BAYESIAN STATISTICS 6, pp. 000--000
J. M. Bernardo, J. O. Berger, A. P. Dawid and A. F. M. Smith (Eds.)

Regression and Classification Using Gaussian Process Priors

RADFORD M. NEAL
University of Toronto, CANADA

SUMMARY

Gaussian processes are a natural way of specifying prior distributions over functions of one or more input variables. When such a function defines the mean response in a regression model with Gaussian errors, inference can be done using matrix computations, which are feasible for datasets of up to about a thousand cases. The covariance function of the Gaussian process can be given a hierarchical prior, which allows the model to discover high-level properties of the data, such as which inputs are relevant to predicting the response. Inference for these covariance hyperparameters can be done using Markov chain sampling. Classification models can be defined using Gaussian processes for underlying latent values, which can also be sampled within the Markov chain. Gaussian processes are in my view the simplest and most obvious way of defining flexible Bayesian regression and classification models, but despite some past usage, they appear to have been rather neglected as a general-purpose technique. This may be partly due to a confusion between the properties of the function being modeled and the properties of the best predictor for this unknown function.

Keywords: GAUSSIAN PROCESSES; NONPARAMETRIC MODELS; MARKOV CHAIN MONTE CARLO.

In this paper, I hope to persuade you that Gaussian processes are a fruitful way of defining prior distributions for flexible regression and classification models in which the regression or class probability functions are not limited to simple parametric forms. The basic idea goes back many years in a regression context, but is nevertheless not widely appreciated. The use of general Gaussian process models for classification is more recent, and to my knowledge the work presented here is the first that implements an exact Bayesian approach.

One attraction of Gaussian processes is the variety of covariance functions one can choose from, which lead to functions with different degrees of smoothness, or different sorts of additive structure. I will describe some of these possibilities, while also noting the limitations of Gaussian processes.

I then discuss computations for Gaussian process models, starting with the basic matrix operations involved. For classification models, one must integrate over the ''latent values'' underlying each case. I show how this can be done using Markov chain Monte Carlo methods. In a full-fledged Bayesian treatment, one must also integrate over the posterior distribution for the hyperparameters of the covariance function, which can also be done using Markov chain sampling. I show how this all works for a synthetic three-way classification problem. The methods I describe are implemented in software that is available from my web page, at `http://www.cs.utoronto.ca/~radford/`.

Gaussian processes and related methods have been used in various contexts for many years. Despite this past usage, and despite the fundamental simplicity of the idea, Gaussian process models appear to have been little appreciated by most Bayesians. I speculate that this could be partly due to a confusion between the properties one expects of the true function being modeled and those of the best predictor for this unknown function. Clarity in this respect is particularly important when thinking of flexible models such as those based on Gaussian processes.

1. MODELS BASED ON GAUSSIAN PROCESS PRIORS

Assume we have observed data for n cases, $(x^{(1)}, t^{(1)}), (x^{(2)}, t^{(2)}), \ldots, (x^{(n)}, t^{(n)})$, in which $x^{(i)} = x_1^{(i)}, \ldots, x_p^{(i)}$ is the vector of p ''inputs'' (predictors) for case i and $t^{(i)}$ is the associated ''target'' (response). Our primary purpose is to predict the target, $t^{(n+1)}$, for a new case where we have observed only the inputs, $x^{(n+1)}$. For a regression problem, the targets will be real-valued; for a classification problem, the targets will be from the set $\{0, \ldots, K-1\}$. It will sometimes be convenient to represent the distributions of the targets, $t^{(i)}$, in terms of unobserved ''latent values'', $y^{(i)}$, associated with each case.

Bayesian regression and classification models are usually formulated in terms of a prior distribution for a set of unknown model parameters, from which a posterior distribution for the parameters is derived. If our focus is on prediction for a future case, however, the final result is a predictive distribution for a new target value, $t^{(n+1)}$, that is obtained by integrating over the unknown parameters. This predictive distribution could be expressed directly in terms of the inputs for the new case, $x^{(n+1)}$, and the inputs and targets for the n observed cases, without any mention of the model parameters. Furthermore, rather than expressing our prior knowledge in terms of a prior for the parameters, we could instead simply specify a prior distribution for the targets in any set of cases. A predictive distribution for an unknown target can then be obtained by conditioning on the known targets.

These operations are most easily carried out if all the distributions are Gaussian. Fortunately, Gaussian processes are flexible enough to represent a wide variety of interesting models, many of which would have an infinite number of parameters if formulated in more conventional fashion.

1.1 *A Gaussian process for linear regression*

Before discussing such flexible models, however, it may help to see how the scheme works for a simple linear regression model, which can be written as

$$t^{(i)} \;=\; \alpha \;+\; \sum_{u=1}^{p} x_u^{(i)} \beta_u \;+\; \epsilon^{(i)} \tag{1}$$

where $\epsilon^{(i)}$ is the Gaussian ''noise'' for case i, assumed to be independent from case to case, and to have mean zero and variance σ_ϵ^2. For the moment, we will assume that σ_ϵ^2 is known, but that α and the β_u are unknown.

Let us give α and the β_u independent Gaussian priors with means of zero and variances σ_α^2 and σ_u^2. For any set of cases with fixed inputs, $x^{(1)}, x^{(2)}, \ldots$, this prior distribution for parameters implies a prior distribution for the associated target values, $t^{(1)}, t^{(2)}, \ldots$, which will be multivariate Gaussian, with mean zero, and with covariances given by

$$\begin{aligned} \mathrm{Cov}[t^{(i)}, t^{(j)}] \;&=\; E\Big[\Big(\alpha \;+\; \sum_{u=1}^{p} x_u^{(i)}\beta_u \;+\; \epsilon^{(i)}\Big)\Big(\alpha \;+\; \sum_{u=1}^{p} x_u^{(j)}\beta_u \;+\; \epsilon^{(j)}\Big)\Big] \\ &=\; \sigma_\alpha^2 \;+\; \sum_{u=1}^{p} x_u^{(i)} x_u^{(j)} \sigma_u^2 \;+\; \delta_{ij}\sigma_\epsilon^2 \end{aligned} \tag{2}$$

where δ_{ij} is one if $i = j$ and zero otherwise. This mean and covariance function define a ''Gaussian process'' giving a distribution over possible relationships between the inputs and the target. (One might wish to confine the term ''Gaussian process'' to distributions over functions from the inputs to the target. The relationship above is not functional, since (due to noise) $t^{(i)}$ may differ from $t^{(j)}$ even if $x^{(i)}$ is identical to $x^{(j)}$, but the looser usage is convenient.)

Suppose now that we know the inputs, $x^{(1)}, \ldots, x^{(n)}$, for n observed cases, as well as $x^{(n+1)}$, the inputs in a case for which we wish to predict the target. We can use the covariance function above to compute the $n+1$ by $n+1$ covariance matrix of the associated targets, $t^{(1)}, \ldots, t^{(n)}, t^{(n+1)}$. Together with the assumption that the means are zero, these covariances define a Gaussian joint distribution for the targets in these cases. We can condition on the known targets to obtain the predictive distribution for $t^{(n+1)}$ given $t^{(1)}, \ldots, t^{(n)}$. Well-known results show that this predictive distribution is Gaussian, with mean and variance given by

$$E\Big[t^{(n+1)} \,|\, t^{(1)}, \ldots, t^{(n)}\Big] \;=\; k^T C^{-1} t \tag{3}$$

$$\mathrm{Var}\Big[t^{(n+1)} \,|\, t^{(1)}, \ldots, t^{(n)}\Big] \;=\; V \;-\; k^T C^{-1} k \tag{4}$$

where C is the n by n covariance matrix of the observed targets, $t = [t^{(1)} \cdots t^{(n)}]^T$ is the vector of known values for these targets, k is the vector of covariances between $t^{(n+1)}$ and the n known targets, and V is the prior variance of $t^{(n+1)}$ (ie, $\mathrm{Cov}[t^{(n+1)}, t^{(n+1)}]$).

1.2 *More general Gaussian processes*

The procedure just described is unnecessarily expensive for a typical linear regression model with $p \ll n$. However, the Gaussian process procedure can handle more interesting models by simply using a different covariance function. For example, a regression model based on a class of smooth functions can be obtained using a covariance function of the form

$$\mathrm{Cov}[t^{(i)}, t^{(j)}] \;=\; \eta^2 \exp\Big(-\sum_{u=1}^{p} \rho_u^2 (x_u^{(i)} - x_u^{(j)})^2\Big) \;+\; \delta_{ij}\sigma_\epsilon^2 \tag{5}$$

There are many other possibilities for the covariance function, some of which are discussed in Section 2.

Our prior knowledge will usually not be sufficient to fix appropriate values for the hyperparameters in the covariance function (σ_ϵ, η, and the ρ_u for the model above). We will therefore give prior distributions to the hyperparameters, and base predictions on a sample of values from their posterior distribution. Sampling from the posterior distribution requires computation of the log likelihood based on the n observed cases, which is

$$L \;=\; -\,\frac{n}{2}\log(2\pi) \;-\; \frac{1}{2}\log\det C \;-\; \frac{1}{2} t^T C^{-1} t \tag{6}$$

The derivatives of L can also be computed, if they are needed by the sampling method.

1.3 *Gaussian process models for classification*

Models for classification problems, where the targets are from the set $\{0, \ldots, K-1\}$, can be defined in terms of a Gaussian process model for ''latent values'' associated with each case. These latent values are used to define a distribution for the target in a case.

For example, a logistic model for binary targets can be defined in terms of latent values $y^{(i)}$ by letting the distribution for the target in case i be given by

$$P(t^{(i)} = 1) \;=\; \Big[1 + \exp(-\,y^{(i)})\Big]^{-1} \tag{7}$$

The latent values are given some Gaussian process prior, generally with zero mean, and with some appropriate covariance function.

When there are three or more classes, an analogous model can be defined using K latent values for each case, $y_0^{(i)}, \ldots, y_{K-1}^{(i)}$, which define class probabilities as follows:

$$P(t^{(i)} = k) \;=\; \exp\,(y_k^{(i)}) \Big/ \sum_{k'=0}^{K-1} \exp\,(y_{k'}^{(i)}) \tag{8}$$

The K latent values can be given independent and identical Gaussian process priors. This representation is redundant, but removing the redundancy by forcing the latent values for one of the classes to be zero would introduce an arbitrary asymmetry into the prior.

The covariance function for the latent values in a classification model must usually include at least a small amount of ''jitter'', which is similar to noise in a regression model. This will certainly be necessary (at least if computations are done as described in this paper) whenever two cases were observed in which the input values were identical, since without jitter the covariance matrix of the latent values will be singular. One possibility is a covariance function analogous to that of equation (5):

$$\mathrm{Cov}[y^{(i)}, y^{(j)}] \;=\; \eta^2 \exp\Big(-\sum_{u=1}^{p} \rho_u^2 (x_u^{(i)} - x_u^{(j)})^2\Big) \;+\; \delta_{ij} J^2 \tag{9}$$

Here, J is the amount of jitter. A small amount (eg, $J = 0.1$) is sufficient to make the matrix computations better conditioned, and to improve the efficiency of some Markov chain methods used to sample for latent values, while altering the logistic model only slightly.

A larger amount of jitter can be used to produce the effect of a probit model, since when viewed on a sufficiently large scale, the logistic function looks like a threshold function; see (Neal 1997) for more details. A similar scheme is used by Wood and Kohn (1998) for a model based on splines. The difference between a logit and a probit model will be minor when the prior produces functions of general form (eg, when using the covariance function of equation (9)), but could be significant if a prior expressing a preference for an additive or other restricted model were used. The amount of jitter can be made a hyperparameter, allowing the data to choose from a continuum of models from logit to probit.

Latent values have also been used to extend Gaussian process models to regression problems with non-Gaussian noise (Neal 1997) or with input-dependent noise (Goldberg, Williams, and Bishop 1998). Many other models (eg, for Poisson regression) can be defined using latent values in a similar way.

2. COVARIANCE FUNCTIONS AND THEIR HYPERPARAMETERS

A wide variety of covariance functions can be used in the Gaussian process framework, subject to the requirement that a valid covariance function must always result in a positive definite covariance matrix for the targets. In a Bayesian model, the covariance function will usually depend on various hyperparameters, which are themselves given prior distributions. These hyperparameters can control the amount of noise in a regression model, the degree to which various input variables are relevant, and the magnitudes of different additive components of a model. The posterior distribution of these hyperparameters will be concentrated on values that are appropriate for the data that was actually observed.

In contrast to the elaborate forms for the covariance function described here, the mean function for the Gaussian process will usually be set to zero. This does not mean that the actual function is expected to be centred around zero, but merely that we lack prior knowledge of the function's overall level.

2.1 *Constructing covariance functions*

A variety of covariance functions can be constructed by adding and multiplying other covariance functions, since the element-by-element sum or product of any two symmetric, positive semidefinite matrices is also symmetric and positive semidefinite. Sums of covariance functions are useful in defining models with an additive structure, since the covariance function for a sum of independent Gaussian processes is simply the sum of their separate covariance functions. Products of covariance functions are useful in defining a covariance function for models with many inputs in terms of covariance functions for single inputs.

In particular, many useful covariance functions can be constructed by adding together one or more of the following: (1) A constant part, which is the same for any pair of cases, regardless of the inputs in those cases. This introduces a constant component to the regression function (or to the latent values for a classification model). The constant part of the covariance function is the prior variance for the value of this constant component in the function being modeled. (2) A linear part, which for the covariance between cases i and j has the form

$$\sum_{u=1}^{p} x_u^{(i)} x_u^{(j)} \sigma_u^2 \tag{10}$$

This can be used to produce a linear function of the inputs, as seen in Section 1.1, or more generally, to add a linear component to the overall function. (3) Any number of exponential parts, each of which has the form

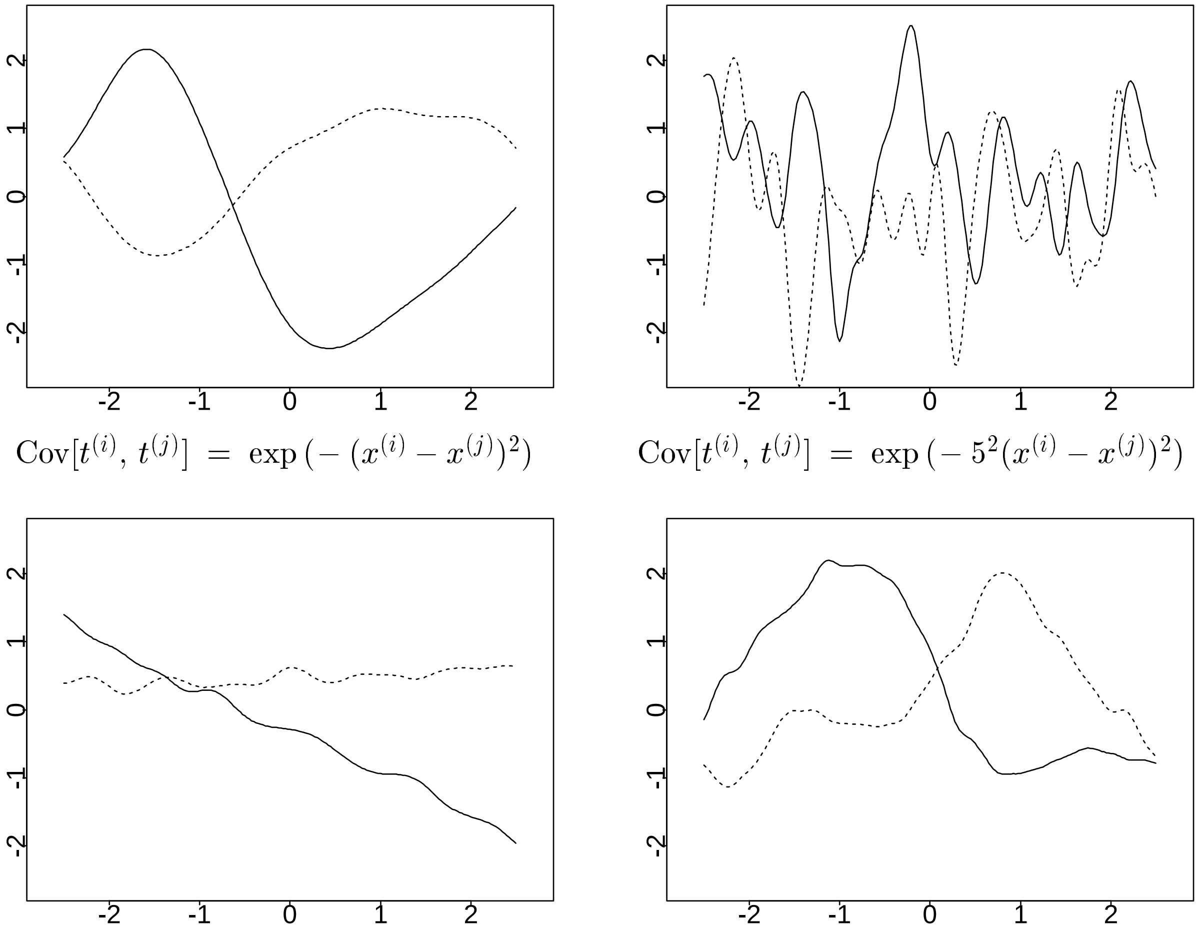
$$\eta^2 \prod_{u=1}^{p} \exp\Big(-\ \rho_u^R \mid x_u^{(i)} - x_u^{(j)} \mid^R\Big) \tag{11}$$

For the covariance function to be positive definite, R must be in the range 0 to 2. An exponential part with $R = 2$ will result in a component of the function that is infinitely differentiable; when $R < 2$, the component is not differentiable.

For a regression model, one would usually also include a noise term, which is zero for different cases and some constant for the covariance of the value in a case with itself. An analogous jitter term is included when defining a Gaussian process for latent values in a classification model.

2.2 *Effects of various covariance functions*

Some of the possible distributions over functions that can be obtained using covariance functions of the above form are illustrated in Figure 1. (These are functions of a single input, so the index u is dropped). The top left and top right each show two functions drawn randomly from a Gaussian process with a covariance function consisting of a single exponential part. The distance over which the function varies by an amount comparable to its full range, given by $1/\rho$, is smaller for the top-right than the top-left. The bottom left shows two functions generated using a covariance function that is the sum of constant, linear, and exponential parts. The magnitude of the exponential part, given by η, is rather small, so the functions depart only slightly from straight lines. The bottom right shows two functions drawn from a prior whose covariance function is the sum of two exponential parts, that produce variation at different scales, and with different magnitudes.



$\mathrm{Cov}[t^{(i)}, t^{(j)}] = \exp\left(-\,(x^{(i)} - x^{(j)})^2\right)$

$\mathrm{Cov}[t^{(i)}, t^{(j)}] = \exp\left(-\,5^2(x^{(i)} - x^{(j)})^2\right)$

$\mathrm{Cov}[t^{(i)}, t^{(j)}] = 1 + x^{(i)}x^{(j)} + 0.1^2 \exp\left(-\,3^2(x^{(i)} - x^{(j)})^2\right)$

$\mathrm{Cov}[t^{(i)}, t^{(j)}] = \exp\left(-\,(x^{(i)} - x^{(j)})^2\right) + 0.1^2 \exp\left(-\,5^2(x^{(i)} - x^{(j)})^2\right)$

Figure 1. *Functions drawn from Gaussian processes with various covariance functions.*

For problems with more than one input variable, the σ_u and ρ_u parameters control the degree to which each input is relevant to predicting the target. If ρ_u is small, input u will have only a small effect on the degree of covariance between cases (or at least, a small effect on the portion due to the exponential part in which this ρ_u hyperparameter occurs). Two cases could then have high covariance even if they have substantially different values for input u.

2.3 *Priors for hyperparameters*

In typical applications, the constant part of the covariance and possibly the jitter part (if present) would have fixed values, but the available prior information would not be sufficient to fix the other hyperparameters in the covariance function. These hyperparameters will often be given prior distributions that are fairly vague, but they should not be improper, since this will often produce an improper posterior. For hyperparameters that come in groups, such as the ρ_u in an exponential part, or the σ_u in a linear part, a hierarchical prior could be used, in which a higher-level hyperparameter determines the mean for the hyperparameters within the group.

Models based on covariance functions in which the unknown hyperparameters have been given suitable priors can discover high-level structure in the data, such as which of the inputs are relevant, and whether or not an additive form for the function is appropriate. For instance, if there are two input variables, the following covariance function might be used:

$$\begin{aligned}\mathrm{Cov}[t^{(i)}, t^{(j)}] &= \eta_1^2 \exp\left(-\ \rho_1^2 (x_1^{(i)} - x_1^{(j)})^2\right) \ +\ \eta_2^2 \exp\left(-\ \rho_2^2 (x_2^{(i)} - x_2^{(j)})^2\right) \\ &\quad +\ \eta_*^2 \exp\left(-\ \rho_{*1}^2 (x_1^{(i)} - x_1^{(j)})^2 - \rho_{*2}^2 (x_2^{(i)} - x_2^{(j)})^2\right) \ +\ \delta_{ij}\sigma_\epsilon^2\end{aligned} \tag{12}$$

Depending on what values for η_1, η_2, and η_* are favoured by the data, this model might produce a function that is univariate (looking at only one input), that has an additive form, that is of general non-additive form, or that (for example) is a sum of a large additive part and a small non-additive part. Inference for the continuously-variable hyperparameters in such a model is generally both easier and more realistic than the alternative of attempting to chose among discrete models having different structures.

3. LIMITATIONS OF GAUSSIAN PROCESSES

It is important to keep in mind that Gaussian processes are not appropriate priors for all problems. For example, a belief that the function being modeled has a discontinuity at some unknown location cannot be expressed in terms of a Gaussian process prior. Functions whose degree of smoothness or scale of variation vary as a function of the inputs, in an unknown fashion, are also problematical.

One approach to such problems is to abandon Gaussian processes, perhaps in favour of models based on non-Gaussian stable distributions, which can be approximated by neural networks (Neal 1996). However, one can also keep using a Gaussian process at the lowest level of the model, but introduce hyperparameters into the mean or covariance function of the Gaussian process, so that the final prior (integrating over these hyperparameters) is non-Gaussian. This is in fact what has been done above, but only for hyperparameters that have global effects, such as determining the relevance of inputs. Modeling features such as discontinuities requires hyperparameters whose effects vary over the input space, producing a non-stationary Gaussian process. This approach has been used by Gibbs (1997) to model spatially-varying length scales.

4. COMPUTATIONS FOR GAUSSIAN PROCESS MODELS

Making predictions from a Gaussian process regression model with known covariance function requires only the matrix operations in equations (3) and (4). For classification models, it is also necessary to integrate over the latent values in the observed cases, which can be done using Markov chain Monte Carlo methods. When hyperparameters in the covariance function are unknown, they too can be sampled in the Markov chain. These computations can take substantial time, so it is worthwhile examining them in some detail.

4.1 *Matrix computations*

The central object in Gaussian process computations is the n by n covariance matrix of the targets or latent values for the observed cases. This covariance matrix, which I will call C, depends on the observed inputs for these cases, and on the particular values of the hyperparameters, both of which are considered fixed here. The difficulty of computations involving C is determined by its condition number --- the ratio of its largest eigenvalue to its smallest eigenvalue. If the condition number is large, round-off error in the matrix computations may cause them to fail or to be highly inaccurate.

This potential problem can be controlled by using covariance functions that include ''jitter'' terms, since the jitter contributes additively to every eigenvalue of the matrix, reducing the condition number. When C is the covariance matrix for the targets in a regression model, the noise variance has an equivalent effect. Adding a small amount of jitter to the covariance function usually has little effect on the statistical properties of the model. Problems of poor accuracy due to bad conditioning therefore seem to be largely avoidable. One point to note, however, is that accuracy will be reduced if the constant part in the covariance is large. There will be no need for a large constant part in a regression model if the targets have been shifted to have a mean of approximately zero.

The first step in computations involving C is to find its Cholesky decomposition: the lower-triangular matrix, L, for which $C = LL^T$. The Cholesky decomposition can be found by a simple algorithm (see, for example, Thisted 1988, Section 3.3), which takes time proportional to n^3. One use of the Cholesky decomposition is in generating latent or target values from the prior. Standard methods can be used to randomly generate a vector, z, composed of n independent Gaussian variates with mean zero and variance one. One can then compute the vector Lz, which will have mean zero and covariance matrix $LL^T = C$. This procedure was used to produce the plots in Figure 1, using the covariance matrix for the targets over a grid of input values, with the addition of an unnoticeable amount of jitter. The primary use of the Cholesky decomposition is in computing the inverse of C, or products such as $C^{-1}y$, which are needed to make predictions for new cases, and to find the log likelihood and its derivatives.

The predictive mean and variance in a new case for a target (for a regression model) or a latent value (for a classification model) can be found as follows. Compute the vector k of covariances between the targets or latent values in the observed cases and the target or latent value in the new case, then use forward substitution (see, for example, Thisted 1988, Section 3.2) to compute $u = L^{-1}k$ and $v = L^{-1}y$, where y is the vector of targets or latent values for the observed cases. The predictive mean for the new case can then be computed as $u^T v = k^T C^{-1} y$. The predictive variance is $V - u^T u$, where V is the prior variance for the target or latent value in the new case (for a regression model, we include the noise variance in V to get the variance of the actual target, or leave it out to find the uncertainty in the regression function at that point). Performing these computations for a new case takes time proportional to n^2, where n is the number of observed cases, once the initial computation of the Cholesky decomposition has been done.

If we are interested only in the predictive mean for new cases, we can instead pre-compute $w = C^{-1}y$ from the Cholesky decomposition by forward and backward substitution, after which the predictive mean for a new case can be computed as $k^T w$ in time proportional to n. This method may be more sensitive to numerical errors, however (Gibbs 1997).

For classification models, we will always need both the mean and the variance for the latent values, since predictions for the target are found by integrating over this distribution. This can be done in various ways, of which the simplest is to just sample one or more points from this Gaussian distribution for the latent values, and use the resulting class probabilities in the Monte Carlo averaging procedure that will be needed in any case to integrate over the posterior distribution of the latent values in the observed cases and of the hyperparameters.

Markov chain methods for sampling from the posterior distribution of the hyperparameters will require computation of the log likelihood, given for a regression model by equation (6). Once the Cholesky decomposition, $C = LL^T$, has been found, $(1/2) \log \det C$ can easily be computed as the sum of the logs of the diagonal elements of L and $t^T C^{-1} t$ can be computed as $u^T u$, where $u = L^{-1}t$ is computed by forward substitution. For a classification model, t is replaced by the vector of latent values, y.

For some of the Markov chain sampling methods, the derivatives of L with respect to the various hyperparameters are also required. The derivative of the log likelihood (for a regression model) with respect to a hyperparameter θ can be written as follows:

$$\frac{\partial L}{\partial \theta} = -\frac{1}{2}\mathrm{tr}\Big(C^{-1}\frac{\partial C}{\partial \theta}\Big) + \frac{1}{2}t^T C^{-1}\frac{\partial C}{\partial \theta}C^{-1}t \tag{13}$$

(For a classification model, y replaces t.) Explicit computaton of C^{-1} here seems unavoidable. It can be found from L by using forward and backward substitution to solve $Cx = e_i$ for the vectors e_i that are zero except for a one in position i. This takes time proportional to n^3, and is unfortunately slower than finding the Cholesky decomposition itself. Once C^{-1} has been found, the trace of the product in the first term can be computed in time proportional to n^2. The second term can also be computed in time proportional to n^2, by first computing $u = L^{-1}t$, then solving $L^T w = u$ by backward substitution, and finally computing $w^T(\partial C/\partial\theta)w$.

For large data sets, the dominant computations are finding the Cholesky decomposition of C and then computing C^{-1} (if derivatives are required), for which the time required grows in proportion to n^3. However, for small data sets (eg, 100 cases), computing the derivatives of C with respect to the hyperparameters can dominate, even though the time for this grows only in proportion to n^2. These computations can be sped up if the individual values for the exponential parts of the covariances have been saved (as these appear in the expressions for the derivatives). When n is small, the memory required to do this is not too large; when n is larger, the other operations dominate anyway.

4.2 *Sampling for latent values*

For classification models, predicting the target in a new case requires integration over the distribution of latent values in the observed cases, which (in conjunction with the hyperparameters) determine the distribution of the latent values in the new case, and hence of the class probabilities. This integration can be done by sampling latent values for the observed cases using a Markov chain, and then averaging the predictions for new cases based on several states from this chain.

Gibbs sampling is one way of defining a Markov chain to sample for the latent values. We can scan the latent values in the observed cases sequentially, updating each by drawing a new value for it from its conditional distribution given the observed target for that case, the other latent values, and the hyperparameters. (For multi-class models with several latent values in each case, it makes no difference whether the inner loop of this scan is over cases or over values within a case.) The density for this conditional distribution is proportional to the product of the probability of the target given the latent variable (equation (7) or (8)) and the conditional density given the other latent values. This conditional density is proportional to $\exp(-\frac{1}{2}y^T C^{-1} y)$, where one of the elements of y is the latent value being updated, and the others elements are the current values of the other latent values. The final conditional density is log-concave, and hence can be sampled from efficiently using the adaptive rejection method of Gilks and Wild (1992).

Once C^{-1} has been computed, in time proportional to n^3, updating a latent value takes time proportional to n, and a complete Gibbs sampling scan takes time proportional to n^2. When hyperparameters are being updated also (see Section 4.3), it makes sense to perform quite a few Gibbs sampling scans between each hyperparameter update (which changes C), as this adds little to the computation time, and probably makes the Markov chain mix faster.

Unfortunately, Gibbs sampling does not work well when one or more of the latent values are almost determined by the other latent values (ie, have tiny conditional variances). If the amount of jitter used is small, this will be a problem whenever the amount of data is fairly large, or when some cases have identical or near-identical inputs (even if the dataset is small). The problem can be alleviated by using a larger amount of jitter, but this changes the characteristics of the model somewhat.

This problem can be avoided by updating the latent variables using the Metropolis-Hastings algorithm, with a proposal distribution that changes all the latent values simultaneously (or for K-way classification, perhaps only all the latent values associated with one of the classes). A suitable proposal can be found using the prior distribution for the latent values, since this prior incorporates the correlations that are expected. One method is to propose a change to the latent values that is a scaled down sample from the prior --- ie, we propose to change the vector of latent values, y, to

$$y' = y + \epsilon L z \tag{14}$$

where L is the Cholesky decomposition of the prior covariance, z is a vector of independent standard Gaussian variates, and ϵ is some small constant. We then accept or reject this proposal in the usual Metropolis fashion, according to the change in the product of the Gaussian prior density for y and the probability of the observed targets given y. Another approach, which is a bit faster and seems to work somewhat better, is to propose the following change:

$$y' = (1-\epsilon^2)^{1/2} y + \epsilon L z \tag{15}$$

This proposal distribution satisfies detailed balance with respect to the Gaussian prior for y, and is therefore accepted or rejected based only on the change in the probability of the targets given y. For both methods, it is important that the covariance function for the latent values not have an excessively large constant part, as the proposed change would then be almost the same for all latent values, which is usually unsuitable. As with Gibbs sampling, each update takes only n^2 time, so it makes sense to do many updates after each change to the hyperparameters.

When the amount of jitter is quite small, both Metropolis-Hastings methods are many times faster than Gibbs sampling. Unfortunately, these methods do require that ϵ be chosen appropriately, based on trial runs if necessary.

Many other ways of sampling latent variables are possible. Wood and Kohn (1998) describe a scheme for a similar model based on splines in which the latent values, y, are updated by first choosing values for a second set of latent values, z, which are the ''jitter-free'' values of y. This can be done directly, in the same way as predictions are made in a regression model. (It is sometimes necessary to let z retain a small portion of the jitter, to improve the conditioning of the matrix operations involved.) Each component of y can then be updated independently, based on the corresponding component of z and on the target class. Unfortunately, this scheme is about as slow as Gibbs sampling when the amount of jitter used is small. Other possibilities are some form of overrelaxed Gibbs sampling, and hybrid Monte Carlo (Duane, *et al* 1987), perhaps with a ''kinetic energy'' defined in terms of C. Sampling values for $w = C^{-1}y$ instead of for y itself could also be considered.

4.3 *Sampling for hyperparameters*

The covariance function for a Gaussian process model will typically contain unknown hyperparameters, which we must integrate over in a fully Bayesian treatment. The number of hyperparameters will vary from around three or four for a very simple regression model up to several dozen or more for a model with many inputs, whose relevances are individually controlled using hyperparameters such as the ρ_u. Integration over the hyperparameters can be done by Markov chain Monte Carlo.

For regression models, the hyperparameters are updated based on their prior and on the likelihood derived from the target values (equation (6)). For classification models, Markov chain updates for hyperparameters are interleaved with updates for the latent values using one of the schemes described in Section 4.2. The updates for the hyperparameters will use the likelihood based on the current latent values (analogous to equation (6) but with y instead of t). The targets influence the hyperparameters only indirectly, through their influence on the latent values.

Sampling from the posterior distribution of the hyperparameters is facilitated by representing them in logarithmic form, as this eliminates constraints, and makes the sampling methods insensitive to the scale of the data. Gibbs sampling cannot be applied to this problem, since it is very difficult to sample from the required conditional distributions. The Metropolis algorithm could be used with some simple proposal distribution, such as a Gaussian with diagonal covariance matrix. However, simple methods such as this explore the region of high probability by an inefficient random walk. It is probably better to use a method that can suppress these random walks, such as hybrid Monte Carlo (Duane, *et al* 1987) or one of its variants. I have reviewed hybrid Monte Carlo elsewhere (Neal 1993, 1996), and its use for Gaussian process models is described by Williams and Rasmussen (1996), Rasmussen (1996), and myself (Neal 1997).

I have previously applied hybrid Monte Carlo to Bayesian inference for neural networks, where it can be hundreds or thousands of times faster than simple versions of the Metropolis algorithm (Neal 1996). When applied to Gaussian process models of moderate complexity, the benefits of using hybrid Monte Carlo are not quite so large, probably due both to the lower dimensionality of the problem, and to smaller dependencies between state variables. Since hybrid Monte Carlo is based on Hamiltonian dynamics, it requires the derivatives of the likelihood with respect to the hyperparameters. This involves some additional computation, but hybrid Monte Carlo seems nevertheless to be the most efficient method, provided that its ''stepsize'' parameters are set properly, based on guesses as to the shape and concentration of the posterior distribution for the hyperparameters. Some heuristics for setting the stepsizes have been developed (Neal 1997), but better ones would be desirable, and might be found if the ''fill-in'' asymptotics of the posterior distribution were examined more closely.

In my experience, Markov chain samplers for Gausian process models do not usually become stuck in local modes. However, they do sometimes spend a long time initially in states where the data is fit poorly, before finding by chance a combination of hyperparameter values that leads to a good fit. This can happen, for example, if there are many inputs, only a few of which are relevant. Finding the right combination of relevant inputs then becomes a difficult combinatorial search problem. Using realistic priors helps narrow this search somewhat.

5. EXAMPLE: A THREE-WAY CLASSIFICATION PROBLEM

I will demonstrate Gaussian process modeling on a synthetic three-way classification problem. Pairs of data items, $(x^{(i)}, t^{(i)})$ were generated by first independently drawing $\tilde{x}_1^{(i)}$, $\tilde{x}_2^{(i)}$, $\tilde{x}_3^{(i)}$, $\tilde{x}_4^{(i)}$ uniformly from $(0,1)$. The class of the item, $t^{(i)}$, encoded as 0, 1, or 2, was then selected as follows: If the Euclidean distance of $(\tilde{x}_1^{(i)}, \tilde{x}_2^{(i)})$ from $(0.4, 0.5)$ was less than 0.35, the class was 0; otherwise, if $(0.8)\tilde{x}_1^{(i)} + (1.8)\tilde{x}_2^{(i)}$ was less than 0.6, the class was 1; and if neither of these conditions held, the class was 2. Note that $\tilde{x}_3^{(i)}$ and $\tilde{x}_4^{(i)}$ have no effect on the class. The inputs, $x_1^{(i)}$, $x_2^{(i)}$, $x_3^{(i)}$, $x_4^{(i)}$, available for prediction of the target were $\tilde{x}_1^{(i)}$, $\tilde{x}_2^{(i)}$, $\tilde{x}_3^{(i)}$, $\tilde{x}_4^{(i)}$ plus independent Gaussian noise of standard deviation 0.1. I generated 1000 cases in this way, of which 400 were used for training the model, and 600 for testing the resulting predictive performance. The 400 training cases are shown in Figure 2, plotted according to $x_1^{(i)}$ and $x_2^{(i)}$.

This data was modeled using a Gaussian process for the latent values, $y^{(i)}$, whose covariance function consisted of three terms --- a constant part (fixed at 10^2), an exponential part in which the magnitude, η, and the scales for the four inputs, ρ_u, were variable hyperparameters, and a jitter part, fixed at $J = 10$. The fairly large amount of jitter produces an effect close to a probit model (Neal 1997). Since each of the ρ_u can vary separately (under the control of a common higher-level hyperparameter), the model is capable of discovering that some of the inputs are in fact irrelevant to the task of predicting the target. We hope that the posterior distribution of ρ_u for these irrelevant inputs will be concentrated near zero, so that they will not degrade predictive performance.

Class 0 = filled square
Class 1 = plus sign
Class 2 = open triangle

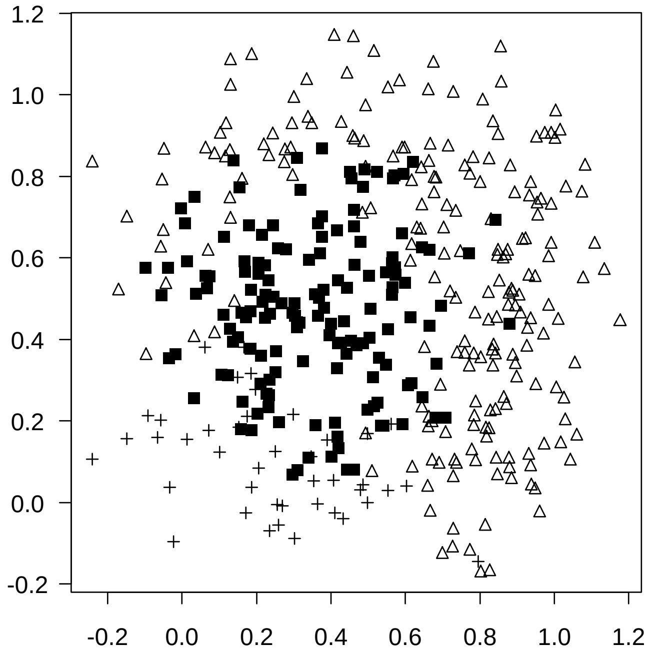


Figure 2. *The 400 training cases used for the three-way classification problem.*

The hyperparameters for this model were sampled using a form of hybrid Monte Carlo, and the latent values associated with training cases were updated using Gibbs sampling. Details on the model and the sampling procedure used can be found in the documentation for the Gaussian process software that I distribute via my web page, where this problem is used as an example.

The convergence of the Markov chain simulation can be assessed by plotting how the values of the hyperparameters change over the course of the simulation. Figure 3 shows the progress of the ρ_u hyperparameters in the exponential part of the covariance, over 100 iterations of the Markov chain simulation (each iteration involves several hybrid Monte Carlo and Gibbs sampling updates). As hoped, we see that by about iteration 50, an apparent equilibrium has been reached in which the hyperparameters ρ_3 and ρ_4, associated with the irrelevant inputs, have values that are much smaller than those for ρ_1 and ρ_2, which are associated with the inputs that provide information about the target class.

Predictions for test cases were made by averaging the predictive probabilities based on iterations after equilibrium was apparently reached. To reduce computation time, only every fifth iteration was used, starting at iteration 55 (for a total of ten iterations). For each such iteration, the predictive mean and variance for the latent values in each of the 600 test cases were found, using the saved values of the hyperparameters and of the latent values for the observed cases. A sample of 100 points from this predictive distribution was used to produce a Monte Carlo estimate of the predictive probabilities for the three classes. These probabilities were averaged over the iterations used to produce the final predictive probabilities. The class in each test case was then guessed to be the one for which this predictive probability was largest.

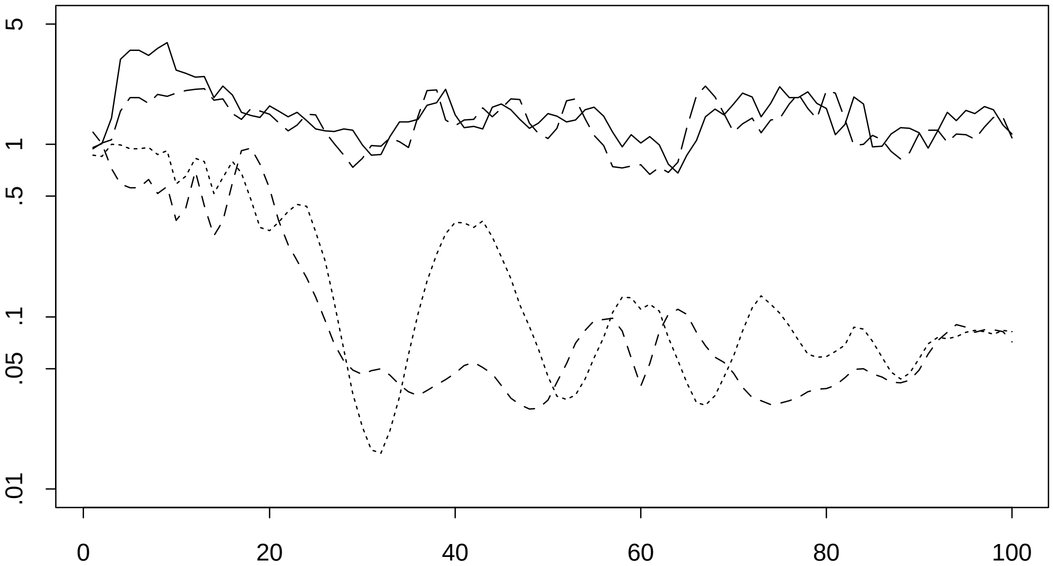


Figure 3. *Relevance hyperparameters:* ρ_1 *= solid,* ρ_2 *= long dash,* ρ_3 *= short dash,* ρ_4 *= dotted.*

Simulating the Markov chain for 100 iterations took about 90 minutes on our SGI machine (with a 194 MHz R10000 processor). Making predictions for the 600 test cases took about 3 minutes. The classification error rate on these 600 test cases was 13% (close to that of an analogous Bayesian neural network model). The time required varies considerably with the number of cases, due to the n^3 scaling for the matrix operations. Using only the first 100 cases, the Markov chain simulation took about 7 minutes, and prediction for the test cases took 13 seconds. The resulting classification error rate was 19%.

6. HISTORY OF METHODS RELATED TO GAUSSIAN PROCESSES

Gaussian processes are a standard tool in probability theory, and have long been used as models in some application areas. Users of Gaussian process models in different fields, or from different schools, often handle problems such as selecting an appropriate covariance function in different ways, however, and may have different views as to the meaning of a Gaussian process model. Completely Bayesian viewpoints are not very common.

Given the assumption of Gaussian noise, many standard parametric time series models define Gaussian processes. However, it is traditional in time series work to avoid models with general covariance functions, for which there are no computational shortcuts. A more fundamental difference compared to the models discussed here is that a time series model can often be plausibly regarded as describing a real-world process, which might in principle operate indefinitely. This allows one to take a frequentist view of the Gaussian process model.

The method of ‘‘kriging’’ in spatial statistics can also be viewed as prediction based on a Gaussian process. The standard approach in this field is to use fairly general covariance functions, estimated from the data. To me, it seems difficult to view these Gaussian process models in other than Bayesian terms --- that is, as defining a degree-of-belief prior distribution for the unknown surface. This interpretation is by no means universal, however, though Bayesian approaches to kriging are becoming more common (eg, Handcock and Stein 1993). Gaussian process models have also been used to analyse noise-free data from ‘‘computer experiments’’ (eg, Sacks, *et al* 1989).

Smoothing splines (see Wahba 1990) are widely used as flexible univariate regression models. They can be viewed as the predictive mean for a Gaussian process with a particular (improper) covariance function, whose form allows for computational shortcuts. Generalizations to more than one dimension are possible, though the shortcuts no longer apply. In recent work (Wahba, *et al* 1995), structured multi-dimensional models are considered, similar to the Gaussian process models with covariance functions built up from several terms that were discussed in Section 2.3.

A smoothing spline can also be seen as a maximum penalized likelihood estimate, which for a regression model with Gaussian errors coincides with the Bayesian predictive mean due to the ''accident'' that the mean and the mode of a Gaussian distribution coincide. A penalization (''regularization'') approach is also taken in the literature on Radial Basis Function networks (eg, Girosi, Jones, and Poggio 1995), some types of which are also equivalent to Gaussian process models. However, this correspondence between regularized estimates and Bayesian predictive means for targets does not hold for classification and other non-Gaussian models.

The smoothing parameter for splines (corresponding to a hyperparameter of the covariance function) is usually found by generalized cross validation (GCV), rather than by the more Bayesian method of generalized maximum likelihood (GML). Wahba (1990) argues that GCV is superior to GML when the true function is not drawn from the Gaussian process corresponding to the spline. If one adopts the frequentist approach in this respect, the Gaussian process interpretation is perhaps not useful, since the the probability distribution over functions that it defines would appear to have no meaning.

Wholeheartedly Bayesian uses of Gaussian processes also go back many years (eg, O'Hagan 1978). This early work was confined to regression models, however, and did not attempt to find the hyperparameters of the covariance function by Bayesian methods. More recently, Upsdell (1996) has discussed how to choose a covariance function suitable for an application, with hyperparameters set by GML. Hsu and Leonard (1997) do a full Bayesian analysis of binomial data with a Gaussian process prior, implemented using importance sampling.

My own interest in Gaussian process models came from considering Bayesian models based on neural networks (Neal 1996), which for a wide class of priors converge to Gaussian process models as the number of ''hidden units'' in the network goes to infinity. This inspired Williams and Rasmussen (1996) to look at models implemented directly in terms of Gaussian processes, with hyperparameters either set by GML, or integrated over using Markov chain Monte Carlo. Rasmussen (1996) found that such Gaussian process regression models were superior to several other methods (including MARS, and its ''bagged'' variant, and neural networks trained by ''early stopping'') in empirical tests on an array of problems. Approximate methods for handling Gaussian process classification models have been developed by Barber and Williams (1997) and by Gibbs (1997). MacKay (1997) and Williams (1998) have recently reviewed work on Gaussian processes and their connections with neural networks.

7. WHY AREN'T GAUSSIAN PROCESS MODELS COMMONLY USED?

Despite their long history and conceptual simplicity, general-purpose Gaussian process models are not commonly used. One reason for this is undoubtably the n^3 computation time required, which is of course multiplied many times when the hyperparameters are integrated over in Bayesian fashion, or when latent values underlying a classification model must be handled. Computational cost does not fully explain the neglect of these models, however. For small datasets, these models would have been practical even with the computers of 10 or 20 years ago. From the standpoint of statistical research, the simplicity of these models would seem to make them useful reference points even apart from any practical applications. Finally, one can always seek to reduce the computational costs using approximations or iterative matrix computations, as has recently been done by Gibbs (1997).

I speculate that a more fundamental reason for the neglect of Gaussian process models is a widespread preference for simple models, resulting from a confusion between prior beliefs regarding the true function being modeled and expectations regarding the properties of the best predictor for this function (the posterior mean, under squared error loss). These need not be at all similar. For example, our beliefs about the true function might sometimes be captured by an Ornstein-Uhlenbeck process, a Gaussian process with covariance function $\exp(-|x^{(i)}-x^{(j)}|)$. Realizations from this process are nowhere differentiable, but the predictive mean function will consist of pieces that are sums of exponentials, as can be seen from equation (3).

Figure 4 shows a perhaps more relevant example. The dotted functions there are from the posterior distribution given four data points of a Gaussian process model with covariance function $10^2 + \eta^2 \exp(-\rho^2(x^{(i)}-x^{(j)})^2) + \delta_{ij}\sigma_\epsilon^2$. The hyperparameters were given gamma priors as follows: $1/\eta^2$ had mean $1/0.3^2$ and shape parameter 1, $1/\rho^2$ had mean $1/0.3^2$ and shape 1, and $1/\sigma_\epsilon^2$ had mean $1/0.1^2$ and shape 5. As can be seen, the posterior distribution includes some functions that are compatible with the data, but which possess features for which the data provide no positive support. However, the predictive mean function, shown as the dark line, has a simpler appearance, since the complexities present in some of the functions drawn from the posterior largely cancel when they are averaged to give the mean function.

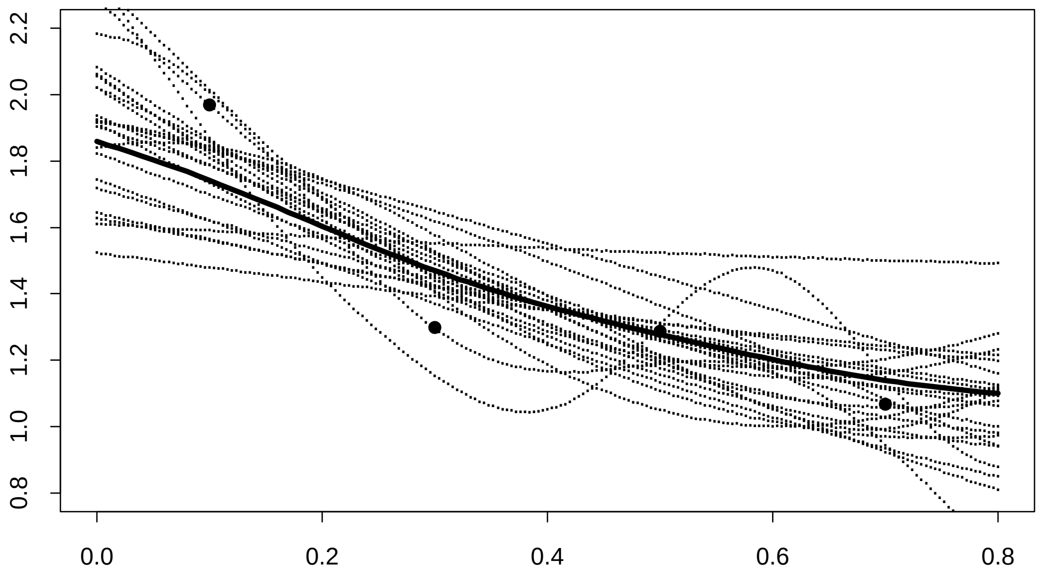


Figure 4. *Predictive mean and functions from the posterior given four data points*

Inappropriately treating intuitions regarding the properties of the best predictor as reflecting prior beliefs regarding the true function can lead to models that are nonsensical from a Bayesian perspective. Consider, for example, a model based on regression splines with a finite number of knots, with the number of knots being given some prior distribution. This model is fine if you believe that the true function is a piecewise polynomial. However, such models have been used in contexts where this is apparently not believed, but where it is instead expected that the posterior distribution will favour a larger number of knots when the amount of data is larger, to allow for closer modeling of the true function. This expectation is inconsistent with a Bayesian interpretation of the procedure as expressing real prior knowledge, so its use could only be justified on frequentist grounds (if at all).

Distinguishing between the properties expected of the true function and of the best predictor for this function is important not only for Gaussian processes, but also for other flexible models, such as those based on neural networks and on mixtures of Gaussians. There is often a tendency to prefer simple forms of such models (eg, with few mixture components), a preference that can be given a Bayesian formulation using a prior over model complexity. However, this makes no sense if there is every reason to believe that the true function cannot be well approximated with a simple model. With the right prior for a suitably complex model, the predictive distribution based on limited data will have a simple form if that is in fact appropriate.

ACKNOWLEDGEMENTS

I thank David MacKay, Carl Rasmussen, and Chris Williams for many discussions about Gaussian process models. This work was supported by the Natural Sciences and Engineering Research Council of Canada.

REFERENCES

Barber, D. and Williams, C. K. I. (1997) ''Gaussian processes for Bayesian classification via hybrid Monte Carlo'', in M. C. Mozer, *et al* (editors) *Advances in Neural Information Processing Systems 9*, MIT Press.

Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. (1987) ''Hybrid Monte Carlo'', *Physics Letters B*, vol. 195, pp. 216-222.

Gibbs, M. N. (1997) *Bayesian Gaussian Processes for Regression and Classification*, Ph.D. thesis, Inferential Sciences Group, Cavendish Laboratory, Cambridge University. Available in Postscript from `http://wol.ra.phy.cam.ac.uk/mng10/GP/`.

Gilks, W. R. and Wild, P. (1992) ''Adaptive rejection sampling for Gibbs sampling'', *Applied Statistics*, vol. 41, pp. 337-348.

Girosi, F., Jones, M., and Poggio, T. (1995) ''Regularization theory and neural network architectures'', *Neural Computation*, vol. 7, pp. 219-269.

Goldberg, P. W., Williams, C. K. I., and Bishop, C. M. (1998) ''Regression with input-dependent noise: A Gaussian process treatment'', in M. I. Jordan, *et al* (editors) *Advances in Neural Information Processing Systems 10*.

Handcock, M. S. and Stein, M. L. (1993) ''A Bayesian analysis of kriging'', *Technometrics*, vol. 35, pp. 403-410.

Hsu, J. S. J. and Leonard, T. (1997) ''Hierarchical Bayesian semiparametric procedures for logistic regression'', *Biometrika*, vol. 84, pp. 85-93.

MacKay, D. J. C. (1997) ''Gaussian processes: A replacement for supervised neural networks?'', preprint. Available in Postscript from `http://wol.ra.phy.cam.ac.uk/mackay/`.

Neal, R. M. (1993) *Probabilistic Inference Using Markov Chain Monte Carlo Methods*, Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto, 140 pages.

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'', Technical Report No. 9702, Dept. of Statistics, University of Toronto, 24 pages.

O'Hagan, A. (1978) ''Curve fitting and optimal design for prediction'' (with discussion), *Journal of the Royal Statistical Society B*, vol. 40, pp. 1-42.

Rasmussen, C. (1996) *Evaluation of Gaussian Processes and other Methods for Non-Linear Regression*, Ph.D. Thesis, Department of Computer Science, University of Toronto. Available in Postscript from `http://www.cs.utoronto.ca/~carl/`.

Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, P. (1989) ''Design and analysis of computer experiments'' (with discussion), *Statistical Science*, vol. 4, pp. 409-435.

Thisted, R. A. (1988) *Elements of Statistical Computing*, New York: Chapman and Hall.

Upsdell, M. P. (1996) ''Choosing an appropriate covariance function in Bayesian smoothing'', in J. M. Bernardo, *et al* (editors) *Bayesian Statistics 5*, pp. 747-756, Oxford University Press.

Wahba, G. (1990) *Spline Models for Observational Data*, Philadelphia: SIAM.

Williams, C. K. I. and Rasmussen, C. E. (1996) ''Gaussian processes for regression'', in D. S. Touretzky, *et al* (editors) *Advances in Neural Information Processing Systems 8*, MIT Press.

Williams, C. K. I. (1998) ''Prediction with Gaussian processes: from linear regression to linear prediction and beyond'', in M. I. Jordan (editor), *Learning in Graphical Models*, Kluwer.

Wood, S. and Kohn, R. (1998) ''A Bayesian approach to robust binary nonparametric regression'', *Journal of the American Statistical Association*, vol. 93, pp. 203-213.