to integrate over all the hyperparameters in order to make our predictions i.e. we would like to find

$$P(t_{N+1}|\mathbf{x}_{N+1},\mathcal{D},C(.)) = \int P(t_{N+1}|\mathbf{x}_{N+1},\mathcal{D},C(.),\Theta)P(\Theta|\mathcal{D},C(.))d\Theta \tag{13}$$

where $C(.)$ represents the form of the covariance function. For arbitrary $C(.)$ this is analytically intractable. There are two approaches we can take to evaluating equation 13. Either we can approximate the average prediction over all the possible values of the hyperparameters using the most probable values of hyperparameters (the Evidence framework (MacKay 1992a)) or we can perform the integration over Θ numerically using Monte Carlo methods (Williams and Rasmussen 1996; Neal 1993).

In this paper we shall only consider an Evidence framework approach. This approach uses an approximation to the integral in equation 13 based on the most probable set of hyperparameters Θ_{MP}.

$$P(t_{N+1}|\mathbf{x}_{N+1},\mathcal{D},C(.)) \simeq P(t_{N+1}|\mathbf{x}_{N+1},\mathcal{D},C(.),\Theta_{MP})P(\Theta_{MP}|\mathcal{D},C(.))\,\Delta\Theta \tag{14}$$

where $\Delta\Theta$ is the width of the distribution $P(\Theta|\mathcal{D},C(.))$ at Θ_{MP}. We already know that $P(t_{N+1}|\mathbf{x}_{N+1},$-$\mathcal{D},C(.),\Theta_{MP})$ is a Gaussian with mean and variance given by equations 7 and 8 when $\Theta = \Theta_{MP}$ so $P(t_{N+1}|\mathbf{x}_{N+1},\mathcal{D},C(.))$ is also Gaussian with the same mean and variance. The approximation is based on the assumption that the posterior distribution over Θ, $P(\Theta|t_{N+1},\mathbf{x}_{N+1},\mathcal{D},C(.))$, is sharply peaked around Θ_{MP} i.e. that $\Delta\Theta$ is small. This approximation is generally very good and the Evidence framework predictions are often identical to those found using the true predictive distribution (MacKay 1996).

In order to make use of the Evidence approximation we need to find the most probable hyperparameters. We can do this using a gradient based optimization routine[1] such as conjugate gradients, if we can find the derivatives of the posterior distribution of Θ. The posterior comprises three parts:

[1]A gradient based optimization routine is considered to be one that optimizes a function using only the derivatives of the function. No evaluation of the function itself need take place.

$$P(\Theta|\mathcal{D}, C(.)) = \frac{\overbrace{P(\mathbf{t}_N|\{\mathbf{x}_n\}, C(.), \Theta)}^{\text{likelihood}} \overbrace{P(\Theta)}^{\text{Prior}}}{\underbrace{P(\mathbf{t}_N|\{\mathbf{x}_n\}, C(.))}_{\text{evidence}}} \tag{15}$$

The evidence term is independent of Θ and will be ignored for the time being. The two remaining terms, the likelihood and the prior on Θ, will be considered in terms of the their logs. The log likelihood $\mathcal{L}$ for a Gaussian process is

$$\mathcal{L} = -\frac{1}{2}\log\det \mathbf{C}_N - \frac{1}{2}\mathbf{t}_N^T \mathbf{C}_N^{-1}\mathbf{t}_N - \frac{N}{2}\log 2\pi \tag{16}$$

and its derivative with respect to a hyperparameter θ is

$$\frac{\partial \mathcal{L}}{\partial \theta} = -\frac{1}{2}\text{trace}\left(\mathbf{C}_N^{-1}\frac{\partial \mathbf{C}_N}{\partial \theta}\right) + \frac{1}{2}\mathbf{t}_N^T \mathbf{C}_N^{-1}\frac{\partial \mathbf{C}_N}{\partial \theta}\mathbf{C}_N^{-1}\mathbf{t}_N \tag{17}$$

It is common practice to ignore the log prior term and perform a maximum likelihood optimization of the hyperparameters. However we shall place prior distributions on all our hyperparameters Θ to reflect any prior knowledge that we may have.

Assuming that finding the derivatives of the priors is straightforward, we can now search for Θ_{MP}. However there are two problems that we need to be aware of. Firstly the posterior distribution over Θ is often multi-modal. This can mean that the Θ_{MP} that is found by the optimization routine is dependent on the initial conditions. Suitable priors and a sensible parameterization of the covariance function often eliminate this problem. Secondly and perhaps most importantly the evaluation of the gradient of the log likelihood requires the evaluation of $\mathbf{C}_N^{-1}$. Any exact inversion method has an associated computational cost that is $\mathcal{O}(N^3)$ and so calculating gradients becomes time consuming for large training data sets.

2.4 Relationship to other Work

The study of Gaussian processes for regression is far from new. Within the geostatistics field, Matheron (1963) proposed a framework for regression using optimal linear estimators which he called 'kriging' after D.G. Krige, a South African mining engineer. This framework is identical to the Gaussian process approach to regression. 'Kriging' has been developed considerably in the last thirty years (see Cressie (1993) for an excellent review) including several Bayesian treatments (Omre 1987; Kitanidis 1986). However the geostatistics approach to the Gaussian process model has concentrated mainly on low-dimensional problems and has has largely ignored any probabilistic interpretation of the model and any interpretation of the individual parameters of the covariance function.

The Gaussian process framework encompasses a wide range of different regression models. O'Hagan (1978) introduced an approach which is essentially similar to Gaussian processes. Generalized radial basis functions (Poggio and Girosi 1989), ARMA models (Wahba 1990) and variable metric kernel methods (Lowe 1995) are all closely related to Gaussian processes.

The present interest in the area has been initiated by the work of Neal (1996) on priors for infinite networks. Neal showed that a neural network with one hidden layer converges to a Gaussian process as the number of hidden neurons tends to infinity if the network has a prior on input-to-hidden weights and hidden unit biases in which the weights for different hidden units are independent and identically distributed. The Bayesian interpretation of Gaussian processes was extended in Williams and Rasmussen (1996) and Neal (1997) and a thorough comparision of Gaussian processes with other methods such as neural networks and MARS was done by Rasmussen (1996).

Let us now consider, in slightly more detail, the relationship between Gaussian processes and radial basis function models. This will highlight the potential benifits of Gaussian processes as well as illustrating an important method for generating new covariance functions. We shall also compare Gaussian processes with neural networks and comment on the advantages of using the former.

2.4.1 Comparison with Radial Basis Function Model

Let us consider how Gaussian processes relate to basis function networks. As before, let us have some noisy data $\mathcal{D} = \{\mathbf{x}_n, t_n\}_{n=1}^{N}$. Let

$$t_n = y_n + \nu_n \tag{18}$$

where ν_n is random Gaussian noise and let us define a basis function model

$$y_n = \sum_{i=1}^{I} w_i \phi_i(\mathbf{x}_n) \tag{19}$$

where $\{\phi_i\}$ are a set of basis functions. We can write down a prior on the weights $\mathbf{w}$ and a prior on the noise

$$\begin{aligned} P(\mathbf{w}|\alpha) &= \frac{1}{Z_w} \exp\left(-\frac{\alpha}{2}\mathbf{w}^T\mathbf{w}\right) & (20) \\ P(\nu\,|\beta) &= \frac{1}{Z_\nu} \exp\left(-\frac{\beta}{2}|\nu|^2\right) & (21) \end{aligned}$$

where $\nu = (\nu_1, \cdots, \nu_N)$ and α and β are appropriate hyperparameters. We can show that the joint probability of the data is given by

$$P(\mathbf{t}|\{\mathbf{x}_n\}, \alpha, \beta) = \frac{1}{Z_t} \exp\left(-\frac{1}{2}\mathbf{t}_N^T \mathbf{G}^{-1} \mathbf{t}_{\mathrm{N}}\right) \tag{22}$$

where

$$\mathbf{G} = \alpha^{-1}\Phi\Phi^{\mathrm{T}} + \beta^{-1}\mathbf{I} \tag{23}$$

with $\Phi_{ij} = \phi_j(\mathbf{x}_i)$. Comparing equation 22 with equation 3 we can see that the basis function model is equivalent to a Gaussian process. Now consider using radial basis functions centred on the data points.

$$\phi_k(\mathbf{x}_i) = \exp\left(-\sum_{l=1}^{L} \frac{(x_i^{(l)} - x_k^{(l)})^2}{r_l^2}\right) \tag{24}$$

with set of length scales $\{r_l\}$. This gives us a covariance function which has the following data dependent part:

$$\mathcal{G}_f(\mathbf{x}_i, \mathbf{x}_j) = \sum_k \exp\left(-\sum_{l=1}^{L}\left\{\frac{(x_i^{(l)} - x_k^{(l)})^2}{r_l^2} + \frac{(x_j^{(l)} - x_k^{(l)})^2}{r_l^2}\right\}\right) \tag{25}$$

Note that this is not equivalent to to the expression in equation 10. However, consider the case where we have an infinite number of basis functions that are centred on an infinite number of distinct points. The summation in equation 25 becomes an integral and so the covariance function becomes

$$\mathcal{G}_f(\mathbf{x}_i, \mathbf{x}_j) = D \exp\left(-\frac{1}{2}\sum_{l=1}^{L} \frac{(x_i^{(l)} - x_j^{(l)})^2}{r_l^2}\right) \tag{26}$$

where D is a constant. This *is* equivalent to equation 10, showing that this form of the covariance function defines a regression model that is equivalent to a radial basis function model with an infinite number of basis functions. We do not however have to invert an infinite matrix in order to make predictions using the Gaussian process model. The rank of the matrix that we must invert scales linearly with the number of data points. Hence the Gaussian process offers us greater resolution and flexibility than a

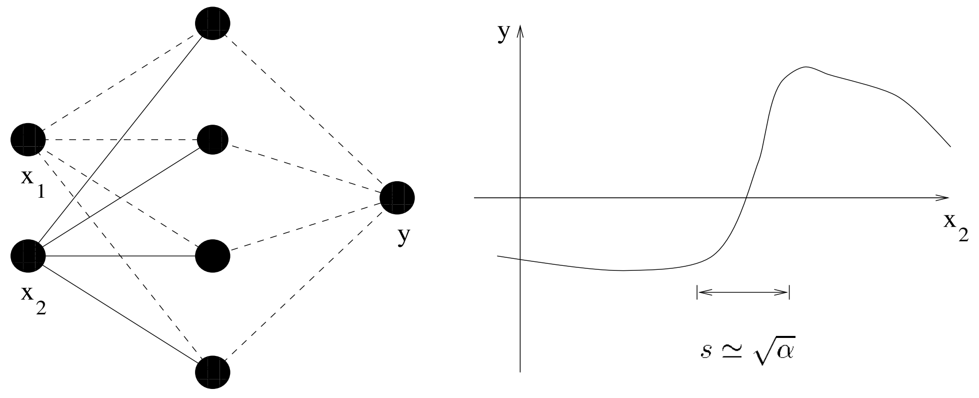


Figure 2: **Regularization and Length Scales :** Consider a fully connected MLP with one hidden layer as shown above left. Let the hidden and output layer neurons have sigmoid and linear activation functions respectively. We can define a prior distribution over the weights from the input x_2 to the hidden layer (shown in bold above) with a regularization constant α : $P(\mathbf{w}) \propto \exp\left(-\frac{\alpha}{2}\mathbf{w}^T\mathbf{w}\right)$. The hyperparameter α controls the smoothness of the output y with respect to the input x_2. A possible output of the MLP is shown above right. Due to the regularization, the output is unlikely to change significantly on a scale smaller than $s = \alpha^{1/2}$ when $\mathbf{w} = \mathbf{w}_{MP}$.

fixed basis function network without greater computational cost (assuming we do not have a very large amount of training data). The analogy to fixed basis function networks provides a good method to construct new forms of the covariance function which have certain desired properties, e.g. periodicity and non-stationarity.

2.4.2 Comparision to Neural Networks

As shown by Neal (1996), there is a strong relationship between Gaussian processes and neural networks. In fact the optimization of the hyperparameters of a Gaussian process is equivalent to the optimization of the hyperparameters of a neural network. There is a one to one correspondence between many of the neural network hyperparameters and the hyperparameters of the covariance function.

Consider an MLP with one hidden layer (see Figure 2) which has a regularization constant α on the weights between the input layer and the hidden layer. It can be shown that the output of the neural network $\mathbf{y}$ is unlikely to change significantly on a scale smaller than $s = \sqrt{\alpha}$. This distance s is analogous to the length scale r_l found in the covariance function in equation 10. Understanding how the regularization constants affect the form of the interpolant is difficult without relating them to length scales. Such a relationship is far from obvious for neural networks with many hidden layers. The length scales can be built into a Gaussian process covariance function explicitly leading to more readily understandable constraints on the form of the interpolant.

Determining the most probable values of the length scales is a straightforward process in the Gaussian process framework. This is because the weight vector has been integrated out using simple matrix operations. Optimizing hyperparameters for neural networks can be more complicated, requiring multiple optimizations of the weight vector. Rasmussen (1996) has done extensive comparisons of Gaussian processes and neural networks using the DELVE software environment. He reports better performance for Gaussian processes over MAP Bayesian neural networks for a wide range of problems.

3 Implementation of the Model

Having constructed a model, we now need to decide how best to implement it. There are two possible approaches we can take. We shall discuss each in turn and compare their strengths and weaknesses.

3.1 Direct Methods

The most obvious implementation of equations 7,8 and 17 is to evaluate the inverse of the covariance matrix exactly. This can be done using a variety of methods (LU decomposition, Gauss-Jordan Elimination etc.). Having obtained the explicit inverse, we then apply it directly to the appropriate vectors. Thus in order to calculate a single prediction $\hat{t}_{N+1}$, we must construct the vector $\mathbf{k}_{N+1}$, invert the matrix $\mathbf{C}_N$, calculate the vector $\mathbf{v} = \mathbf{C}_{\mathrm{N}}^{-1}\mathbf{t}_{\mathrm{N}}$ and then find the dot product $\hat{t}_{N+1} = (\mathbf{k}_{N+1}^T\mathbf{v})$. However if we wish to find the most probable value of the interpolant at a new point $\mathbf{x}_{N+2}$ given the data $\mathcal{D}$ (which does *not* include t_{N+1}), we need only construct the new vector $\mathbf{k}_{N+2}$ and find the dot product of this vector with $\mathbf{v}$ - only $\mathcal{O}(N)$ operations assuming perfect precision. This means that given the most probable hyperparameters and using the explicit representation of $\mathbf{C}_N^{-1}$ we can calculate the most probable value of the interpolant at M points for the cost of only *one* matrix inversion, *one* application of a matrix to a vector and M dot products. Each evaluation of the gradient of the log likelihood also requires the inversion of $\mathbf{C}_N$ as well as four matrix to vector applications and one dot product (note that the evaluation of trace $\left(\mathbf{C}_N^{-1}\frac{\partial \mathbf{C}_N}{\partial \theta}\right)$ does not require the explicit calculation of $\mathbf{C}_N^{-1}\frac{\partial \mathbf{C}_N}{\partial \theta}$ as we need only evaluate diagonal elements to find the trace).

The explicit method does have two principle disadvantages. Firstly, the inversion of $\mathbf{C}_N$ can be time consuming for large data sets. Secondly, the method is prone to numerical inaccuracies. The dot product $\mathbf{k}_{N+1}^T\mathbf{v}$ may turn out to be the sum of very large positive and negative numbers although its magnitude may be small. This can lead to large inaccuracies in the evaluation of the most probable value of the interpolant and its error bars when the model is implemented on a computer. The problem is caused by an ill-conditioned covariance matrix, generally corresponding to a model with a very small noise level. To avoid such numerical errors, we can use the LU decomposition of $\mathbf{C}_N$. Instead of calculating $\mathbf{v} = \mathbf{C}_{\mathrm{N}}^{-1}\mathbf{t}_{\mathrm{N}}$ and then finding the dot product $\mathbf{k}_{N+1}^T\mathbf{v}$, we find

$$\begin{aligned} \mathbf{d}_1 &= \mathbf{L}^{-1}\mathbf{t}_{\mathrm{N}} && (27) \\ \mathbf{d}_2 &= \left(\mathbf{U}^{-1}\right)^T \mathbf{k}_{N+1} && (28) \end{aligned}$$

and then calculate

$$\hat{t}_{N+1} = \mathbf{d}_1^{\mathrm{T}}\mathbf{d}_2 \qquad (29)$$

where the LU decomposition of $\mathbf{C}_N$ is defined as $\mathbf{C}_N = \mathbf{LU}$ where $\mathbf{L}$ and $\mathbf{U}$ are lower triangular and upper triangular matrices respectively. In tests using a covariance function of the form given in equation 9 with a very low noise level, errors that occured using the explicit method were removed using this approach. We must pay a price for this improved accuracy. Whenever we wish to calculate the value of the interpolant for a new $\mathbf{x}$, we must perform two applications of matrices to vectors in constrast to the simple dot product required for the explicit method. The LU Decomposition method still uses inversion techniques which scale $\mathcal{O}(N^3)$ and can be time consuming for large data sets.

3.2 Approximate Methods

We shall now discuss a method for the implementation of Gaussian processes based upon the ideas of Skilling (1993). The basic rationale behind these ideas is that we shall restrict ourselves in the computational cost of the matrix and vector operations that we can perform. We shall only allow the application of a matrix to a vector or the calculation of a dot product. No operations that scale greater than $\mathcal{O}(N^2)$ will be allowed. Thus multiplying a matrix by a matrix and exact matrix inversion are forbidden. Using the techniques described below, we can evaluate approximations to the inverse of a matrix applied to an arbitrary vector and the trace of the inverse of a matrix while not breaking these restrictions. Furthermore we can quantify the accuracy of our approximations also within the restrictions. This then provides us with the basic tools to implement the Gaussian process framework described in Section 2.1.

3.2.1 The Inverse

Say we have a symmetric matrix $\mathbf{C}_N$ and a vector $\mathbf{u}$ and that we wish to calculate $\mathbf{C}_N^{-1}\mathbf{u}$. Let us define the function

$$Q(\mathbf{y}) = \mathbf{y}^T\mathbf{u} - \frac{1}{2}\mathbf{y}^T\mathbf{C}_N\mathbf{y} \tag{30}$$

so that the matrix $\mathbf{C}_N$ is the effective Hessian matrix of Q. At the maximum of Q the gradient w.r.t $\mathbf{y}$ satisfies

$$\nabla Q_{max} = \mathbf{u} - \mathbf{C}_N\mathbf{y}_{max} = \mathbf{0} \tag{31}$$

and hence

$$\mathbf{y}_{max} = \mathbf{C}_N^{-1}\mathbf{u} \tag{32}$$

Hence finding the maximum of Q gives us the inverse of $\mathbf{C}_N$ applied to $\mathbf{u}$. Let us consider the following method for maximizing Q. Taking an initial vector $\mathbf{y}_1 = \mathbf{0}$, we define $\mathbf{g}_1 = \nabla Q(\mathbf{y}_1) = \mathbf{u}$ and $\mathbf{h}_1 = \mathbf{g}_1$. We then find the point $\mathbf{y}_2$ along the direction $\mathbf{h}_1$ which maximizes Q. It is straightforward to show that

$$\mathbf{y}_2 = \mathbf{y}_1 + \lambda_1\mathbf{h}_1 \tag{33}$$

where

$$\lambda_k = \frac{\mathbf{g}_k^T\mathbf{g}_k}{\mathbf{g}_k^T\mathbf{C}_N\mathbf{h}_k} \tag{34}$$

$\mathbf{y}_2$ can be thought of as a new estimate of $\mathbf{C}_N^{-1}\mathbf{u}$. We can write down $\mathbf{g}_2 = \nabla Q(\mathbf{y}_2)$ in terms of $\mathbf{g}_1$ using the recurrence

$$\mathbf{g}_{k+1} = \mathbf{g}_k - \lambda_k\mathbf{C}_N\mathbf{h}_k \tag{35}$$

We now define $\mathbf{h}_2$ using the recurrence

$$\mathbf{h}_{k+1} = \mathbf{g}_{k+1} + \gamma_k\mathbf{h}_k \tag{36}$$

where the scalar γ_k is given by

$$\gamma_k = \frac{\mathbf{g}_{k+1}\cdot\mathbf{g}_{k+1}}{\mathbf{g}_k\cdot\mathbf{g}_k} \tag{37}$$

This ensures that the new maximization direction $\mathbf{h}_2$ is conjugate to the old gradient $\mathbf{g}_1$. We can then calculate $\mathbf{y}_3$, a new estimate of $\mathbf{C}_N^{-1}\mathbf{u}$, using the relation

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \lambda_k\mathbf{h}_k. \tag{38}$$

Continuing this process for, say, K iterations, we calculate three sets of K vectors $\{\mathbf{g}_k\}, \{\mathbf{h}_k\}$ and $\{\mathbf{y}_k\}$ for $k = 1$ to K. Note that if $K = N$ then we are guaranteed to reach the maximum of Q where $\mathbf{y}_N$ is precisely equal to $\mathbf{C}^{-1}\mathbf{u}$ assuming perfect precision. Each iteration provides us with a new estimate, $\mathbf{y}_k$, of $\mathbf{C}_N^{-1}\mathbf{u}$ and a new lower bound, $Q(\mathbf{y}_k)$, on $Q(\mathbf{y}_{max})$:

$$0 = Q(\mathbf{y}_1) \leq Q(\mathbf{y}_2) \leq \cdots \leq Q(\mathbf{y}_K) \leq Q(\mathbf{y}_{max}) = \frac{1}{2}\mathbf{u}^T\mathbf{C}_N^{-1}\mathbf{u} \tag{39}$$

The approximation $\mathbf{y}_K \simeq \mathbf{C}_N^{-1}\mathbf{u}$ can, in fact, be good for K significantly less than N. Thus the approximate inversion process scales only as $\mathcal{O}(K \times N^2)$. This maximization procedure is analogous to the conjugate gradient algorithm. Because we are dealing with a quadratic form, the line minimizations are replaced by simple matrix manipulations. Conjugate gradient methods for inverting matrices are well established and further details can be found in Lanczos (1950), (G. H. Golub 1990) and (Brown *et al.* 1994).

It is useful to be able to gauge the accuracy of the approximation $\mathbf{y}_K$ to $\mathbf{C}_N^{-1}\mathbf{u}$. We can then stop the maximization procedure when we have achieved a specified accuracy and, hopefully, when K is significantly smaller than N. Using the Skilling method we already have a sequence of increasing lower bounds on $Q_{max} \equiv Q(\mathbf{y}_{max})$ (see equation 39). We can find a similar sequence of decreasing upper bounds for certain forms of $\mathbf{C}_N$. Let us assume $\mathbf{C}_N$ can be written

$$\mathbf{C}_N = \mathbf{A} + \theta_3\mathbf{I} \tag{40}$$

where $\mathbf{A}$ is symmetric and positive semi-definite. Let us also construct the function

$$Q^*(\mathbf{y}) = \mathbf{y}^T\mathbf{A}\mathbf{u} - \frac{1}{2}\mathbf{y}^T\mathbf{C}_N\mathbf{A}\mathbf{y}. \tag{41}$$

Consider the maximization of Q^* with respect to $\mathbf{A}^{\frac{1}{2}}\mathbf{y}$, i.e., the maximization of

$$Q^*(\mathbf{x}) = \mathbf{x}^T\mathbf{v} - \frac{1}{2}\mathbf{x}^T\mathbf{C}_N\mathbf{x} \tag{42}$$

with respect to $\mathbf{x} = \mathbf{A}^{\frac{1}{2}}\mathbf{y}$ where $\mathbf{v} = \mathbf{A}^{\frac{1}{2}}\mathbf{u}$. This mimics the maximization of Q except that $\mathbf{u}$ is replaced by $\mathbf{v}$. Consequently we can describe the maximization of Q^* with respect to $\mathbf{y}$ in terms of the conjugate gradient equations with modified λ_k and γ_k:

$$\begin{aligned}
\mathbf{h}_{k+1}^* &= \mathbf{g}_{k+1}^* + \gamma_k^*\mathbf{h}_k^* && (43)\\
\mathbf{g}_{k+1}^* &= \mathbf{g}_k^* - \lambda_k^*\mathbf{C}_N\mathbf{h}_k^* && (44)\\
\gamma_k^* &= \frac{(\mathbf{g}_{k+1}^*)^T\mathbf{A}\mathbf{g}_{k+1}^*}{(\mathbf{g}_k^*)^T\mathbf{A}\mathbf{g}_k^*} && (45)\\
\lambda_k^* &= \frac{(\mathbf{g}_k^*)^T\mathbf{A}\mathbf{g}_k^*}{(\mathbf{g}_k^*)^T\mathbf{A}\mathbf{C}_N\mathbf{h}_k^*} && (46)
\end{aligned}$$

and hence

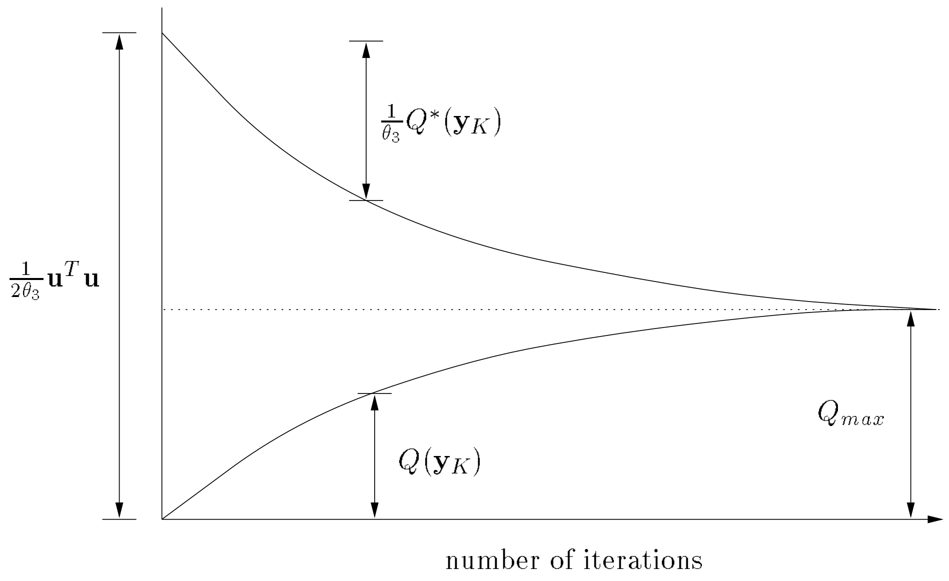
$$\mathbf{y}_{k+1}^* = \mathbf{y}_k^* + \lambda_k^* h_k^*. \tag{47}$$

The above equations are identical to those derived for Q except that λ_k^* and γ_k^* contain extra factors of $\mathbf{A}$. Starting with $\mathbf{y}_1^* = \mathbf{0}$ and $\mathbf{g}_1^* = \mathbf{h}_1^* = \mathbf{u}$, we can now derive a new sequence of lower bounds

$$0 = Q^*(\mathbf{y}_1^*) \leq Q^*(\mathbf{y}_2^*) \leq \cdots \leq Q^*(\mathbf{y}_K^*) \leq Q^*(\mathbf{y}_{max}^*) = \frac{1}{2}\mathbf{u}^T\mathbf{C}_N^{-1}\mathbf{A}\mathbf{u} \tag{48}$$

Hence, as

$$Q_{max}^* + \theta_3 Q_{max} = \frac{1}{2}\mathbf{u}^T\mathbf{C}_N^{-1}\left(\mathbf{A} + \theta_3\mathbf{I}\right)\mathbf{u} = \frac{1}{2}\mathbf{u}^T\mathbf{u} \tag{49}$$



$\frac{1}{\theta_3}Q^*(\mathbf{y}_K)$
$\frac{1}{2\theta_3}\mathbf{u}^T\mathbf{u}$
Q_{max}
$Q(\mathbf{y}_K)$
number of iterations

Figure 3: **Variation of lower and upper bounds on Q_{max} :** This figure shows a typical variation of the upper and lower bounds on Q_{max} with the number of iterations of the conjugate gradient inversion algorithm. Notice that the plot is not symmetrical about Q_{max}. However the relative convergence rates for the upper and lower bounds are the same because subspaces for the optimization of Q and Q^* are the same.

and $\mathbf{u}$ is known before we start the maximization, we can construct progressively decreasing upper bounds for Q_{max}.

$$Q(\mathbf{y}_K) \;\leq\; Q_{max} \;\leq\; \frac{1}{\theta_3}\left(\frac{1}{2}\mathbf{u}^T\mathbf{u} - Q^*(\mathbf{y}_K^*)\right) \tag{50}$$

Now that we have upper and lower bounds on Q_{max}, we can gauge the accuracy of our approximation. The ratio $\Delta Q_K/Q(\mathbf{y}_K)$ where ΔQ_K is the difference between the upper and lower bound at the K^{th} iteration proved a useful guide to the accuracy of the approximation in experiments performed by the authors. Simply fixing the number of conjugate gradient iterations K without checking to see if the approximation was accurate enough led to severe numerical instabilities. An example of the variation of upper and lower bounds can be seen in Figure 3.

3.2.2 The Trace

As well as calculating the inverse of a matrix applied to a vector, we need to calculate the trace of the inverse of a matrix under the same restrictions on computational cost. Let us construct

$$\tau = \mathbf{d}^T\mathbf{C}_N^{-1}\frac{\partial \mathbf{C}_N}{\partial \theta}\mathbf{d} \tag{51}$$

where $\mathbf{d}$ is a random vector with its elements d_n having a Gaussian distribution with zero mean and unit variance. Taking the expectation with respect to the distribution over $\mathbf{d}$,

$$E\left[\tau\right] = \text{trace}\left[\mathbf{C}_N^{-1}\frac{\partial \mathbf{C}_N}{\partial \theta}\right] \tag{52}$$

and

	Explicit	LU decomp.	Approximate	
			operations	$\mathbf{g}_i$ calc.
Initialization	$2N^3$	$2N^3/3$	$3KN^2 + (4+L)N^2$	K
Prediction :				
mean	N	N^2	N	0
error bars	N^2	$2N^2$	$3KN^2$	K
Gradient	$5N^2$	$5N^2$	$3\rho K'N^2$	$\rho K'$

Table 1: **Computational Cost of Methods.** This tables gives, to leading order, the number of floating point operations required to perform four tasks for three different methods: the initialization cost and the extra cost to make a prediction, calculate error bars and evaluate the gradient of the log likelihood. The table also gives the number of $\mathbf{g}_i$ vectors needed for each task using the approximate inversion method. N is the dimension of the matrix $\mathbf{C}_N$, K is the number of $\mathbf{g}_i$ vectors generated in order to find $\mathbf{C}_N^{-1}\mathbf{t}$ using the approximate approach. ρ is the number of random samples τ and K' is the number of $\mathbf{g}_i$ vectors used in trace estimation for the approximate approach.

$$\sigma_\tau^2 = 2\,\mathrm{trace}\left[\left(\mathbf{C}_N^{-1}\frac{\partial \mathbf{C}_N}{\partial \theta}\right)^2\right] \tag{53}$$

Hence we may approximate the trace by averaging several different τ's (i.e over several $\mathbf{d}$'s). In fact, for large matrices, the number of τ's needed to obtain a good estimate of the trace is surprisingly small. Also, as we increase the size of the matrix, the number of random samples we require to achieve a given level of accuracy on our estimation of the trace decreases which makes the method extremely efficient for large matrices.

Calculating the individual τ's used to find the trace requires us to evaluate $\mathbf{d}^T\mathbf{C}_N^{-1}\frac{\partial \mathbf{C}_N}{\partial \theta}\mathbf{d}$. This can be done using the methods described in Section 3.2.1 to calculate $\mathbf{y}_K$ and $\mathbf{y}_K^*$ (which are both approximations to $\mathbf{C}_N^{-1}\mathbf{d}$)

3.3 Counting the Cost

Having now described different methods for implementing a Gaussian process we shall look more closely at their computational cost. Table 1 gives the cost associated with the implementation of a Gaussian process using three different methods - the explicit inversion method, the LU decomposition method and the approximate inversion method. Only leading order terms are included. The number of gradient vectors $\mathbf{g}_i$ that need to be evaluated by the approximate inversion method is also given for each task. Table 1 first gives the initialization cost which is primarily the cost of inverting $\mathbf{C}_N$ (whether explicitly or otherwise). This initialization is necessary before we are able to make any predictions or to evaluate any gradients. The extra cost of making a prediction at a new point $\mathbf{x}_{N+1}$ and calculating the error bars is then given. The table also gives the extra cost of calculating the gradient of the log likelihood $\mathcal{L}$ given that we have already paid the initialization cost.

In order to produce meaningful statistics for these tasks, we have made several assumptions. Firstly the computational cost of evaluating $\mathbf{C}_N$ has been assumed to be $\mathcal{O}(LN^2)$ where L is the dimensionality of the input space. This may not be the case for very complicated noise models. Secondly we have assumed for the approximate inversion method that any set of bounds is only calculated once. This is not unreasonable providing K and K' are sufficiently large (for definitions of K and K' see Table 1).

There are two important points to note from Table 1. Firstly, the scaling of the initialization cost with the number of data points N for the direct methods is $\mathcal{O}(N^3)$ but for the approximate methods it is $\mathcal{O}(KN^2)$. Thus as the amount of data increases, the approximate methods are significantly faster than the direct ones as, for moderately well conditioned matrices, K can be significantly smaller than N and ρ can be as low as 2. Secondly, we should note that, for both the explicit and the approximate inversion methods, once we have paid the initial cost to invert $\mathbf{C}_N$ we can make predictions at new points very efficiently. The extra accuracy of the LU decomposition requires slightly more computation for this task.

4 Further Methods

4.1 On-line Data addition

Consider the case in which we have a data set $\mathcal{D}$ and have found the most probable hyperparameters of the Gaussian process Θ_{MP} given the data. Then we obtain a new piece of data and wish to incorporate this into the model. Let us assume that the new piece of data does not imply a change in the hyperparameters i.e. that the form of the function described by the new larger data set is approximately the same as that described by the old.

Consider a data set $\mathcal{D} = \{\mathbf{x}_n, t_n\}$ $(n = 1 \cdots N)$ with a covariance matrix $\mathbf{C}_N$. We obtain a new data point $\{\mathbf{x}_{N+1}, t_{N+1}\}$ and wish to calculate $\mathbf{C}_{N+1}^{-1}$. $\mathbf{C}_{N+1}$ can be defined in a partitioned form (see Figure 1). Using the partitioned inverse equations (Barnett 1979)

$$\mathbf{C}_{N+1}^{-1} = \begin{bmatrix} \mathbf{M} & \mathbf{m} \\ \mathbf{m}^{\mathrm{T}} & \mu \end{bmatrix}$$

where

$$\begin{aligned} \mu &= \left(\kappa - \mathbf{k}_{N+1}^T \mathbf{C}_N^{-1} \mathbf{k}_{N+1}\right)^{-1} \\ \mathbf{m} &= -\mu\, \mathbf{C}_N^{-1} \mathbf{k}_{N+1} \\ \mathbf{M} &= \mathbf{C}_N^{-1} - \mu\, \mathbf{k}_{N+1} \mathbf{k}_{N+1}^T \end{aligned}$$

For the direct implementation, where we already have an explicit form for $\mathbf{C}_N^{-1}$, the most costly part of calculating $\mathbf{C}_{N+1}^{-1}$ is the application of a matrix to a vector requiring $\mathcal{O}(N^2)$ operations in comparison to the $\mathcal{O}(N^3)$ operations required for the inversion of $\mathbf{C}_{N+1}$ from scratch.

Now consider the situation for the approximate inversion approach. We wish to calculate $\mathbf{C}_{N+1}^{-1}\mathbf{t}_{N+1}$ using only $\mathbf{C}_N^{-1}\mathbf{t}_N$ and the data. This is not possible as we need to evaluate $\mathbf{k}_{N+1}^T \mathbf{C}_N^{-1} \mathbf{k}_{N+1}$ in order to calculate the new inverse. However $\mathbf{k}_{N+1}^T \mathbf{C}_N^{-1} \mathbf{k}_{N+1}$ can be calculated in a similar way to $\mathbf{C}_N^{-1}\mathbf{t}_N$ and hence we can find $\mathbf{C}_{N+1}^{-1}\mathbf{t}_{N+1}$ using the form of $\mathbf{C}_{N+1}^{-1}$ given above.

4.2 Multiple Outputs

The subject of multiple outputs (or co-kriging (Cressie 1993)) is problematic. It is possible to re-formulate Gaussian processes such that they can deal with multiple outputs. This requires us to consider the definition of the covariance function $C_{nm}^{ab} = C(\mathbf{x}_n, \mathbf{x}_m, n, m, a, b; \Theta)$ where a and b are dimensions in the output space. It is not clear how such a covariance function should be defined. However for many problems symmetry suggests that the covariance function will be of the form $C_{nm}^{ab} = \delta_{ab} C_{nm}$. Thus the method of Multi-kriging suggested by Williams and Rasmussen (1996) in which each output is modelled independently seems the most sensible approach.

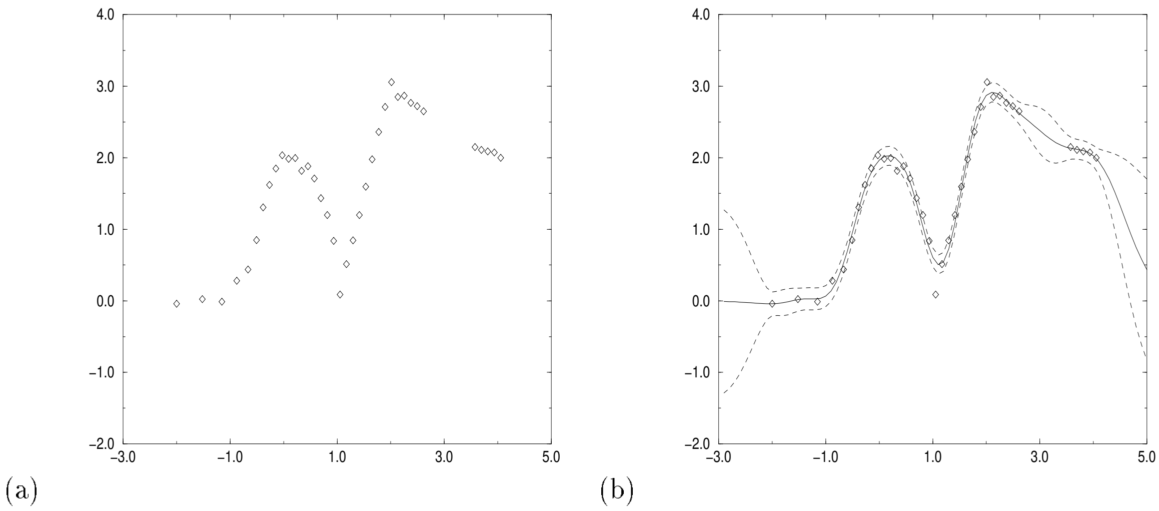


Figure 4: **1d Example :** (a) shows the 37 noisy data points used as the training data. Note the lack of data in the region $2.5 \le x \le 3.5$. In (b) we see the interpolant and its 1σ error bars. The error bars represent how uncertain we are about the interpolant at each point assuming that the model is correct. Note how the error bars increase as the data point density decreases. Note that the point near $(1, 0)$ is outside the error bars because the prior on the length scales specifies that the interpolant is moderately smooth. Hence this point is treated as an improbable outlier.

5 Examples

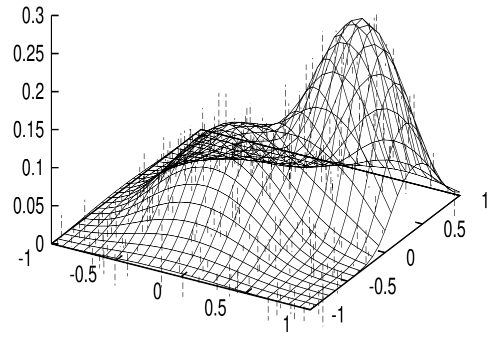
We shall consider two simple examples. The first example is a simple 1 dimensional regression problem. This consists of a set of 37 noisy training data points which are shown in Figure 4(a). We used the basic covariance function given in equation 10 and an input independent noise model to provide a regression model.

10 runs were performed with different sets of initial conditions to guard against the possibility of multiple maxima in $P(\Theta|\mathcal{D})$. Broad priors were placed on all the hyperparameters (Gamma priors on the length scales; Inverse Gamma prior on θ_1, θ_2 and θ_3) as it was assumed that there was little prior knowledge other than a belief that the interpolant should be relatively smooth. For each run the initial values of the hyperparameters were sampled from their priors and then a conjugate gradient optimization routine was used to find Θ_{MP}. The matrix inversions involved in the optimization were performed using exact LU-decomposition methods. Each run took approximately 10 seconds on a Sun SPARC classic.

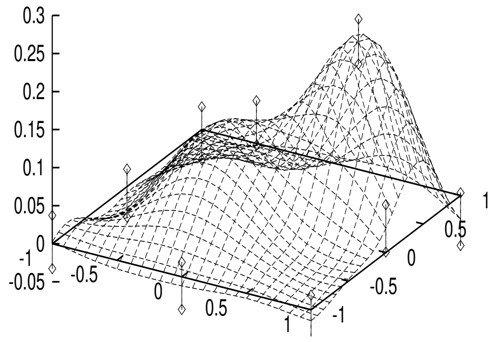
The results can be seen in Figure 4(b). One can compare these results with those of MacKay (1992a) using a radial basis function model. It should be noted that a radial basis function model is not equivalent to a Gaussian process defined by a covariance function of the form of equation 10 (see Section 2.4.1). This explains the slight discrepancies between the results in Figure 4(b) and those in MacKay (1992a).

For the second example 400 noisy data points were generated from a 2D function (see Figure 5(a)). Again broad priors were placed on all the hyperparameters and ten runs were performed, each with initial hyperparameters sampled from their priors. However in this case the approximate techniques of Section 3.2 were used to evaluate inverses and traces. Ten iterations of the inversion process were used ($K = K' = 10$) to approximate the inverse applied to a vector and two random vectors ($\rho = 2$) were used to evaluate the trace. An accuracy of $\Delta Q_K/Q_K = 0.01$ was required for the approximate inversions. The results from each run were very similar and the interpolant obtained from the first run is shown in Figure 5(b).

Let us consider the time required to perform the optimization in the second example. Each training run



(a)



(b)

Figure 5: **2d Example :** (a) shows the function from which 400 noisy data points were generated and the noisy data points in relation to the function. The offset of each datum due to noise is shown as a dashed line. In (b) we see the most probable interpolant generated using an approximate implementation of a Gaussian process. 1σ error bars are only given at selected points in order to preserve clarity.

took about 16 minutes using the approximate inversion techniques. Training runs performed using direct methods which achieved the same test error took approximately 39 minutes. This shows the principal advantage of the approximate methods when dealing with large amounts of training data. Figure 6 shows the training times for data sets of varying size generated using the same function.

6 Discussion

We have presented the method of interpolation using Gaussian processes and detailed how this can be efficiently implemented using both exact and approximate techniques.

Areas for further investigation include (i) analysis of the eigenvalue structure of $\mathbf{C}_N$ and its relationship to the covariance function to determine the conditions under which numerical problems occur; (ii) investigation of alternative parameterization of the covariance function to avoid numerical problems; (iii) application of Gaussian processes to classification.

7 Acknowledgements

The authors would like to thank Steve Gull, Simon Wilson and Anthony Challinor for helpful discussions and John Skilling for inspiration.

References

Barnett, S. (1979) *Matrix Methods for Engineers and Scientists*. McGraw-Hill.

Brown, J. D., Chu, M. T., Ellison, D. C., and Plemmons, R. J. eds. (1994) *Proceedings of the Cornelius Lanczos International Centenary Conference*. Philadelphia,PA: SIAM Publications.

Cressie, N. (1993) *Statistics for Spatial Data*. Wiley.

G. H. Golub, C. V. L. (1990) *Matrix Computation*. Baltimore: John Hopkins University Press.

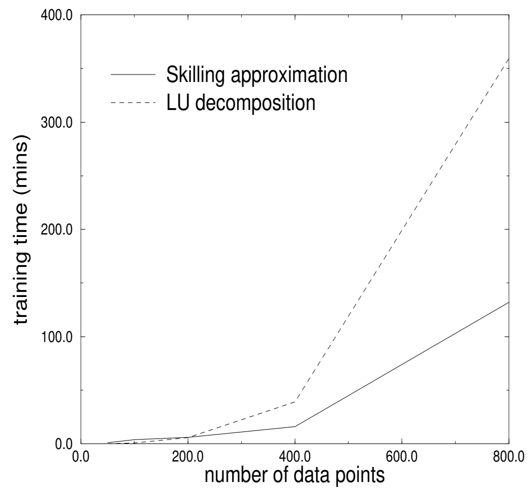


Figure 6: **Scaling of training time with amount of data :** This figure shows the training times for noisy data sets of different sizes generated from the function shown in Figure 5. Times are shown for both the LU decomposition method and the approximate method. For small data sets the LU method is significantly quicker than the approximate approach. At around 200 data points we can see that the $\mathcal{O}(N^2)$ scaling of the approximate method begins to yield benifits. For 400 and 800 data points the approximate method is significantly faster than the LU decomposition method.

Kitanidis, P. K. (1986) Parameter uncertainty in estimation of spatial functions: Bayesian analysis. *Water Resources Research* **22**: 499–507.

Lanczos, C. (1950) An iteration method for the solution of the eigenvalue problem for linear differential and integral operators. *Journal of Research (National Bureau of Standards)* **45**: 255–282.

Lowe, D. G. (1995) Similarity metric learning for a variable kernel classifier. *Neural Computation* **7**: 72–85.

MacKay, D. J. C. (1992a) Bayesian interpolation. *Neural Computation* **4** (3): 415–447.

MacKay, D. J. C. (1992b) A practical Bayesian framework for backpropagation networks. *Neural Computation* **4** (3): 448–472.

MacKay, D. J. C. (1994) Bayesian methods for backpropagation networks. In *Models of Neural Networks III*, ed. by E. Domany, J. L. van Hemmen, and K. Schulten, chapter 6. New York: Springer-Verlag.

MacKay, D. J. C. (1996) Hyperparameters: Optimize, or integrate out? In *Maximum Entropy and Bayesian Methods, Santa Barbara 1993*, ed. by G. Heidbreder, pp. 43–60, Dordrecht. Kluwer.

Matheron, G. (1963) Principles of geostatistics. *Economic Geology* **58**: 1246–1266.

Neal, R. M. (1993) Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG–TR–93–1, Dept. of Computer Science, University of Toronto.

Neal, R. M. (1996) *Bayesian Learning for Neural Networks*. Number 118 in Lecture Notes in Statistics. New York: Springer.

Neal, R. M. (1997) Monte Carlo Implementation of Gaussian process Models for Bayesian Regression and Classification. Technical Report CRG–TR–97–2, Dept. of Computer Science, University of Toronto.

O'Hagan, A. (1978) On curve fitting and optimal design for regression. *Journal of the Royal Statistical Society, B* **40**: 1–42.

Omre, H. (1987) Bayesian kriging - merging observations and qualified guesses in kriging. *Mathematical Geology* **19**: 25–39.

Poggio, T., and Girosi, F. (1989) A theory of networks for approximation and learning. Technical Report A.I. 1140, M.I.T.

Rasmussen, C. E., (1996) *Evaluation of Gaussian Processes and Other Methods for Non-Linear Regression*. University of Toronto dissertation.

Skilling, J. (1993) Bayesian numerical analysis. In *Physics and Probability*, ed. by W. T. G. Jr. and P. Milonni. C.U.P.

Wahba, G. (1990) *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics. CBMS-NSF Regional Conference series in applied mathematics.

Williams, C. K. I., (1995) Regression with Gaussian processes. To appear in Annals of Mathematics and Artificial Intelligence: Models, Algorithms and Applications.

Williams, C. K. I., and Rasmussen, C. E. (1996) Gaussian processes for regression. In *Advances in Neural Information Processing Systems 8*, ed. by D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo. MIT Press.