INTRODUCTION TO GAUSSIAN PROCESSES

DAVID J.C. MACKAY
Department of Physics, Cambridge University.
Cavendish Laboratory, Madingley Road,
Cambridge, CB3 0HE. United Kingdom.
`mackay@mrao.cam.ac.uk`

Abstract. Feedforward neural networks such as multilayer perceptrons are popular tools for nonlinear regression and classification problems. From a Bayesian perspective, a choice of a neural network model can be viewed as defining a prior probability distribution over non-linear functions, and the neural network's learning process can be interpreted in terms of the posterior probability distribution over the unknown function. (Some learning algorithms search for the function with maximum posterior probability and other Monte Carlo methods draw samples from this posterior probability).

In the limit of large but otherwise standard networks, Neal (1996) has shown that the prior distribution over non-linear functions implied by the Bayesian neural network falls in a class of probability distributions known as Gaussian processes. The hyperparameters of the neural network model determine the characteristic lengthscales of the Gaussian process. Neal's observation motivates the idea of discarding parameterized networks and working directly with Gaussian processes. Computations in which the parameters of the network are optimized are then replaced by simple matrix operations using the covariance matrix of the Gaussian process.

In this chapter I will review work on this idea by Williams and Rasmussen (1996), Neal (1997), Barber and Williams (1997) and Gibbs and MacKay (1997), and will assess whether, for supervised regression and classification tasks, the feedforward network has been superceded.

FOREWORD

My lectures on neural networks and Gaussian processes feature a sequence of computer demonstrations written in the free language `octave`. The source

code is available at:
`http://wol.ra.phy.cam.ac.uk/mackay/`.

Mark Gibbs's software for Gaussian processes is available at:
`http://wol.ra.phy.cam.ac.uk/mng10/GP/GP.html`.

Radford Neal's is at `http://www.cs.toronto.edu/~radford/`.

1. Overview

Since the publication of Rumelhart, Hinton and Williams's (1986) paper on supervised learning in neural networks there has been a surge of interest in the empirical modelling of relationships in high–dimensional data using nonlinear parametric models such as multi–layer perceptrons and radial basis functions. In the Bayesian interpretation of these modelling methods, a nonlinear function $y(\mathbf{x})$ parameterized by parameters $\mathbf{w}$ is assumed to underlie the data $\{\mathbf{x}^{(n)}, t_n\}_{n=1}^N$, and the adaptation of the model to the data corresponds to an *inference* of the function given the data. We will denote the set of input vectors by $\mathbf{X}_N \equiv \{\mathbf{x}^{(n)}\}_{n=1}^N$ and the set of corresponding target values by the vector $\mathbf{t}_N \equiv \{t_n\}_{n=1}^N$. The inference of $y(\mathbf{x})$ is described by the posterior probability distribution

$$P(y(\mathbf{x})|\mathbf{t}_N, \mathbf{X}_N) = \frac{P(\mathbf{t}_N|y(\mathbf{x}), \mathbf{X}_N)P(y(\mathbf{x}))}{P(\mathbf{t}_N|\mathbf{X}_N)}. \tag{1}$$

Of the two terms on the right hand side, the first, $P(\mathbf{t}_N|y(\mathbf{x}), \mathbf{X}_N)$, is the probability of the target values given the function $y(\mathbf{x})$, which in the case of regression problems is often implicitly assumed to be a separable Gaussian distribution; and the second term, $P(y(\mathbf{x}))$, is the prior distribution on functions assumed by the model. This prior is implicit in the choice of parametric model and the choice of regularizers used during the model adaptation. The prior typically specifies that the function $y(\mathbf{x})$ is expected to be continuous and smooth, having less high frequency power than low frequency power, but the precise meaning of the prior is somewhat obscured by the use of the parametric model.

Now, from the point of view of prediction of future values of t, all that matters is the assumed prior $P(y(\mathbf{x}))$ and the assumed noise model — the parameterization of the function $y(\mathbf{x}; \mathbf{w})$ is irrelevant.

The idea of Gaussian process modelling is, without parameterizing $y(\mathbf{x})$, to place a prior $P(y(\mathbf{x}))$ directly on the space of functions. The simplest type of prior over functions is called a Gaussian process. It can be thought of as the generalization of a Gaussian distribution over a finite vector space to a function space of infinite dimension. Just as a Gaussian distribution is fully specified by its mean and covariance matrix, a Gaussian process is specified by a mean and a covariance function. Here, the mean is a function of $\mathbf{x}$ (which we will often take to be the zero function), and the covariance is a function $C(\mathbf{x}, \mathbf{x}')$ which expresses the expected covariance between the value of the function y at the points $\mathbf{x}$ and $\mathbf{x}'$. The actual function $y(\mathbf{x})$ in any one data modelling problem is assumed to be a single sample from this Gaussian distribution. Gaussian processes are already well established models for various spatial and temporal problems (Ripley 1991) — for example, Brownian motion, Langevin processes and Wiener processes are all examples of Gaussian processes; Kalman filters, widely used to model speech waveforms, also correspond to Gaussian process models; the method of 'kriging' in geostatistics is a Gaussian process regression method.

1.1. RESERVATIONS ABOUT GAUSSIAN PROCESSES

It might be thought that it is not possible to reproduce the interesting properties of neural network interpolation methods with something so simple as a Gaussian distribution, but as we shall now see, many popular nonlinear interpolation methods are equivalent to particular Gaussian processes. (I will use the term 'interpolation' to cover both the problem of 'regression' — fitting a curve through noisy data — and the task of fitting an interpolant that passes exactly through the given data points.)

It might also be thought that the computational complexity of inference when we work with priors over infinite dimensional functions spaces might be infinitely large. But by concentrating on the joint probability distribution of the observed data and the quantities we wish to predict, it is possible to make predictions with resources that scale as polynomial functions of N, the number of data points.

1.2. SUMMARY

In this chapter we describe how inferences with a Gaussian process work, and how they can be implemented with finite computational resources. We next discuss the role of the hyperparameters controlling a Gaussian process and describe how they can be adapted to data. We then study a variety of different ways in which Gaussian processes can be constructed. We also give an overview of advanced methods using Gaussian processes and discuss their application to classification problems as well as regression problems. It is surprising how much you can do with a single Gaussian distribution!

2. Nonlinear Regression

2.1. THE PROBLEM

We are given N data points $\mathbf{X}_N, \mathbf{t}_N = \{\mathbf{x}^{(n)}, t_n\}_{n=1}^{N}$. The inputs $\mathbf{x}$ are vectors of some fixed input dimension I. The targets t are either real numbers, in which case the task will be a regression or interpolation task, or they are categorical variables, for example $t \in \{0, 1\}$, in which case the task is a classification task. We will concentrate on the case of regression for the time being.

Assuming that a function $y(\mathbf{x})$ underlies the observed data, the task is to infer the function from the given data, and predict its value — or the value of the observation t_{N+1} — at new points $\mathbf{x}^{(N+1)}$.

2.2. PARAMETRIC APPROACHES TO THE PROBLEM

In a parametric approach to regression we express the unknown function $y(\mathbf{x})$ in terms of a nonlinear function $y(\mathbf{x}; \mathbf{w})$ parameterized by parameters $\mathbf{w}$.

Example 1: Fixed basis functions. Using a set of basis functions $\{\phi_h(\mathbf{x})\}_{h=1}^{H}$, we can write

$$y(\mathbf{x}; \mathbf{w}) = \sum_{h=1}^{H} w_h \phi_h(\mathbf{x}). \tag{2}$$

If the basis functions are nonlinear functions of $\mathbf{x}$ such as the following radial basis functions centred at fixed points $\{\mathbf{c}_h\}_{h=1}^{H}$,

$$\phi_h(\mathbf{x}) = \exp\left[-\frac{(\mathbf{x} - \mathbf{c}_h)^2}{2r^2}\right], \tag{3}$$

then $y(\mathbf{x}; \mathbf{w})$ is a nonlinear function of $\mathbf{x}$; however, since the dependence of y on the parameters $\mathbf{w}$ is linear, we might sometimes refer to this as a 'linear' model.

Other possible sets of fixed basis functions include polynomials such as $\phi_h(\mathbf{x}) = x_i^p x_j^q$ where p and q are integer powers that depend on h.

Example 2: Adaptive basis functions. Alternatively, we might make a function $y(\mathbf{x})$ from basis functions which depend on additional parameters included in the vector $\mathbf{w}$. In a two layer feedforward neural network with nonlinear hidden units and a linear output, the function can be written

$$y(\mathbf{x}; \mathbf{w}) = \sum_{h=1}^{H} w_h^{(2)} \tanh\left(\sum_{i=1}^{I} w_{hi}^{(1)} x_i + w_{h0}^{(1)}\right) + w_0^{(2)} \tag{4}$$

where I is the dimensionality of the input space and the weight vector $\mathbf{w}$ consists of the input weights $\{w_{hi}^{(1)}\}$, the hidden unit biases $\{w_{h0}^{(1)}\}$, the output weights $\{w_h^{(2)}\}$ and the output bias $w_0^{(2)}$.

We then infer the function $y(\mathbf{x};\mathbf{w})$ by inferring the parameters $\mathbf{w}$. Most sensible methods for inferring the parameters can be interpreted in terms of a Bayesian model for the problem, in which the posterior probability of the parameters is given by

$$P(\mathbf{w}|\mathbf{t}_N,\mathbf{X}_N)=\frac{P(\mathbf{t}_N|\mathbf{w},\mathbf{X}_N)P(\mathbf{w})}{P(\mathbf{t}_N|\mathbf{X}_N)}. \tag{5}$$

The factor $P(\mathbf{t}_N|\mathbf{w},\mathbf{X}_N)$ states the probability of the observed data points when the parameters $\mathbf{w}$ (and hence, the function y) are known. This probability distribution is often taken to be a separable Gaussian, each data point t_n differing from the underlying value $y(\mathbf{x}^{(n)};\mathbf{w})$ by additive noise. The factor $P(\mathbf{w})$ specifies the prior probability distribution of the parameters. This too is often taken to be a separable Gaussian distribution. If the dependence of y on $\mathbf{w}$ is nonlinear the posterior distribution $P(\mathbf{w}|\mathbf{t}_N,\mathbf{X}_N)$ is not in general a Gaussian distribution.

The inference can be implemented in various ways. One technique is to minimize an objective function

$$M(\mathbf{w})=-\log\left[P(\mathbf{t}_N|\mathbf{w},\mathbf{X}_N)P(\mathbf{w})\right] \tag{6}$$

with respect to $\mathbf{w}$, locating the locally most probable parameters, then use the curvature of M, $\partial^2 M(\mathbf{w})/\partial w_i \partial w_j$ to define error bars on $\mathbf{w}$. A more general method uses Markov chain Monte Carlo techniques to create samples from the posterior distribution $P(\mathbf{w}|\mathbf{t}_N,\mathbf{X}_N)$.

Having obtained one of these representations of the inference of $\mathbf{w}$ given the data, predictions are then made by marginalizing over the parameters:

$$P(t_{N+1}|\mathbf{t}_N,\mathbf{X}_{N+1})=\int d^H\mathbf{w}\; P(t_{N+1}|\mathbf{w},\mathbf{x}^{(N+1)})P(\mathbf{w}|\mathbf{t}_N,\mathbf{X}_N). \tag{7}$$

If we have found a Gaussian representation of the posterior distribution $P(\mathbf{w}|\mathbf{t}_N,\mathbf{X}_N)$, then this integral can typically be evaluated directly. In the alternative Monte Carlo approach which generates R samples $\mathbf{w}^{(r)}$ which are intended to be samples from the posterior distribution $P(\mathbf{w}|\mathbf{t}_N,\mathbf{X}_N)$, we approximate the predictive distribution by

$$P(t_{N+1}|\mathbf{t}_N,\mathbf{X}_{N+1})\simeq\frac{1}{R}\sum_{r=1}^{R}P(t_{N+1}|\mathbf{w}^{(r)},\mathbf{x}^{(N+1)}). \tag{8}$$

2.3. NONPARAMETRIC APPROACHES.

In nonparametric methods, predictions are obtained without giving the unknown function $y(\mathbf{x})$ an explicit parameterization. One well known nonparametric approach to the regression problem is the spline smoothing method (Kimeldorf and Wahba 1970). A spline solution to a one–dimensional regression problem can be described as follows: we define the estimator of $y(\mathbf{x})$ to be the function $\hat{y}(\mathbf{x})$ which minimizes the functional

$$M(y(x)) = -\frac{1}{2}\beta\sum_{n=1}^{N}(y(x^{(n)}) - t_n)^2 - \frac{1}{2}\alpha\int dx\, [y^{(p)}(x)]^2 \qquad (9)$$

where $y^{(p)}$ is the pth derivative of y and p is a positive number. If p is set to 2 then the resulting function $\hat{y}(\mathbf{x})$ is a cubic spline, that is, a piecewise cubic function that has 'knots' — discontinuities in its second derivative — at the data points $\{x^{(n)}\}$.

This estimation method can be turned into a Bayesian method by identifying the prior for the function $y(x)$ as:

$$\log P(y(x)|\alpha) = -\frac{1}{2}\alpha\int dx\, [y^{(p)}(x)]^2 + \text{const}, \qquad (10)$$

and the probability of the data measurements $\mathbf{t}_N = \{t_n\}_{n=1}^N$ assuming independent Gaussian noise as:

$$\log P(\mathbf{t}_N|\, y(x), \beta) = -\frac{1}{2}\beta\sum_{n=1}^{N}(y(x^{(n)}) - t_n)^2 + \text{const}. \qquad (11)$$

[The constants in equations (10) and (11) are functions of α and β respectively. Strictly the prior (10) is improper since addition of an arbitrary polynomial of degree $p-1$ to $y(x)$ is not constrained. This impropriety is easily rectified by the addition of $(p-1)$ appropriate terms to (10).] Given this interpretation of the functions in equation (9), $M(y(x))$ is equal to the log of the posterior probability $P(y(x)|\mathbf{t}_N, \alpha, \beta)$, within an additive constant, and the splines estimation procedure can be interpreted as yielding a Bayesian MAP estimate. The Bayesian approach allows us additionally to put error bars on the splines estimate and to draw typical samples from the posterior distribution. The issue of setting the hyperparameters α and β is an important one, as reviewed in MacKay (1992).

2.4. COMMENTS

2.4.1. *Splines priors are Gaussian processes*

The prior distribution defined in equation (10) is in fact our first example of a Gaussian process. Throwing mathematical precision to the winds, a

Gaussian process can be defined as a probability distribution on a space of functions $y(x)$ which can be written in the form

$$P(y(x)|\mu(x), A) = \frac{1}{Z} \exp\left[-\frac{1}{2}(y(x) - \mu(x))^{\mathrm{T}} A (y(x) - \mu(x))\right], \tag{12}$$

where $\mu(x)$ is the mean function of the distribution and A is a linear operator, and where the inner product of two functions $y(x)^{\mathrm{T}} z(x)$ is defined by, for example, $\int dx\, y(x) z(x)$. Here, if we denote by D the linear operator which maps $y(x)$ to the derivative of $y(x)$, we can write

$$\log P(y(x)|\alpha) = -\frac{1}{2}\,\alpha \int dx\, [D^p y(x)]^2 + \text{const} = -\frac{1}{2}\, y(x)^{\mathrm{T}} A y(x) + \text{const}, \tag{13}$$

which has the same form as (13) with $\mu(x) = 0$, and $A \equiv [D^p]^{\mathrm{T}} D^p$.

In order for the prior in equation (12) to be a proper prior, A must be a positive definite operator, *i.e.*, one satisfying $y(x)^{\mathrm{T}} A y(x) > 0$ for all functions $y(x)$ other than $y(x) = 0$.

2.4.2. *Splines can be written as parametric models*

Splines may be written in terms of an infinite set of fixed basis functions, as in equation (2), as follows. First rescale the x axis so that the interval $(0, 2\pi)$ is much wider than the range of x values of interest. Let the basis functions be a Fourier set $\{\cos hx, \sin hx, h = 0, 1, 2, \ldots\}$. Use the regularizer

$$E_W(\mathbf{w}) = \sum_{h=0}^{\infty} \frac{1}{2} h^{\frac{p}{2}} w_{h(\cos)}^2 + \sum_{h=1}^{\infty} \frac{1}{2} h^{\frac{p}{2}} w_{h(\sin)}^2 \tag{14}$$

to define a Gaussian prior on $\mathbf{w}$,

$$P(\mathbf{w}|\alpha) = \frac{1}{Z_W(\alpha)} \exp(-\alpha E_W). \tag{15}$$

If $p = 2$ then we have the cubic splines regularizer $E_W(\mathbf{w}) = \int y^{(2)}(x)^2 dx$, as in equation (9); if $p = 1$ we have the regularizer $E_W(\mathbf{w}) = \int y^{(1)}(x)^2 dx$, etc. (To make the prior proper we must add an extra regularizer on the term $w_{0(\cos)}$.) Thus in terms of the prior $P(y(x))$ there is no fundamental difference between the 'nonparametric' splines approach and other parametric approaches.

2.4.3. *Representation is irrelevant for prediction*

From the point of view of prediction at least, there are two objects of interest. The first is the conditional distribution $P(t_{N+1}|\mathbf{t}_N, \mathbf{X}_{N+1})$ defined in equation (7). The other object of interest, should we wish to compare one model with others, is the joint probability of all the observed data given the model, $P(\mathbf{t}_N|\mathbf{X}_N)$, which appeared as the normalizing constant in equation (5). Neither of these quantities makes any reference to the representation of the unknown function $y(x)$. So at the end of the day, our choice of representation is irrelevant.

The question we now address is, in the case of popular parametric models, what form do these two quantities take? We will see that for standard models with fixed basis functions and Gaussian distributions on the unknown parameters, the joint probability of all the observed data given the model, $P(\mathbf{t}_N|\mathbf{X}_N)$, is a multivariate Gaussian distribution with mean zero and with a covariance matrix determined by the basis functions; this implies that the conditional distribution $P(t_{N+1}|\mathbf{t}_N, \mathbf{X}_{N+1})$ is also a Gaussian distribution, whose mean depends linearly on the values of the targets $\mathbf{t}_N$. Standard parametric models are simple examples of Gaussian processes.

3. From parametric models to Gaussian processes

3.1. LINEAR MODELS

Let us consider a regression problem using H fixed basis functions, for example one–dimensional radial basis functions as defined in equation (3). We will then consider what happens as we increase H.

Let us assume that a list of N input points $\{\mathbf{x}^{(n)}\}$ has been specified and define the $N \times H$ matrix $\mathbf{R}$ to be the matrix of values of the basis functions $\{\phi_h(\mathbf{x})\}_{h=1}^{H}$ at the points $\{\mathbf{x}_n\}$,

$$R_{nh} \equiv \phi_h(\mathbf{x}^{(n)}). \tag{16}$$

We define the vector $\mathbf{y}_N$ to be the vector of values of $y(\mathbf{x})$ at the N points,

$$y_n \equiv \sum_h R_{nh} w_h. \tag{17}$$

If the prior distribution of $\mathbf{w}$ is Gaussian with zero mean,

$$P(\mathbf{w}) = \text{Normal}(\mathbf{0}, \sigma_w^2 \mathbf{I}), \tag{18}$$

then clearly $\mathbf{y}$, being a linear function of $\mathbf{w}$, is also Gaussian distributed, with mean zero. The covariance matrix of $\mathbf{y}$ is

$$\begin{aligned} \mathbf{Q} &= \langle \mathbf{y}\mathbf{y}^{\mathrm{T}} \rangle = \langle \mathbf{R}\mathbf{w}\mathbf{w}^{\mathrm{T}}\mathbf{R}^{\mathrm{T}} \rangle = \mathbf{R} \langle \mathbf{w}\mathbf{w}^{\mathrm{T}} \rangle \mathbf{R}^{\mathrm{T}} && (19) \\ &= \sigma_w^2 \mathbf{R}\mathbf{R}^{\mathrm{T}}. && (20) \end{aligned}$$

So the prior distribution of $\mathbf{y}$ is:

$$P(\mathbf{y}) = \text{Normal}(\mathbf{0}, \mathbf{Q}) = \text{Normal}(\mathbf{0}, \sigma_w^2 \mathbf{R}\mathbf{R}^{\mathrm{T}}). \tag{21}$$

This result, that the vector of N function values $\mathbf{y}$ has a Gaussian distribution, is true for any selected points $\mathbf{X}_N$. This is the defining property of a Gaussian process. *The probability distribution of a function* $y(\mathbf{x})$ *is a Gaussian process if for any finite selection of points* $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N)}$*, the marginal density* $P(y(\mathbf{x}^{(1)}), y(\mathbf{x}^{(2)}), \ldots, y(\mathbf{x}^{(N)}))$ *is a Gaussian.*

Now, if the number of basis functions H is smaller than the number of data points N, then the matrix $\mathbf{Q}$ will not have full rank. In this case the probability distribution of $\mathbf{y}$ might be thought of as a flat elliptical pancake confined to an H–dimensional subspace in the N–dimensional space in which $\mathbf{y}$ lives.

What about the target values? If each target t_n is assumed to differ by additive Gaussian noise of variance σ_ν^2 from the corresponding function value y_n then $\mathbf{t}$ also has a Gaussian prior distribution,

$$P(\mathbf{t}) = \text{Normal}(\mathbf{0}, \mathbf{Q} + \sigma_\nu^2 \mathbf{I}). \tag{22}$$

We will denote the covariance matrix of $\mathbf{t}$ by $\mathbf{C}$:

$$\mathbf{C} = \mathbf{Q} + \sigma_\nu^2 \mathbf{I} = \sigma_w^2 \mathbf{R}\mathbf{R}^{\mathrm{T}} + \sigma_\nu^2 \mathbf{I}. \tag{23}$$

Whether or not $\mathbf{Q}$ has full rank, the covariance matrix $\mathbf{C}$ always has full rank since $\sigma_\nu^2 \mathbf{I}$ is full rank.

What does the covariance matrix $\mathbf{Q}$ look like? In general, the (n, n') entry of $\mathbf{Q}$ is

$$Q_{nn'} = [\sigma_w^2 \mathbf{R}\mathbf{R}^{\mathrm{T}}]_{nn'} = \sigma_w^2 \sum_h \phi_h(\mathbf{x}^{(n)}) \phi_h(\mathbf{x}^{(n')}) \tag{24}$$

and the (n, n') entry of $\mathbf{C}$ is

$$C_{nn'} = \sigma_w^2 \sum_h \phi_h(\mathbf{x}^{(n)}) \phi_h(\mathbf{x}^{(n')}) + \delta_{nn'}, \tag{25}$$

where $\delta_{nn'} = 1$ if $n = n'$ and 0 otherwise.

Let's take as an example a one–dimensional case, with radial basis functions. The expression for $Q_{nn'}$ becomes simplest if we assume we have uniformly spaced basis functions and take the limit $H \to \infty$, so that the sum over h becomes an integral; to avoid having a covariance that diverges with H, we had better make σ_w^2 scale as $S/(\Delta H)$, where ΔH is the number of basis functions per unit length of the x–axis; then

$$\begin{aligned} Q_{nn'} &= S \int_{h_{\min}}^{h_{\max}} dh\, \phi_h(x^{(n)}) \phi_h(x^{(n')}) && (26) \\ &= S \int_{h_{\min}}^{h_{\max}} dh\, \exp\left[-\frac{(x^{(n)} - h)^2}{2r^2}\right] \exp\left[-\frac{(x^{(n')} - h)^2}{2r^2}\right]. && (27) \end{aligned}$$

If we let the limits of integration be $\pm\infty$, we can solve this integral:

$$Q_{nn'} = \sqrt{\pi r^2}\, S \exp\left[-\frac{(x^{(n')} - x^{(n)})^2}{4r^2} \right]. \tag{28}$$

We are arriving at a new perspective on the interpolation problem. Instead of specifying the prior distribution on functions that the standard radial basis function model assumes in terms of basis functions and priors on parameters, it can be summarised simply by a covariance function,

$$C(x^{(n)}, x^{(n')}) \equiv \theta_1 \exp\left[-\frac{(x^{(n')} - x^{(n)})^2}{4r^2} \right], \tag{29}$$

where we have given a new name, θ_1, to the constant out front.

Generalizing from this particular case, a vista of interpolation methods opens up. Given any valid covariance function $Q(\mathbf{x}, \mathbf{x}')$ — we'll discuss in a moment what 'valid' means — we can define the covariance matrix for N function values at locations $\mathbf{X}_N$ to be the matrix $\mathbf{Q}$ given by

$$Q_{nn'} = C(\mathbf{x}^{(n)}, \mathbf{x}^{(n')}) \tag{30}$$

and the covariance matrix for N corresponding target values, assuming Gaussian noise, to be the matrix $\mathbf{C}$ given by

$$C_{nn'} = C(\mathbf{x}^{(n)}, \mathbf{x}^{(n')}) + \sigma_\nu^2 \delta_{nn'}. \tag{31}$$

3.2. MULTILAYER NEURAL NETWORKS AND GAUSSIAN PROCESSES

The recent interest among neural network researchers in Gaussian processes was initiated by the work of Neal (1996) on priors for infinite networks. Figures 1 and 2 show some random samples from the prior distribution over functions defined by a selection of standard multilayer perceptrons with large numbers of hidden units. Neal showed that the properties of a neural network with one hidden layer (as in equation (4)) converge to those of a Gaussian process as the number of hidden neurons tends to infinity if standard 'weight decay' priors are assumed. The covariance function of this Gaussian process depends on the details of the priors assumed for the weights in the network and the activation functions of the hidden units.

This observation motivated the idea of replacing supervised neural networks by Gaussian processes, a research direction explored by Williams and Rasmussen (1996) and Neal (1997). A thorough comparision of Gaussian processes with other methods such as neural networks and MARS was made by Rasmussen (1996).

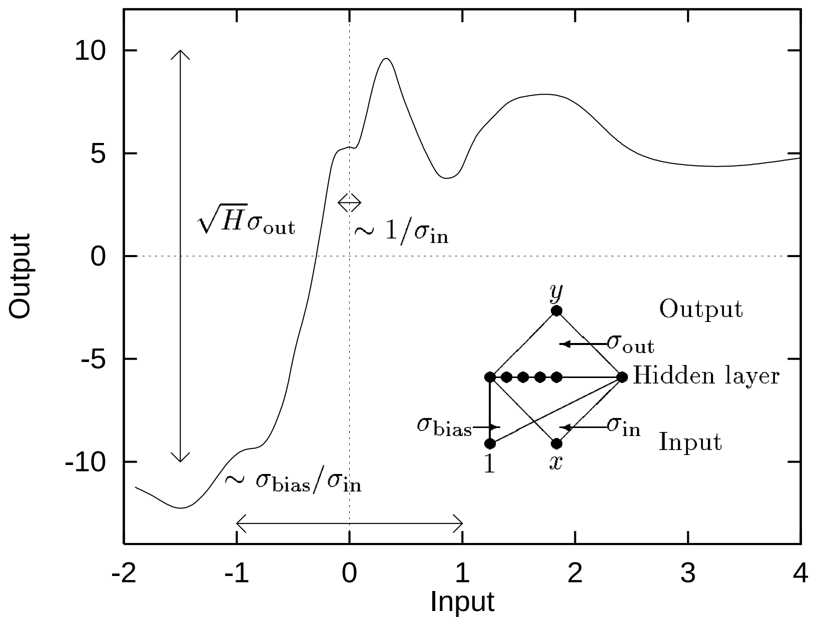


Figure 1. Properties of a function produced by a random network. The vertical scale of a typical function produced by the network with random weights is of order $\sqrt{H}\sigma_{\rm out}$; the horizontal range in which the function varies significantly is of order $\sigma_{\rm bias}/\sigma_{\rm in}$; and the shortest horizontal length scale is of order $1/\sigma_{\rm in}$. This network had $H = 400$, and Gaussian weights were generated with $\sigma_{\rm bias} = 4$, $\sigma_{\rm in} = 8$, and $\sigma_{\rm out} = 0.5$.

4. Using a given Gaussian Process model in regression

We have spent some time talking about priors. We now return to our data and the problem of prediction. How do we make predictions with a Gaussian process?

Having formed the covariance matrix $\mathbf{C}$ defined in equation (31) our task is going to be to infer t_{N+1} given the observed vector $\mathbf{t}_N$. The inference of t_{N+1} given $\mathbf{t}_N$ is simple because the joint density $P(t_{N+1}, \mathbf{t}_N)$ is a Gaussian; so the conditional distribution

$$P(t_{N+1}|\mathbf{t}_N) = \frac{P(t_{N+1}, \mathbf{t}_N)}{P(\mathbf{t}_N)} \tag{32}$$

is also a Gaussian. We now distinguish between different sizes of covariance matrix $\mathbf{C}$ with a subscript, such that $\mathbf{C}_{N+1}$ is the $(N{+}1)\times(N{+}1)$ covariance matrix for the vector $\mathbf{t}_{N+1} \equiv (t_1, \ldots, t_{N+1})^{\rm T}$. We define submatrices of

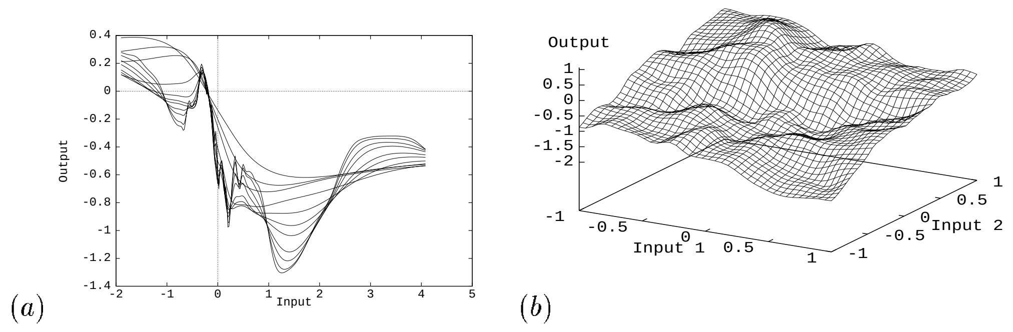


Figure 2. **Samples from the prior of** (*a*) **a one input network; and** (*b*) **a two input network.** (*a*) **Varying $\sigma^w_{\rm bias}$ and $\sigma^w_{\rm in}$ in a one input network:** In this figure, $H = 400$, $\sigma^w_{\rm out} = 0.05$. For each graph the parameter $\sigma^w_{\rm bias}$ takes a different value in the sequence: 8, 6, 4, 3, 2, 1.6, 1.2, 0.8, 0.4, 0.3, 0.2 (from most wiggly to smoothest). The parameter $\sigma^w_{\rm in}$ was also varied such that $\sigma^w_{\rm in}/\sigma^w_{\rm bias} = 5.0$. The larger values of $\sigma^w_{\rm in}$ and $\sigma^w_{\rm bias}$ produce the more complex functions with more fluctuations. (*b*) **A typical function produced by a two input network** with $\{H, \sigma^w_{\rm in}, \sigma^w_{\rm bias}, \sigma^w_{\rm out}\} = \{400, 8.0, 8.0, 0.05\}$.

$\mathbf{C}_{N+1}$ as follows:

$$\mathbf{C}_{N+1} \equiv \left[\begin{array}{cc} \left[\quad \mathbf{C}_N \quad\right] & \left[\, \mathbf{k} \,\right] \\ \left[\quad \mathbf{k}^{\rm T} \quad\right] & \left[\, \kappa \,\right] \end{array}\right] \tag{33}$$

The posterior distribution (32) is given by

$$P(t_{N+1}|\mathbf{t}_N) \propto \exp\left[-\frac{1}{2}\left[\begin{array}{cc} \mathbf{t}_N & t_{N+1} \end{array}\right] \mathbf{C}^{-1}_{N+1} \left[\begin{array}{c} \mathbf{t}_N \\ t_{N+1} \end{array}\right]\right]. \tag{34}$$

We can evaluate the mean and standard deviation of the posterior distribution of t_{N+1} by brute force inversion of $\mathbf{C}_{N+1}$. There is a more elegant expression for the predictive distribution, however, which is useful whenever predictions are to be made at a number of new points on the basis of the data set of size N. We can write $\mathbf{C}^{-1}_{N+1}$ in terms of $\mathbf{C}_N$ and $\mathbf{C}^{-1}_N$ using the partitioned inverse equations (Barnett 1979)

$$\mathbf{C}^{-1}_{N+1} = \left[\begin{array}{cc} \mathbf{M} & \mathbf{m} \\ \mathbf{m}^{\rm T} & \mu \end{array}\right] \tag{35}$$

where

$$\mu \;=\; \left(\kappa - \mathbf{k}^T \mathbf{C}^{-1}_N \mathbf{k}\right)^{-1} \tag{36}$$

$$\begin{aligned} \mathbf{m} &= -\mu\, \mathbf{C}_N^{-1}\mathbf{k} & (37) \\ \mathbf{M} &= \mathbf{C}_N^{-1} + \frac{1}{\mu}\mathbf{m}\mathbf{m}^T. & (38) \end{aligned}$$

When we substitute this matrix into equation (34) we find

$$P(t_{N+1}|\mathbf{t}_N) = \frac{1}{Z}\exp\left[-\frac{(t_{N+1}-\hat{t}_{N+1})^2}{2\sigma^2_{\hat{t}_{N+1}}}\right] \tag{39}$$

where

$$\begin{aligned} \hat{t}_{N+1} &= \mathbf{k}^T\mathbf{C}_N^{-1}\mathbf{t}_N & (40) \\ \sigma^2_{\hat{t}_{N+1}} &= \kappa - \mathbf{k}^T\mathbf{C}_N^{-1}\mathbf{k}. & (41) \end{aligned}$$

The predictive mean at the new point is given by $\hat{t}_{N+1}$ and $\sigma_{\hat{t}_{N+1}}$ defines 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. Thus Gaussian Processes allow one to effectively implement a model with a number of basis functions H much larger than the number of data points N, with the computational requirement being only of order N^3.

The predictions produced by a Gaussian process depend entirely on the covariance matrix $\mathbf{C}$. We will now discuss the sorts of covariance functions one might choose to define $\mathbf{C}$, then how we can automate the selection of the covariance function in response to data.

5. Examples of covariance functions

5.1. GENERAL POINTS

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 will denote the parameters of a covariance function by Θ. The covariance matrix of $\mathbf{t}$ has entries given by

$$C_{mn} = C(\mathbf{x}^{(m)}, \mathbf{x}^{(n)}; \Theta) + \delta_{mn}\mathcal{N}(\mathbf{x}^{(n)}; \Theta) \tag{42}$$

where C is the covariance function on which we concentrate from now on, and $\mathcal{N}$ is a noise model which might be stationary or spatially varying, for example,

$$\mathcal{N}(\mathbf{x};\Theta) = \begin{cases} \theta_3 & \text{for input–independent noise} \\ \exp\left(\sum_{j=1}^J \beta_j\phi_j(\mathbf{x})\right) & \text{for input–dependent noise.} \end{cases} \tag{43}$$

The continuity properties of C determine the continuity properties of typical samples from the Gaussian process prior. If $C(\mathbf{x}, \mathbf{x}')$ is a continuous function of its arguments then typical functions $y(x)$ are continuous too. An encyclopaedic paper on Gaussian processes giving many valid covariance functions has been written by Abrahamsen (1997).

5.2. STATIONARY COVARIANCE FUNCTIONS

A *stationary* covariance function is one which is translation invariant in that it satisfies

$$C(\mathbf{x}, \mathbf{x}'; \Theta) = D(\mathbf{x} - \mathbf{x}'; \Theta) \tag{44}$$

for some function D, *i.e.*, the covariance is a function of separation only, also known as the autocovariance function. If additionally C only depends on the *magnitude* of the distance between $\mathbf{x}$ and $\mathbf{x}'$ then the covariance function is said to be homogenous. Stationary covariance functions may also be described in terms of the Fourier transform of the function D, which is known as the power spectrum of the Gaussian process. This Fourier transform is necessarily a positive function of frequency. One way of constructing a valid stationary covariance function is simply to invent a positive function of frequency and define D to be its inverse Fourier transform. A simple example of this relationship is given by a power spectrum that is a Gaussian function of frequency. Since the Fourier transform of a Gaussian is a Gaussian, the autocovariance function corresponding to this power spectrum is a Gaussian function of separation. This argument rederives the covariance function we derived at equation (29).

One possible form for C is

$$C(\mathbf{x}, \mathbf{x}'; \Theta) = \theta_1 \exp\left[-\frac{1}{2}\sum_{i=1}^{I}\frac{(x_i - x_i')^2}{r_i^2}\right] + \theta_2 \tag{45}$$

where x_i is the i^{th} component of $\mathbf{x}$, an I dimensional vector, and $\Theta = (\theta_1, \theta_2, \{r_i\})$. There is a length scale r_i corresponding to each input which characterizes the distance in that particular direction over which y is expected to vary significantly. A very large length scale means that y is expected to be essentially a constant function of that input. Such an input could be said to be irrelevant, as in the automatic relevance determination (ARD) method for neural networks (MacKay 1994, Neal 1996). The θ_1 hyperparameter defines the vertical scale of variations of a typical function. The θ_2 hyperparameter allows the whole function to be offset away from zero by some unknown constant — to understand this term, examine equation (24) and consider the basis function $\phi(\mathbf{x}) = 1$.

Another stationary covariance function is

$$C(x, x') = \exp(-|x - x'|^\nu); 0 < \nu \leq 2. \tag{46}$$

For $\nu = 2$, this is a special case of the previous covariance function. For $\nu \in (1, 2)$, the typical functions from this prior are smooth but not analytic functions. For $\nu \leq 1$ typical functions are continuous but not smooth.

A covariance function that models a function that is periodic with known period λ_i in the i^{th} input direction is

$$C(\mathbf{x}, \mathbf{x}'; \Theta) = \theta_1 \exp\left[-\frac{1}{2}\sum_i \left(\frac{\sin\left(\frac{\pi}{\lambda_i}(x_i - x_i')\right)}{r_i}\right)^2\right]. \tag{47}$$

Figure 3 shows some random samples drawn from Gaussian processes with a variety of different covariance functions.

5.3. NONSTATIONARY COVARIANCE FUNCTIONS

The simplest nonstationary covariance function is the one corresponding to a linear trend. Consider the plane $y(\mathbf{x}) = \sum_i w_i x_i + c$. If the $\{w_i\}$ and c have Gaussian distributions with zero mean and variances σ_w and σ_c respectively then the plane has a covariance function

$$C_{\text{lin}}(\mathbf{x}, \mathbf{x}'; \{\sigma_w, \sigma_c\}) = \sum_{i=1}^{I} \sigma_w^2 x_i x_i' + \sigma_c^2. \tag{48}$$

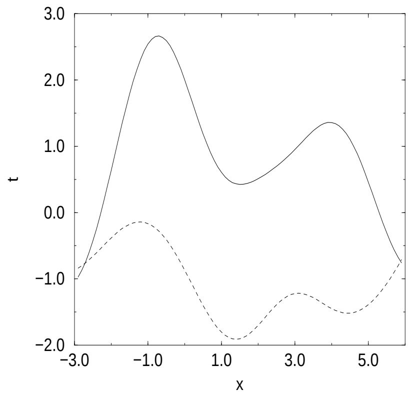
An example of random sample functions incorporating the linear term can be seen in figure 3(d).

5.3.1. *Spatially varying length scales:*

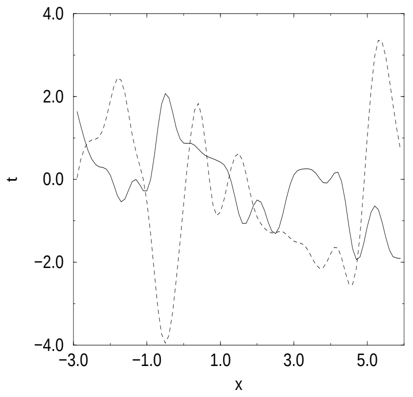
The standard covariance function (45) assumes that the length scales $\{r_i\}$ are fixed. We might be interested in a model in which the length scales are somehow made to be functions of $\mathbf{x}$. We cannot simply substitute a parameterized form for $r_i(\mathbf{x})$ into equation (45) as this will not in general give us a positive definite covariance function. Gibbs (1997) shows that the covariance function

$$C^{\text{ns}}(\mathbf{x}, \mathbf{x}') = \theta_1 \prod_i \left\{\frac{2r_i(\mathbf{x})r_i(\mathbf{x}')}{r_i^2(\mathbf{x}) + r_i^2(\mathbf{x}')}\right\}^{1/2} \exp\left(-\sum_i \frac{(x_i - x_i')^2}{r_i^2(\mathbf{x}) + r_i^2(\mathbf{x}')}\right), \tag{49}$$

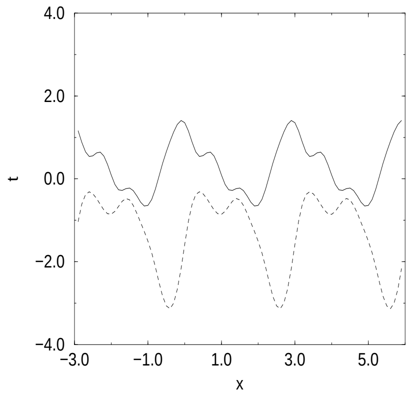
is positive definite and has spatially varying length scales, where $r_i(\mathbf{x})$ is an arbitrary positive function of $\mathbf{x}$. It also has the property that the marginal variance is independent of $\mathbf{x}$ and equal to θ_1.



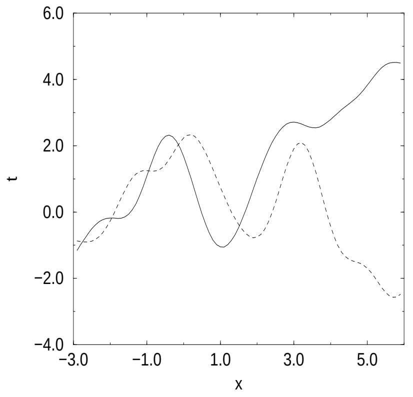
(a) $2.0 \exp\left(-\frac{(x-x')^2}{2(1.5)^2}\right)$



(b) $2.0 \exp\left(-\frac{(x-x')^2}{2(0.35)^2}\right)$



(c) $2.0 \exp\left(-\frac{\sin^2(\pi(x-x')/3.0)}{2(0.5)^2}\right)$



(d) $2.0 \exp\left(-\frac{(x-x')^2}{2(1.5)^2}\right) + 1.0xx'$

Figure 3. **Samples drawn from Gaussian process priors:** This figure shows two functions drawn from each of four Gaussian process priors with different covariance functions. The corresponding covariance function is given below each plot. The decrease in length scale from (a) to (b) produces more rapidly fluctuating functions. The periodic properties of the covariance function in (c) can clearly be seen. The covariance function in (d) contains the non-stationary term xx' corresponding to the covariance of a straight line, so that typical functions include linear trends.

5.4. WAYS OF MAKING FURTHER COVARIANCE FUNCTIONS

Sum of covariance functions: If $C_1(x, x')$ and $C_2(x, x')$ are both covariance functions over the same space x then $D(x, x') = C_1(x, x') + C_2(x, x')$ is also a covariance function.

Simple product: If $C_1(x, x')$ and $C_2(x, x')$ are covariance functions on the same space then so is $D(x, x') \equiv C_1(x, x')C_2(x, x')$.

Covariance functions for product spaces: If $C_1(x, x')$ and $C_2(y, y')$ are covariance functions over different spaces, and we define a product space $\mathbf{z} = (x, y)$ then $D(\mathbf{z}, \mathbf{z}') = C_1(x, x') + C_2(y, y')$ and $E(\mathbf{z}, \mathbf{z}') = C_1(x, x')C_2(y, y')$ are also covariance functions.

Hence we can generate new functions using simpler covariance functions as the building blocks. For example, we can use a sum of several of the Gaussian terms which appear in equation (45), each with independent hyperparameters, in the covariance function. This gives us the possibility of modelling large scale fluctuations in one direction with one Gaussian and smaller scale fluctuations with another, i.e., producing an additive model.

5.4.1. *Blurring*

We can imagine making a new Gaussian process by convolving an old one C with an arbitrary kernel h:

$$D(x, x') = \int dy\, dy'\, h(x - y)C(y, y')h(y' - x'). \tag{50}$$

A simple Gaussian process from which to start is the white noise process having $C(x, x') = \delta(x, x')$. If we select as our kernel a one–dimensional top hat function of width 1, we obtain:

$$C_1(x, x') = \begin{cases} 1 - |x - x'| & |x - x'| < 1 \\ 0 & |x - x'| > 1 \end{cases} . \tag{51}$$

If we move to three dimensions and use a spherical top hat function we obtain:

$$C_3(x, x') = \begin{cases} 1 - \frac{3}{2}|x - x'| + \frac{1}{2}|x - x'|^3 & |x - x'| < 1 \\ 0 & |x - x'| > 1 \end{cases} , \tag{52}$$

which is a valid covariance function in one, two or three dimensions.

Other kernels can be used and give rise to other covariance functions. The kernel need not be a strictly positive function.

5.4.2. *Vertical rescaling*

A trivial way of making a nonstationary covariance function from some given covariance function $C(\mathbf{x}, \mathbf{x}')$ is to introduce any function $a(\mathbf{x})$ and define

$$D(\mathbf{x}, \mathbf{x}') \equiv a(\mathbf{x})C(\mathbf{x}, \mathbf{x}')a(\mathbf{x}'). \tag{53}$$

This function is a valid covariance function, and if the original function was stationary with marginal variance $C(\mathbf{x}, \mathbf{x}) \equiv C_0$ for all $\mathbf{x}$, the new function has marginal variance $D(\mathbf{x}, \mathbf{x}) = C_0 a(\mathbf{x})^2$.

5.4.3. *Warping or embedding*

Given any covariance function $C(\mathbf{u}, \mathbf{u}')$, we can introduce an arbitrary non-linear mapping $\mathbf{x} \to \mathbf{u}(\mathbf{x})$ and define a new covariance function

$$D(\mathbf{x}, \mathbf{x}') \equiv C(\mathbf{u}(\mathbf{x}), \mathbf{u}(\mathbf{x}')). \tag{54}$$

If the original covariance function C is stationary, this mapping will in general produce a nonstationary covariance function with uniform variance.

Note that $\mathbf{x}$ and $\mathbf{u}$ need not have the same dimensionality as each other, and the function $\mathbf{x} \to \mathbf{u}(\mathbf{x})$ need not be invertible — though if two distinct values of $\mathbf{x}$ satisfy $\mathbf{u}(\mathbf{x}^{(A)}) = \mathbf{u}(\mathbf{x}^{(B)})$ then all sample functions $y(\mathbf{x})$ from the Gaussian process will necessarily have $y(\mathbf{x}^{(A)}) = y(\mathbf{x}^{(B)})$, which might or might not be desired.

An example of an embedding of a one–dimensional x in a higher dimensional space $\mathbf{u} = (u_1, u_2)$ is the mapping $\mathbf{u} = (\cos(x), \sin(x))$. Then $D(x, x') = C(\mathbf{u}(x), \mathbf{u}(x'))$ is automatically a covariance function for a periodic function. Using the standard C of equation (45), this is one way of deriving the covariance function given in equation (47).

5.4.4. *A few derived covariance functions*

Using the tricks of vertical rescaling and multiplication, we may obtain some more covariance functions from Gibbs's function (49). Assume we define some positive function $r(x)$.

$$\begin{aligned} C_a(x, x') &= \frac{1}{r(x)^2 + r(x')^2} \exp\left[-\frac{(x-x')^2}{r(x)^2 + r(x')^2}\right] && (55) \\ C_b(x, x') &= \frac{r(x) r(x')}{r(x)^2 + r(x')^2} \exp\left[-\frac{(x-x')^2}{r(x)^2 + r(x')^2}\right] && (56) \\ C_c(x, x') &= \frac{1}{[r(x)^{-2} + r(x')^{-2}]^{1/2}} \exp\left[-\frac{(x-x')^2}{r(x)^2 + r(x')^2}\right] && (57) \end{aligned}$$

Whether these covariance functions are useful remains to be seen. The function C_a looks like a reasonable model for neuronal spikes, as it associates short lengthscales with large vertical variations.

5.4.5. *O'Hagan's model*

Another covariance function obtained by multiplication and addition (O'Hagan 1978) is

$$D(x, x') = x C(x, x') x' + C(x, x'), \tag{58}$$

where C is any convenient covariance function, for example the standard one (45). If the length scale of C is long then typical functions have quasi–linear trends, except that the slope and intercept of the linear trend may wander with x.

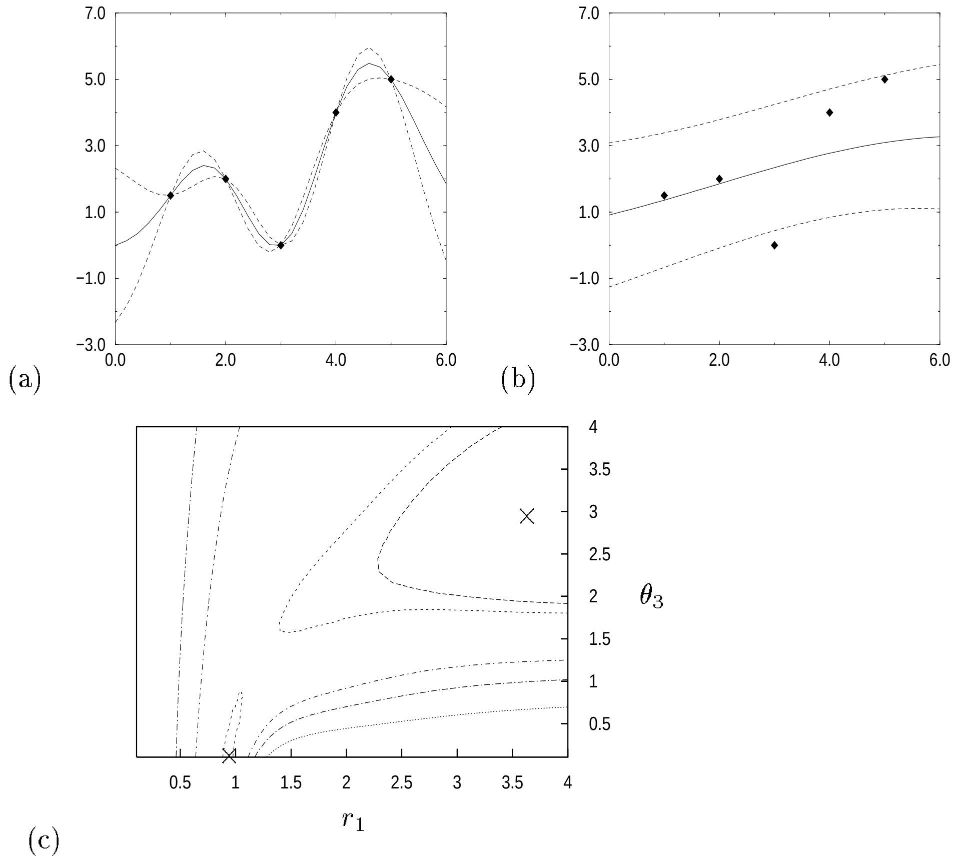


Figure 4. **Multimodal Likelihood Functions:** This figure highlights the problems that can be encountered when using a Gaussian process with no priors on the hyperparameters. (a) shows the most probable interpolant and its 1σ error bars obtained when a Gaussian process was used to model the data shown by the black points. The covariance function used is given in equation (45). The hyperparameters of the Gaussian process were initialized with the noise level and length scale both set to low values ($\theta_3 = 0.01, r_1 = 1.0$) and the evidence was maximized with respect to the hyperparameters using conjugate gradients. (b) shows the most probable interpolant and its 1σ error bars obtained when an identical Gaussian process with identical training data and learning procedure was given different initial conditions (high noise level and large length scale ($\theta_3 = 2.0, r_1 = 4.0$)). Panel (c) shows a contour plot of the likelihood of Θ. The two distinct modes (one at $r_l = 0.95$, $\theta_3 = 0.0$ and one at $r_l = 3.5$, $\theta_3 = 3.0$) which give rise to the two different solutions are shown by crosses.

6. Adaptation of Gaussian Process models

Let us assume that a form of covariance function has been chosen, but that it depends on undetermined hyperparameters Θ. We would like to 'learn' these hyperparameters from the data. This learning process is equivalent to the inference of the hyperparameters of a neural network, for example, weight decay hyperparameters. Ideally we would like to define a prior distribution on the hyperparameters and integrate over them in order to make our predictions, *i.e.*, we would like to find

$$P(t_{N+1}|\mathbf{x}_{N+1},\mathbf{t}_N,\mathbf{X}_N) = \int P(t_{N+1}|\mathbf{x}_{N+1},\Theta,\mathbf{t}_N,\mathbf{X}_N)P(\Theta|\mathbf{t}_N,\mathbf{X}_N)d\Theta \tag{59}$$

But this integral is usually intractable. There are two approaches we can take.

1. We can approximate the integral by using the most probable values of hyperparameters.

$$P(t_{N+1}|\mathbf{x}_{N+1},\mathbf{t}_N,\mathbf{X}_N) \simeq P(t_{N+1}|\mathbf{x}_{N+1},\mathbf{t}_N,\mathbf{X}_N,\Theta_{\text{MP}}) \tag{60}$$

2. Or we can perform the integration over Θ numerically using Monte Carlo methods (Williams and Rasmussen 1996, Neal 1997).

Either of these approaches is implemented most efficiently if the gradient of the posterior probability of Θ can be evaluated.

6.1. GRADIENT

The posterior probability of Θ is

$$P(\Theta\,|\,\mathbf{t}_N,\mathbf{X}_N) \propto P(\mathbf{t}_N\,|\,\mathbf{X}_N,\Theta)P(\Theta) \tag{61}$$

The log of the first term (the evidence for the hyperparameters) $\mathcal{L}$ is

$$\mathcal{L} = -\frac{1}{2}\log\det\mathbf{C}_N - \frac{1}{2}\mathbf{t}_N^T\mathbf{C}_N^{-1}\mathbf{t}_N - \frac{N}{2}\log 2\pi \tag{62}$$

and its derivative with respect to a hyperparameter θ is

$$\frac{\partial\mathcal{L}}{\partial\theta} = -\frac{1}{2}\text{Trace}\left(\mathbf{C}_N^{-1}\frac{\partial\mathbf{C}_N}{\partial\theta}\right) + \frac{1}{2}\mathbf{t}_N^T\mathbf{C}_N^{-1}\frac{\partial\mathbf{C}_N}{\partial\theta}\mathbf{C}_N^{-1}\mathbf{t}_N. \tag{63}$$

6.2. COMMENTS

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, as illustrated in figure 4, the data–dependent term $\mathcal{L}$ is often multimodal. 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 of order N^3 and so calculating gradients becomes time consuming for large training data sets.

7. Implementation of the Model

There are two possible approaches to the implementation of prediction (equations (40) and (41)) and gradient computation (equation (63)).

7.1. DIRECT METHODS

The most obvious implementation of these equations is to evaluate the inverse of the covariance matrix exactly. This can be done using a variety of methods such as Cholesky decomposition, LU decomposition or Gauss–Jordan Elimination. 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 construct the vector $\mathbf{k}$, 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 $\hat{t}_{N+1} = (\mathbf{k}^T\mathbf{v})$. If we wish to find the most probable value of the interpolant at another new point $\mathbf{x}_{N+2}$ given the same data $\mathbf{t}_N, \mathbf{X}_N$ (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. 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. Thus finding the predictive mean has a cost similar to the cost of using a feedforward network with N fixed basis functions. In this analogy, the vector $\mathbf{v}$ is the weight vector and the N basis functions are $\phi_n(\mathbf{x}) \equiv C(\mathbf{x}^{(n)}, \mathbf{x})$, for $n = 1, \ldots, N$.

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. 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 its diagonal elements.

The explicit method has two principal 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 $\mathbf{k}^T\mathbf{v}$ may turn out to be the sum of very large positive and negative numbers although its magnitude may be small. This may lead to 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, for example, when the model assumes a small noise level. To reduce such numerical errors, we can use the LU decomposition of $\mathbf{C}_N$ or eigenvector/eigenvalue decompositions. Another way to deal with ill–conditioning is to split $\mathbf{C}$ into several pieces and make use of the identity

$$[\mathbf{C}+\mathbf{M}\mathbf{N}]^{-1} = \mathbf{C}^{-1} - \mathbf{C}^{-1}\mathbf{M}(\mathbf{I}+\mathbf{N}\mathbf{C}^{-1}\mathbf{M})^{-1}\mathbf{N}\mathbf{C}^{-1}. \tag{64}$$

This is a useful way to handle ill conditioning produced by a large value of the constant term θ_2 in the covariance function of equation (45).

These exact methods scale as $\mathcal{O}(N^3)$ and can be time consuming for large data sets.

7.2. APPROXIMATE METHODS

An alternative method for the implementation of Gaussian processes based on the ideas of Skilling (1993) makes approximations to $\mathbf{C}^{-1}\mathbf{t}$ and Trace$\mathbf{C}^{-1}$ using iterative methods which scale as $\mathcal{O}(N^2)$. These methods are useful when the number of data points exceeds a few hundred (Gibbs and MacKay 1996).

8. Regression Examples

We present two simple examples. The first is a one dimensional regression problem. This consists of a set of 37 noisy training data points shown in figure 5(a). We used the basic covariance function given in equation (45).

Ten runs were performed with different initial conditions to guard against the possibility of multiple maxima in $P(\Theta|\mathbf{t}_N, \mathbf{X}_N)$. Broad priors were placed on all the hyperparameters (gamma priors on the length scales; inverse gamma priors 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 5(b).

For the second example 400 noisy data points were generated from a 2D function (see figure 6(a)). Again broad priors were placed on all the

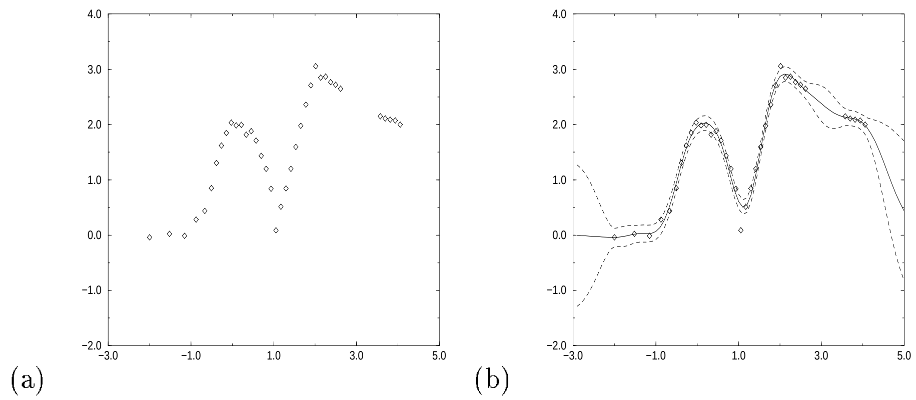


Figure 5. **1d Example :** (a) shows the 37 noisy data points used as the training data. Note the lack of data in the region $2.5 \leq x \leq 3.5$. In (b) we see the interpolant and its 1σ error bars. The error bars represent how uncertain we are about the interpolant at each point assuming that the model is correct. Note how the error bars increase where the data point density decreases. 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.

hyperparameters and ten runs were performed, each with initial hyperparameters sampled from their priors. In this case the approximate techniques of Section 7.2 were used to evaluate inverses and traces. The results from each run were similar and the interpolant obtained from the first run is shown in figure 6(b).

The time required to perform the optimization in the second example was as follows. Each training run took about 16 minutes using the approximate inversion techniques. Training runs performed using direct methods took approximately 39 minutes. This shows the advantage of the approximate methods when dealing with large amounts of training data. Figure 7 shows the training times for data sets of varying size generated using the same function.

9. Advanced topics in regression

9.1. ON-LINE DATA ADDITION

Consider the case in which we have a data set $\{\mathbf{x}_n, t_n\}(n = 1 \cdots N)$ and have found the most probable hyperparameters of the Gaussian process Θ_{MP}. Then we obtain a new data point $\{\mathbf{x}_{N+1}, t_{N+1}\}$ 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

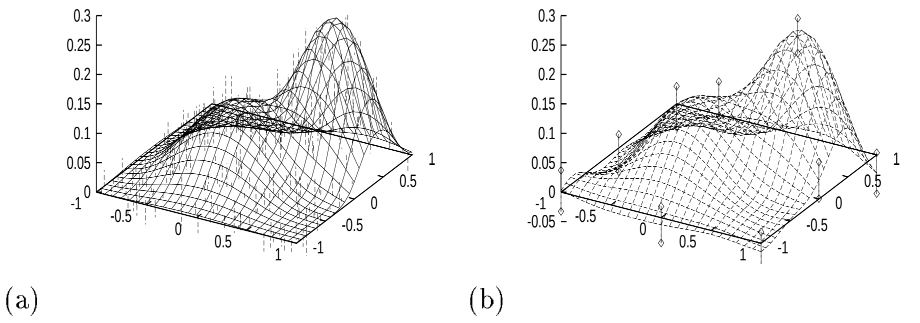


Figure 6. **2d Example :** (a) shows the function from which 400 noisy data points were generated and the noisy data points in relation to the function. The offset of each datum due to noise is shown as a dashed line. In (b) we see the most probable interpolant generated using an approximate implementation of a Gaussian process. 1σ error bars are only given at selected points in order to preserve clarity.

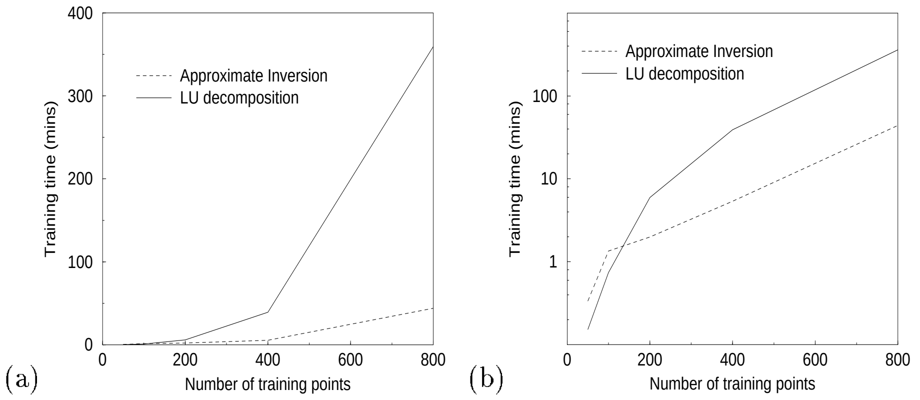


Figure 7. **Scaling of training time with amount of data :** (a) shows the training times for noisy data sets of different sizes generated from the function shown in figure 6. Times are shown for both the direct LU decomposition method and the approximate method. (b) shows the same graph with a logarithmic vertical scale. For small data sets the LU method is significantly quicker than the approximate approach. At over 100 data points we can see that the $\mathcal{O}(N^2)$ scaling of the approximate method begins to yield benefits. For 400 and 800 data points the approximate method is significantly faster than the LU decomposition method. It should, however, be noted that the problem is only two dimensional and the benefits of the approximate method for higher dimensional problems will only be obvious for larger data sets.

described by the new larger data set is approximately the same as that described by the old. We can calculate $\mathbf{C}_{N+1}^{-1}$ from $\mathbf{C}_N$ and $\mathbf{C}_N^{-1}$ using the partitioned inverse equations (35–38). 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.

9.2. MULTIPLE OUTPUTS

The subject of multiple outputs (or co-kriging (Cressie 1993)) is problematic. It is possible to define Gaussian processes with multiple outputs but it is not clear in general how the covariance function should be defined. Many problems are symmetric *a priori*, in that the sign of an output variable could be flipped, and the user would be none the wiser. If this symmetry of the user's prior distribution holds, then the covariance function can only be of the form $C_{nm}^{ab} = \delta_{ab} C_{nm}$, where a and b run over the output variables. This means that each output is modelled independently — a method known as multi-kriging (Williams and Rasmussen 1996). At this point we may feel that Gaussian processes are missing out on something, since we might intuitively expect two output variables to have some features in common.

One way to put in the 'missing something' might be to introduce some common *hyperparameters* into the covariance functions for outputs a and b. In particular, one might introduce warping functions $\mathbf{u}(\mathbf{x})$, as in section 5.4.3, that are common to both covariance functions, so that we have covariance functions $C^{aa}(\mathbf{x}, \mathbf{x}') = B^{(a)}(\mathbf{u}(\mathbf{x}), \mathbf{u}(\mathbf{x}'))$ and $C^{bb}(\mathbf{x}, \mathbf{x}') = B^{(b)}(\mathbf{u}(\mathbf{x}), \mathbf{u}(\mathbf{x}'))$, with $\mathbf{u}(\mathbf{x})$ being an *a priori* undetermined function. If $\mathbf{u}$ is subsequently inferred to be some nonlinear function, then the various output functions $y^{(a)}(\mathbf{x}), y^{(b)}(\mathbf{x}) \ldots$ will be functions which, while they are still uncorrelated in terms of their prior second order statistics, show their most complex variations in the same regions of $\mathbf{x}$ space.

10. Classification

The work of Rasmussen (1996) has shown that for non–linear regression problems with between one and one thousand data points, Gaussian processes should certainly be considered as replacement for supervised neural networks. What about classification problems?

Gaussian processes can be integrated into classification modelling once we identify a variable which can sensibly be given a Gaussian process prior.

In a a binary classification problem, we can define a quantity $a_n \equiv$

$a(\mathbf{x}^{(n)})$ such that the probability that the class is 1 rather than 0 is

$$P(t_n = 1|a_n) = \frac{1}{1 + e^{-a_n}}. \tag{65}$$

Large positive values of a correspond to probabilities close to one; large negative values of a define probabilities that are close to zero. In a classification problem, we typically intend that the probability $P(t_n = 1)$ should be a smoothly varying function of $\mathbf{x}$. We can embody this prior belief by defining $a(\mathbf{x})$ to have a Gaussian process prior. Neal (1997) gives other ways of connecting Gaussian processes to classification models.

10.1. IMPLEMENTATION

It is not so easy to perform inferences and adapt the Gaussian process model to data in a classification model as in regression problems because the likelihood function is not a Gaussian function of a_n. So the posterior distribution of $\mathbf{a}$ given some observations $\mathbf{t}$ is not Gaussian and the normalization constant $P(\mathbf{t}_N|\mathbf{X}_N)$ cannot be written down analytically. Barber and Williams (1997) have implemented classifiers based on Gaussian process priors using Laplace approximations. Neal (1997) has implemented a Monte Carlo approach to implementing a Gaussian process classifier. Gibbs and MacKay (1997) have implemented another cheap and cheerful approach based on the methods of Jaakkola and Jordan (1996). In this *variational Gaussian process classifier* (VGC), we obtain tractable upper and lower bounds for the unnormalized posterior density over $\mathbf{a}$, $P(\mathbf{t}_N|\mathbf{a})P(\mathbf{a})$. These bounds are parameterized by variational parameters which are adjusted in order to obtain the tightest possible fit. Using normalized versions of the optimized bounds we then compute approximations to the predictive distributions.

10.2. RESULTS ON TEXTBOOK PROBLEMS

We tried our VGC method on two well known classification problems, the *Leptograpsus* crabs and Pima Indian diabetes datasets[1]. The results for both tasks, together with comparisons with several other methods are given in table 1.

In the *Leptograpsus* crabs problem we attempted to classify the sex of crabs based upon six characteristics. 200 labelled examples are split into a training set of 80 and a test set of 120. The performance of the VGC is not significantly different from the best of the other methods. The Pima Indian diabetes problem involved the prediction of the occurence of diabetes in women of Pima Indian heritage based on seven characteristics. 532 examples

[1]Available from `http://markov.stats.ox.ac.uk/pub/PRNN`.

	Crab		Pima	
Method	Error	% Error	Error	% Error
Neural Network (1)	3 ± 1.7	2.5 ± 1.4	-	-
Neural Network (2)	5 ± 2.1	4.2 ± 1.8	-	-
Neural Network (3)	-	-	75	22.6
Linear Discriminant	8 ± 2.7	6.7 ± 2.3	67 ± 7.3	20.2 ± 2.2
MARS (degree = 1)	8 ± 2.7	6.7 ± 2.3	75 ± 7.6	22.6 ± 2.3
2 Gaussian Mixture	-	-	64 ± 7.2	19.3 ± 2.2
HMC Gaussian process	3 ± 1.7	2.5 ± 1.4	68 ± 7.4	20.5 ± 2.2
VGC	4 ± 2	3.3 ± 1.6	70 ± 7.4	21.1 ± 2.2

TABLE 1. **Pima and Crabs Results :** The table shows the performance of a range of different classification models on the Pima and Crabs problems (Ripley 1994, Ripley 1996). The number of classification errors and the percentage of errors both refer to the test set. The error bars given are calculated using binomial statistics. The results quoted for the VGC are those obtained using the approximations from the lower bound. The HMC Gaussian process is the classifier described by Barber and Williams (1997).

were available and these were split into 200 training examples and 332 test examples. 33% of the population were reported to have diabetes so an error rate of 33% can be achieved by declaring all examples to be non-diabetic. The VGC achieved an error rate of 21% — again comparable with the best of the other methods.

10.3. WELD CRACKING EXAMPLE

Hot cracking can occur in welds as they cool. The occurence of such cracks depends on the chemical composition of the weld metal, the cooling rate and the weld geometry. We wish to predict whether a given weld will crack by examining the dependence of cracking on 13 input variables describing a weld. This problem has previously been tackled using Bayesian neural networks (Ichikawa, Bhadeshia and MacKay 1996).

An initial test was performed using a training set of 77 examples and a test set of 77 examples. The test error rates and test log likelihoods for the VGC and the Bayesian neural network approach (Ichikawa et al. 1996) can be seen in table 2 where the test log likelihood is defined as

$$\text{test log likelihood} = \sum_{n=1}^{N_{\text{test}}} t_n \log(y_n) + (1 - t_n) \log(1 - y), \tag{66}$$

Method	Test Error	Log Likelihood
Bayesian Neural Network	8	-23.6
Variational GP Classifier	10 , 10	-25.73 , -31.57

TABLE 2. **Weld Cracking Classification Problem:** This table shows the test error and log likelihood scores of the VGC and the Bayesian neural network of Ichikawa et al. (1996). The two results given for the VGC correspond to the approximations using the lower and upper bound respectively.

where t_n is the true test set classification (either 0 or 1) and y_n is the prediction $P(t_n = 1|\mathbf{t}_N, \mathbf{X}_N)$. The performance of the VGC is slightly inferior to that of the Bayesian neural network. However the neural network result was obtained using a committee of four networks. A large amount of experimentation with different architectures and parameter settings was performed and the four networks found *with the best test error* were used in the committee (this is generally regarded as cheating, in such comparisons!). The VGC results required no such experimentation. 20 runs of the VGC with differing initial conditions were performed to check whether there were multiple minima. The results quoted in table 2 are from the first run; all the runs produced almost identical results.

We then trained the VGC using all 154 examples and studied the carbon dependence of the probability of cracking as in Ichikawa et al. (1996). A plot of the carbon dependence can be seen in figure 8 along with the corresponding results of Ichikawa *et al.*

10.4. CONCLUSION

Gaussian processes can be used to produce effective binary classifiers. The results using the variational Gaussian process classifier are comparable to the best of current classification models. Multi–class classification problems can also be solved with Monte Carlo methods (Neal 1997) and variational methods (Gibbs 1997).

11. Discussion

Gaussian processes are moderately simple to implement and use. Because very few parameters of the model need to be determined by hand (generally only the priors on the hyperparameters), Gaussian processes are useful tools for automated tasks where fine tuning for each problem is not possible.

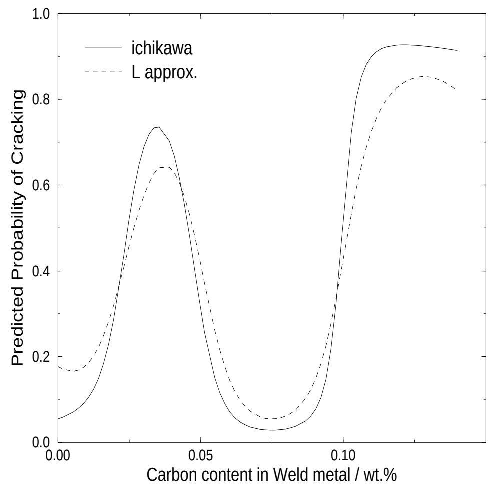


Figure 8. **Carbon Dependence of Weld Cracking probability:** These graphs show the predictions as a function of one the 13 input variables given by a committee of neural networks (Ichikawa et al. 1996) and those found using the lower bound approximation of a VGC. Both have the same large scale features.

However we do not appear to sacrifice any performance for this simplicity.

It is easy to construct Gaussian processes that have particular desired properties; for example we can make a transparently simple automatic relevance determination model.

One obvious problem with any method based upon Gaussian processes is the computational cost associated with inverting an $N \times N$ matrix. The cost of direct methods of inversion may become prohibitive when the number of data points N is greater than $\simeq 1000$. In Gibbs and MacKay (1996) efficient methods for matrix inversion (Skilling 1993) are developed that allow large data sets to be tackled. But further research is going to be needed before Gaussian processes can be applied to more than about 10,000 data points. I speculate that there may be a useful connection to be made between Gaussian processes and ‘support vector’ learning machines (Scholkopf, Burges and Vapnik 1995, Vapnik 1995), which are in some ways quite similar to Gaussian processes.

A problem with the variational approach for classification is the profileration of variational parameters when dealing with large amounts of the data. Reducing the number of these variational parameters is an important direction for further research.

11.1. HAVE WE THROWN THE BABY OUT WITH THE BATH WATER?

According to the hype of 1987, neural networks were meant to be intelligent models which discovered features and patterns in data. Gaussian processes in contrast are simply smoothing devices. How can Gaussian processes possibly replace neural networks? What is going on?

I think what the work of Williams and Rasmussen (1996) shows is that many real–world data modelling problems are perfectly well solved by sensible smoothing methods. The most interesting problems, the task of feature discovery for example, are not ones which Gaussian processes will solve. But maybe multilayer perceptrons can't solve them either.

On the other hand, it may be that the limit of an infinite number of hidden units, to which Gaussian processes correspond, was a bad limit to take; maybe we should backtrack, or modify the prior on neural network parameters, so as to create new models more interesting than Gaussian processes. Evidence that this infinite limit has lost something compared with finite neural networks comes from the observation that in a finite neural network with more than one output, there are non–trivial correlations between the outputs (since they share inputs from common hidden units); but in the limit of an infinite number of hidden units, these correlations vanish. Radford Neal has suggested the use of non–Gaussian priors in networks with multiple hidden layers. Another option suggested in section 9.2 is to introduce high–order correlations between output variables by having their Gaussian process priors share adaptable hyperparameters.

But perhaps a fresh start is needed, approaching the problem of machine learning from a paradigm different from the supervised feedforward mapping.

LITERATURE

The study of Gaussian processes for regression is far from new. Time series analysis was being performed by the astronomer T.N. Thiele using Gaussian processes in 1880 (Lauritzen 1981). In the 1940s, Wiener–Kolmogorov prediction theory was introduced for prediction of trajectories of military targets (Wiener 1948). 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 largely ignored any probabilistic interpretation of the model and any interpretation of the individual parameters of the covariance function. Kalman filters are widely used to implement inferences for stationary one–dimensional Gaussian processes (Bar-Shalom and Fortmann 1988). O'Hagan (1978) introduced an approach 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.

References

Abrahamsen, P.: 1997, A review of Gaussian random fields and correlation functions, *Technical Report 917*, Norwegian Computing Center, Box 114, Blindern, N-0314 Oslo, Norway. 2nd edition.

Bar-Shalom, Y. and Fortmann, T.: 1988, *Tracking and Data Association*, Academic Press.

Barber, D. and Williams, C. K. I.: 1997, Gaussian processes for Bayesian classification via hybrid Monte Carlo, *in* M. C. Mozer, M. I. Jordan and T. Petsche (eds), *Neural Information Processing Systems 9*, MIT Press, p. ?

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

Cressie, N.: 1993, *Statistics for Spatial Data*, Wiley.

Gibbs, M. N.: 1997, *Bayesian Gaussian Processes for Regression and Classification*, PhD thesis, Cambridge University.

Gibbs, M. N. and MacKay, D. J. C.: 1996, Efficient implementation of Gaussian processes for interpolation, in preparation.

Gibbs, M. N. and MacKay, D. J. C.: 1997, Variational Gaussian process classifiers, in preparation.

Ichikawa, K., Bhadeshia, H. K. D. H. and MacKay, D. J. C.: 1996, Model for hot cracking in low-alloy steel weld metals, *Science and Technology of Welding and Joining* **1**, 43–50.

Jaakkola, T. S. and Jordan, M. I.: 1996, Computing upper and lower bounds on likelihoods in intractable networks, *Proceedings of the Twelfth Conference on Uncertainty in AI*, Morgan Kaufman.

Kimeldorf, G. S. and Wahba, G.: 1970, A correspondence between Bayesian estimation of stochastic processes and smoothing by splines, *Annals of Mathematical Statistics* **41**(2), 495–502.

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

Lauritzen, S. L.: 1981, Time series analysis in 1880, a discussion of contributions made by T.N. Thiele, *ISI Review* **49**, 319–333.

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

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

MacKay, D. J. C.: 1994, Bayesian methods for backpropagation networks, *in* E. Domany, J. L. van Hemmen and K. Schulten (eds), *Models of Neural Networks III*, Springer-Verlag, New York, chapter 6, pp. 211–254.

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

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

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*, PhD thesis, University of Toronto.

Ripley, B. D.: 1991, *Statistical Inference for Spatial Processes*, Cambridge.

Ripley, B. D.: 1994, Flexible non–linear approaches to classification, *in* V. Cherkassky, J. H. Friedman and H. Wechsler (eds), *From Statistics to Neural Networks. Theory and Pattern Recognition Applications*, number subseries F in *ASI Proceedings*, Springer-Verlag.

Ripley, B. D.: 1996, *Pattern Recognition and Neural Networks*, Cambridge.

Rumelhart, D. E., Hinton, G. E. and Williams, R. J.: 1986, Learning representations by back–propagating errors, *Nature* **323**, 533–536.

Scholkopf, B., Burges, C. and Vapnik, V.: 1995, Extracting support data for a given task, *in* U. M. Fayyad and R. Uthurusamy (eds), *Proceedings First International Conference on Knowledge Discovery and Data Mining*, AAAI Press, Menlo Park, CA.

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

Vapnik, V.: 1995, *The Nature of Statistical Learning Theory*, Springer Verlag, New York.

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

Wiener, N.: 1948, *Cybernetics*, Wiley.

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