
About Chapter 45

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 nonlinear 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 nonlinear 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 (1997b), Barber and Williams (1997) and Gibbs and MacKay (2000), and will assess whether, for supervised regression and classification tasks, the feedforward network has been superceded.

- ▷ Exercise 45.1.^[3] I regret that this chapter is rather dry. There's no simple explanatory examples in it, and few pictures. This exercise asks you to create interesting pictures to explain to yourself this chapter's ideas.

Source code for computer demonstrations written in the free language `octave` is available at:

<http://www.inference.phy.cam.ac.uk/mackay/itprnn/software.html>.

Radford Neal's software for Gaussian processes is available at:

<http://www.cs.toronto.edu/~radford/>.

45

Gaussian Processes

After the publication of Rumelhart, Hinton and Williams's (1986) paper on supervised learning in neural networks there was a surge of interest in the empirical modelling of relationships in high-dimensional data using nonlinear parametric models such as multilayer 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

$$P(y(\mathbf{x}) | \mathbf{t}_N, \mathbf{X}_N) = \frac{P(\mathbf{t}_N | y(\mathbf{x}), \mathbf{X}_N)P(y(\mathbf{x}))}{P(\mathbf{t}_N | \mathbf{X}_N)}. \quad (45.1)$$

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 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 fitting. The prior typically specifies that the function $y(\mathbf{x})$ is expected to be continuous and smooth, and has 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, for the prediction of future values of t , all that matters is the assumed prior $P(y(\mathbf{x}))$ and the assumed noise model $P(\mathbf{t}_N | y(\mathbf{x}), \mathbf{X}_N)$ – the parameterization of the function $y(\mathbf{x}; \mathbf{w})$ is irrelevant.

The idea of Gaussian process modelling is to place a prior $P(y(\mathbf{x}))$ directly on the space of functions, without parameterizing $y(\mathbf{x})$. 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 $C(\mathbf{x}, \mathbf{x}')$ that expresses the expected covariance between the values of the function y at the points \mathbf{x} and \mathbf{x}' . The 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 – 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.

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 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 function 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.

► 45.1 Standard methods for nonlinear regression

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 the function’s value – or the value of the observation t_{N+1} – at a new point $\mathbf{x}^{(N+1)}$.

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

$$y(\mathbf{x}; \mathbf{w}) = \sum_{h=1}^H w_h \phi_h(\mathbf{x}). \quad (45.2)$$

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

$$\phi_h(\mathbf{x}) = \exp \left[-\frac{(\mathbf{x} - \mathbf{c}_h)^2}{2r^2} \right], \quad (45.3)$$

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. In neural network terms, this model is like a multilayer network whose connections from the input layer to the nonlinear hidden layer are fixed; only the output weights \mathbf{w} are adaptive.

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 45.3. Adaptive basis functions. Alternatively, we might make a function $y(\mathbf{x})$ from basis functions that 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

$$y(\mathbf{x}; \mathbf{w}) = \sum_{h=1}^H w_h^{(2)} \tanh \left(\sum_{i=1}^I w_{hi}^{(1)} x_i + w_{h0}^{(1)} \right) + w_0^{(2)} \quad (45.4)$$

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 $w_0^{(2)}$. In this model, the dependence of y on \mathbf{w} is nonlinear.

Having chosen the parameterization, we then infer the function $y(\mathbf{x}; \mathbf{w})$ by inferring the parameters \mathbf{w} . The posterior probability of the parameters is

$$P(\mathbf{w} | \mathbf{t}_N, \mathbf{X}_N) = \frac{P(\mathbf{t}_N | \mathbf{w}, \mathbf{X}_N)P(\mathbf{w})}{P(\mathbf{t}_N | \mathbf{X}_N)}. \quad (45.5)$$

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 in general not a Gaussian distribution.

The inference can be implemented in various ways. In the Laplace method, we minimize an objective function

$$M(\mathbf{w}) = -\ln [P(\mathbf{t}_N | \mathbf{w}, \mathbf{X}_N)P(\mathbf{w})] \quad (45.6)$$

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} . Alternatively we can use more general 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:

$$P(t_{N+1} | \mathbf{t}_N, \mathbf{X}_{N+1}) = \int d^H \mathbf{w} P(t_{N+1} | \mathbf{w}, \mathbf{x}^{(N+1)})P(\mathbf{w} | \mathbf{t}_N, \mathbf{X}_N). \quad (45.7)$$

If we have 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 $\mathbf{w}^{(r)}$ that are intended to be samples from the posterior distribution $P(\mathbf{w} | \mathbf{t}_N, \mathbf{X}_N)$, we approximate the predictive distribution by

$$P(t_{N+1} | \mathbf{t}_N, \mathbf{X}_{N+1}) \simeq \frac{1}{R} \sum_{r=1}^R P(t_{N+1} | \mathbf{w}^{(r)}, \mathbf{x}^{(N+1)}). \quad (45.8)$$

Nonparametric approaches.

In nonparametric methods, predictions are obtained without explicitly parameterizing the unknown function $y(\mathbf{x})$; $y(\mathbf{x})$ lives in the infinite-dimensional space of all continuous functions of \mathbf{x} . 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 $\hat{y}(\mathbf{x})$ that minimizes the functional

$$M(y(x)) = \frac{1}{2} \beta \sum_{n=1}^N (y(x^{(n)}) - t_n)^2 + \frac{1}{2} \alpha \int dx [y^{(p)}(x)]^2, \quad (45.9)$$

where $y^{(p)}$ is the p th 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 interpreted as a Bayesian method by identifying the prior for the function $y(x)$ as:

$$\ln P(y(x)|\alpha) = -\frac{1}{2} \alpha \int dx [y^{(p)}(x)]^2 + \text{const}, \quad (45.10)$$

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

$$\ln P(\mathbf{t}_N | y(x), \beta) = -\frac{1}{2} \beta \sum_{n=1}^N (y(x^{(n)}) - t_n)^2 + \text{const}. \quad (45.11)$$

[The constants in equations (45.10) and (45.11) are functions of α and β respectively. Strictly the prior (45.10) 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 (45.10).] Given this interpretation of the functions in equation (45.9), $M(y(x))$ is equal to minus 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 perspective allows us additionally to put error bars on the splines estimate and to draw typical samples from the posterior distribution, and it gives an automatic method for inferring the hyperparameters α and β .

Comments

Splines priors are Gaussian processes

The prior distribution defined in equation (45.10) is 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 $y(x)$ that can be written in the form

$$P(y(x) | \mu(x), A) = \frac{1}{Z} \exp \left[-\frac{1}{2} (y(x) - \mu(x))^\top A (y(x) - \mu(x)) \right], \quad (45.12)$$

where $\mu(x)$ is the mean function and A is a linear operator, and where the inner product of two functions $y(x)^\top z(x)$ is defined by, for example, $\int dx y(x)z(x)$.

45.1: Standard methods for nonlinear regression

Here, if we denote by D the linear operator that maps $y(x)$ to the derivative of $y(x)$, we can write equation (45.10) as

$$\ln P(y(x) | \alpha) = -\frac{1}{2} \alpha \int dx [D^p y(x)]^2 + \text{const} = -\frac{1}{2} y(x)^T A y(x) + \text{const}, \quad (45.13)$$

which has the same form as equation (45.12) with $\mu(x) = 0$, and $A \equiv [D^p]^T D^p$.

In order for the prior in equation (45.12) to be a proper prior, A must be a positive definite operator, i.e., one satisfying $y(x)^T A y(x) > 0$ for all functions $y(x)$ other than $y(x) = 0$.

Splines can be written as parametric models

Splines may be written in terms of an infinite set of fixed basis functions, as in equation (45.2), 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 $\{\cos hx, \sin hx, h=0, 1, 2, \dots\}$, so the function is

$$y(x) = \sum_{h=0}^{\infty} w_{h(\cos)} \cos(hx) + \sum_{h=1}^{\infty} w_{h(\sin)} \sin(hx). \quad (45.14)$$

Use the regularizer

$$E_W(\mathbf{w}) = \sum_{h=0}^{\infty} \frac{1}{2} h^{\frac{p}{2}} w_{h(\cos)}^2 + \sum_{h=1}^{\infty} \frac{1}{2} h^{\frac{p}{2}} w_{h(\sin)}^2 \quad (45.15)$$

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

$$P(\mathbf{w} | \alpha) = \frac{1}{Z_W(\alpha)} \exp(-\alpha E_W). \quad (45.16)$$

If $p=2$ then we have the cubic splines regularizer $E_W(\mathbf{w}) = \int y^{(2)}(x)^2 dx$, as in equation (45.9); 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.

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 (45.7). 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, the evidence $P(\mathbf{t}_N | \mathbf{X}_N)$, which appeared as the normalizing constant in equation (45.5). 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.

► **45.2 From parametric models to Gaussian processes**

Linear models

Let us consider a regression problem using H fixed basis functions, for example one-dimensional radial basis functions as defined in equation (45.3).

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 $\{\mathbf{x}_n\}$,

$$R_{nh} \equiv \phi_h(\mathbf{x}^{(n)}). \quad (45.17)$$

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

$$y_n \equiv \sum_h R_{nh} w_h. \quad (45.18)$$

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

$$P(\mathbf{w}) = \text{Normal}(\mathbf{w}; \mathbf{0}, \sigma_w^2 \mathbf{I}), \quad (45.19)$$

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

$$\mathbf{Q} = \langle \mathbf{y}\mathbf{y}^\top \rangle = \langle \mathbf{R}\mathbf{w}\mathbf{w}^\top \mathbf{R}^\top \rangle = \mathbf{R} \langle \mathbf{w}\mathbf{w}^\top \rangle \mathbf{R}^\top \quad (45.20)$$

$$= \sigma_w^2 \mathbf{R}\mathbf{R}^\top. \quad (45.21)$$

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

$$P(\mathbf{y}) = \text{Normal}(\mathbf{y}; \mathbf{0}, \mathbf{Q}) = \text{Normal}(\mathbf{y}; \mathbf{0}, \sigma_w^2 \mathbf{R}\mathbf{R}^\top). \quad (45.22)$$

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 $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(N)}$, the density $P(y(\mathbf{x}^{(1)}), y(\mathbf{x}^{(2)}), \dots, y(\mathbf{x}^{(N)}))$ 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,

$$P(\mathbf{t}) = \text{Normal}(\mathbf{t}; \mathbf{0}, \mathbf{Q} + \sigma_\nu^2 \mathbf{I}). \quad (45.23)$$

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

$$\mathbf{C} = \mathbf{Q} + \sigma_\nu^2 \mathbf{I} = \sigma_w^2 \mathbf{R}\mathbf{R}^\top + \sigma_\nu^2 \mathbf{I}. \quad (45.24)$$

Whether or not \mathbf{Q} has full rank, the covariance matrix \mathbf{C} 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

$$Q_{nn'} = [\sigma_w^2 \mathbf{R}\mathbf{R}^\top]_{nn'} = \sigma_w^2 \sum_h \phi_h(\mathbf{x}^{(n)}) \phi_h(\mathbf{x}^{(n')}) \quad (45.25)$$

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

$$C_{nn'} = \sigma_w^2 \sum_h \phi_h(\mathbf{x}^{(n)})\phi_h(\mathbf{x}^{(n')}) + \delta_{nn'}\sigma_v^2, \quad (45.26)$$

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

Example 45.4. 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 with the basis function labelled h being centred on the point $x = h$ and take the limit $H \rightarrow \infty$, 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, and S is a constant; then

$$Q_{nn'} = S \int_{h_{\min}}^{h_{\max}} dh \phi_h(x^{(n)})\phi_h(x^{(n')}) \quad (45.27)$$

$$= S \int_{h_{\min}}^{h_{\max}} dh \exp\left[-\frac{(x^{(n)} - h)^2}{2r^2}\right] \exp\left[-\frac{(x^{(n')} - h)^2}{2r^2}\right]. \quad (45.28)$$

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

$$Q_{nn'} = \sqrt{\pi r^2} S \exp\left[-\frac{(x^{(n')} - x^{(n)})^2}{4r^2}\right]. \quad (45.29)$$

We are arriving at a new perspective on the interpolation problem. Instead of specifying the prior distribution on functions in terms of basis functions and priors on parameters, the prior can be summarized simply by a covariance function,

$$C(x^{(n)}, x^{(n')}) \equiv \theta_1 \exp\left[-\frac{(x^{(n')} - x^{(n)})^2}{4r^2}\right], \quad (45.30)$$

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 $C(\mathbf{x}, \mathbf{x}')$ – 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

$$Q_{nn'} = C(\mathbf{x}^{(n)}, \mathbf{x}^{(n')}) \quad (45.31)$$

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

$$C_{nn'} = C(\mathbf{x}^{(n)}, \mathbf{x}^{(n')}) + \sigma_v^2 \delta_{nn'}. \quad (45.32)$$

In conclusion, the prior probability of the N target values \mathbf{t} in the data set is:

$$P(\mathbf{t}) = \text{Normal}(\mathbf{t}; \mathbf{0}, \mathbf{C}) = \frac{1}{Z} e^{-\frac{1}{2}\mathbf{t}^T \mathbf{C}^{-1} \mathbf{t}}. \quad (45.33)$$

Samples from this Gaussian process and a few other simple Gaussian processes are displayed in figure 45.1.

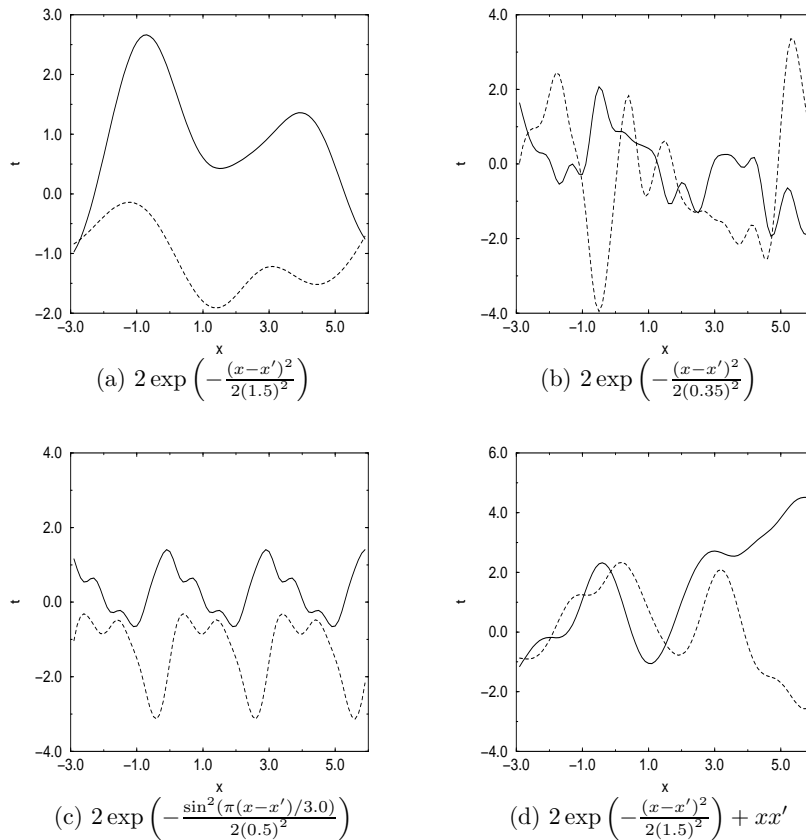


Figure 45.1. Samples drawn from Gaussian process priors. Each panel shows two functions drawn from a Gaussian process prior. The four corresponding covariance functions are given below each plot. The decrease in length scale from (a) to (b) produces more rapidly fluctuating functions. The periodic properties of the covariance function in (c) can 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. From Gibbs (1997).

Multilayer neural networks and Gaussian processes

Figures 44.2 and 44.3 show some random samples from the prior distribution over functions defined by a selection of standard multilayer perceptrons with large numbers of hidden units. Those samples don't seem a million miles away from the Gaussian process samples of figure 45.1. And indeed Neal (1996) showed that the properties of a neural network with one hidden layer (as in equation (45.4)) 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.

► 45.3 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 (45.32) our task is to infer t_{N+1} given the observed vector \mathbf{t}_N . The joint density $P(t_{N+1}, \mathbf{t}_N)$ is a Gaussian; so the conditional distribution

$$P(t_{N+1} | \mathbf{t}_N) = \frac{P(t_{N+1}, \mathbf{t}_N)}{P(\mathbf{t}_N)} \quad (45.34)$$

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 $\mathbf{t}_{N+1} \equiv (t_1, \dots, t_{N+1})^T$. We define submatrices of \mathbf{C}_{N+1} as follows:

$$\mathbf{C}_{N+1} \equiv \begin{bmatrix} \left[\begin{array}{c} \mathbf{C}_N \\ \mathbf{k} \end{array} \right] & \left[\begin{array}{c} \mathbf{k} \\ \kappa \end{array} \right] \\ \left[\begin{array}{c} \mathbf{k}^T \\ \kappa \end{array} \right] & \left[\begin{array}{c} \kappa \end{array} \right] \end{bmatrix}. \quad (45.35)$$

The posterior distribution (45.34) is given by

$$P(t_{N+1} | \mathbf{t}_N) \propto \exp \left[-\frac{1}{2} \begin{bmatrix} \mathbf{t}_N & t_{N+1} \end{bmatrix} \mathbf{C}_{N+1}^{-1} \begin{bmatrix} \mathbf{t}_N \\ t_{N+1} \end{bmatrix} \right]. \quad (45.36)$$

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}_{N+1}^{-1} in terms of \mathbf{C}_N and \mathbf{C}_N^{-1} using the partitioned inverse equations (Barnett, 1979)

$$\mathbf{C}_{N+1}^{-1} = \begin{bmatrix} \mathbf{M} & \mathbf{m} \\ \mathbf{m}^T & m \end{bmatrix} \quad (45.37)$$

where

$$m = (\kappa - \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{k})^{-1} \quad (45.38)$$

$$\mathbf{m} = -m \mathbf{C}_N^{-1} \mathbf{k} \quad (45.39)$$

$$\mathbf{M} = \mathbf{C}_N^{-1} + \frac{1}{m} \mathbf{m} \mathbf{m}^T. \quad (45.40)$$

When we substitute this matrix into equation (45.36) we find

$$P(t_{N+1} | \mathbf{t}_N) = \frac{1}{Z} \exp \left[-\frac{(t_{N+1} - \hat{t}_{N+1})^2}{2\sigma_{\hat{t}_{N+1}}^2} \right] \quad (45.41)$$

where

$$\hat{t}_{N+1} = \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{t}_N \quad (45.42)$$

$$\sigma_{\hat{t}_{N+1}}^2 = \kappa - \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{k}. \quad (45.43)$$

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 implement a model with a number of basis functions H much larger than the number of data points N , with the computational requirement being of order N^3 , independent of H . [We'll discuss ways of reducing this cost later.]

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

► 45.4 Examples of covariance functions

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 $\boldsymbol{\theta}$. The covariance matrix of \mathbf{t} has entries given by

$$C_{mn} = C(\mathbf{x}^{(m)}, \mathbf{x}^{(n)}; \boldsymbol{\theta}) + \delta_{mn} \mathcal{N}(\mathbf{x}^{(n)}; \boldsymbol{\theta}) \quad (45.44)$$

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

$$\mathcal{N}(\mathbf{x}; \boldsymbol{\theta}) = \begin{cases} \theta_3 & \text{for input-independent noise} \\ \exp\left(\sum_{j=1}^J \beta_j \phi_j(\mathbf{x})\right) & \text{for input-dependent noise.} \end{cases} \quad (45.45)$$

The continuity properties of C determine the continuity properties of typical samples from the Gaussian process prior. An encyclopaedic paper on Gaussian processes giving many valid covariance functions has been written by Abrahamson (1997).

Stationary covariance functions

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

$$C(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}) = D(\mathbf{x} - \mathbf{x}'; \boldsymbol{\theta}) \quad (45.46)$$

for some function D , i.e., the covariance is a function of separation only, also known as the autocovariance function. If additionally C depends only 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 to invent a positive function of frequency and define D to be its inverse Fourier transform.

Example 45.5. Let the power spectrum be 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 (45.30).

Generalizing slightly, a popular form for C with hyperparameters $\boldsymbol{\theta} = (\theta_1, \theta_2, \{r_i\})$ is

$$C(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}) = \theta_1 \exp\left[-\frac{1}{2} \sum_{i=1}^I \frac{(x_i - x'_i)^2}{r_i^2}\right] + \theta_2. \quad (45.47)$$

\mathbf{x} is an I -dimensional vector and r_i is a length scale associated with input x_i , the lengthscale in that direction on 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 method for neural networks (MacKay, 1994a; 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 (45.25) and consider the basis function $\phi(\mathbf{x}) = 1$.

Another stationary covariance function is

$$C(x, x') = \exp(-|x - x'|^\nu) \quad 0 < \nu \leq 2. \quad (45.48)$$

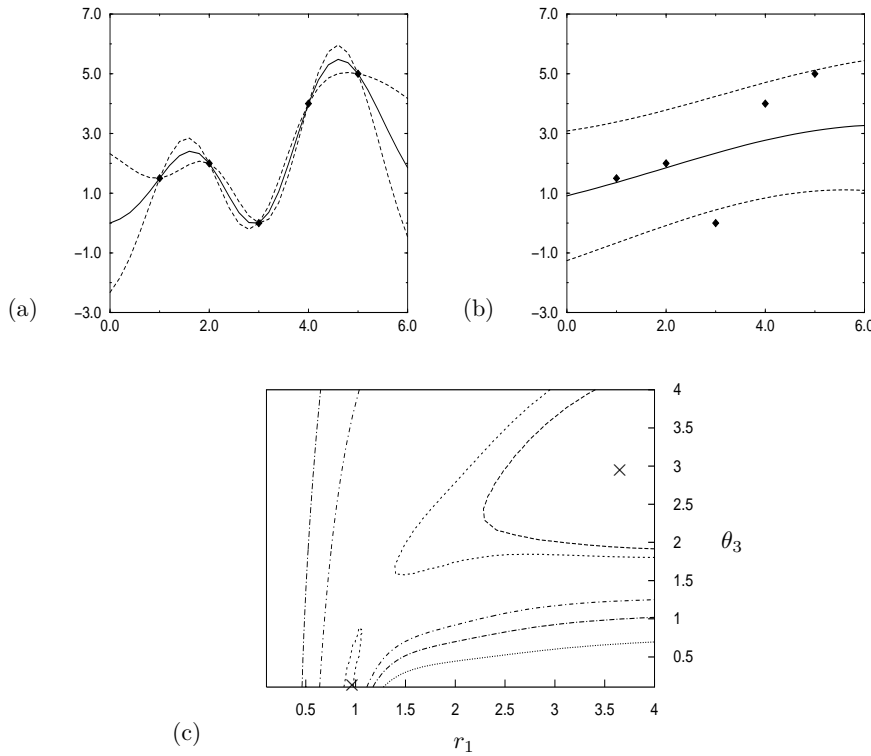


Figure 45.2. Multimodal likelihood functions for Gaussian processes. A data set of five points is modelled with the simple covariance function (45.47), with one hyperparameter θ_3 controlling the noise variance. Panels a and b show the most probable interpolant and its 1σ error bars when the hyperparameters θ are set to two different values that (locally) maximize the likelihood $P(\mathbf{t}_N | \mathbf{X}_N, \theta)$: (a) $r_1 = 0.95$, $\theta_3 = 0.0$; (b) $r_1 = 3.5$, $\theta_3 = 3.0$. Panel c shows a contour plot of the likelihood as a function of r_1 and θ_3 , with the two maxima shown by crosses. From Gibbs (1997).

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

$$C(\mathbf{x}, \mathbf{x}'; \theta) = \theta_1 \exp \left[-\frac{1}{2} \sum_i \left(\frac{\sin \left(\frac{\pi}{\lambda_i} (x_i - x'_i) \right)}{r_i} \right)^2 \right]. \quad (45.49)$$

Figure 45.1 shows some random samples drawn from Gaussian processes with a variety of different covariance functions.

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^2 and σ_c^2 respectively then the plane has a covariance function

$$C_{\text{lin}}(\mathbf{x}, \mathbf{x}'; \{\sigma_w, \sigma_c\}) = \sum_{i=1}^I \sigma_w^2 x_i x'_i + \sigma_c^2. \quad (45.50)$$

An example of random sample functions incorporating the linear term can be seen in figure 45.1d.

► 45.5 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. It is a complexity control problem, one that is solved nicely by the Bayesian Occam's razor.

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

$$P(t_{N+1} | \mathbf{x}_{N+1}, \mathcal{D}) = \int P(t_{N+1} | \mathbf{x}_{N+1}, \boldsymbol{\theta}, \mathcal{D}) P(\boldsymbol{\theta} | \mathcal{D}) d\boldsymbol{\theta}. \quad (45.51)$$

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.

$$P(t_{N+1} | \mathbf{x}_{N+1}, \mathcal{D}) \simeq P(t_{N+1} | \mathbf{x}_{N+1}, \mathcal{D}, \boldsymbol{\theta}_{\text{MP}}) \quad (45.52)$$

2. Or we can perform the integration over $\boldsymbol{\theta}$ numerically using Monte Carlo methods (Williams and Rasmussen, 1996; Neal, 1997b).

Either of these approaches is implemented most efficiently if the gradient of the posterior probability of $\boldsymbol{\theta}$ can be evaluated.

Gradient

The posterior probability of $\boldsymbol{\theta}$ is

$$P(\boldsymbol{\theta} | \mathcal{D}) \propto P(\mathbf{t}_N | \mathbf{X}_N, \boldsymbol{\theta}) P(\boldsymbol{\theta}). \quad (45.53)$$

The log of the first term (the evidence for the hyperparameters) is

$$\ln P(\mathbf{t}_N | \mathbf{X}_N, \boldsymbol{\theta}) = -\frac{1}{2} \ln \det \mathbf{C}_N - \frac{1}{2} \mathbf{t}_N^T \mathbf{C}_N^{-1} \mathbf{t}_N - \frac{N}{2} \ln 2\pi, \quad (45.54)$$

and its derivative with respect to a hyperparameter θ is

$$\frac{\partial}{\partial \theta} \ln P(\mathbf{t}_N | \mathbf{X}_N, \boldsymbol{\theta}) = -\frac{1}{2} \text{Trace} \left(\mathbf{C}_N^{-1} \frac{\partial \mathbf{C}_N}{\partial \theta} \right) + \frac{1}{2} \mathbf{t}_N^T \mathbf{C}_N^{-1} \frac{\partial \mathbf{C}_N}{\partial \theta} \mathbf{C}_N^{-1} \mathbf{t}_N. \quad (45.55)$$

Comments

Assuming that finding the derivatives of the priors is straightforward, we can now search for $\boldsymbol{\theta}_{\text{MP}}$. However there are two problems that we need to be aware of. Firstly, as illustrated in figure 45.2, the evidence may be multimodal. Suitable priors and sensible optimization strategies often eliminate poor optima. 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 (such as Cholesky decomposition, LU decomposition or Gauss–Jordan elimination) has an associated computational cost that is of order N^3 and so calculating gradients becomes time consuming for large training data sets. Approximate methods for implementing the predictions (equations (45.42) and (45.43)) and gradient computation (equation (45.55)) are an active research area. One approach based on the ideas of Skilling (1993) makes approximations to $\mathbf{C}^{-1} \mathbf{t}$ and $\text{Trace } \mathbf{C}^{-1}$ using iterative methods with cost $\mathcal{O}(N^2)$ (Gibbs and MacKay, 1996; Gibbs, 1997). Further references on this topic are given at the end of the chapter.

► 45.6 Classification

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

In 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

$$P(t_n = 1 | a_n) = \frac{1}{1 + e^{-a_n}}. \quad (45.56)$$

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.

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 (45.56) 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 (Chapter 27). Neal (1997b) has implemented a Monte Carlo approach to implementing a Gaussian process classifier. Gibbs and MacKay (2000) have implemented another cheap and cheerful approach based on the methods of Jaakkola and Jordan (section 33.8). In this *variational Gaussian process classifier*, 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.

Multi-class classification problems can also be solved with Monte Carlo methods (Neal, 1997b) and variational methods (Gibbs, 1997).

► 45.7 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. 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 straightforward automatic relevance determination model.

One obvious problem with Gaussian processes is the computational cost associated with inverting an $N \times N$ matrix. The cost of direct methods of inversion becomes prohibitive when the number of data points N is greater than about 1000.

Have we thrown the baby out with the bath water?

According to the hype of 1987, neural networks were meant to be intelligent models that discovered features and patterns in data. Gaussian processes in

contrast are simply smoothing devices. How can Gaussian processes possibly replace neural networks? Were neural networks over-hyped, or have we underestimated the power of smoothing methods?

I think both these propositions are true. The success of Gaussian processes shows 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 that Gaussian processes will solve. But maybe multilayer perceptrons can't solve them either. Perhaps a fresh start is needed, approaching the problem of machine learning from a paradigm different from the supervised feedforward mapping.

Further reading

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 a 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. Kalman filters are widely used to implement inferences for stationary one-dimensional Gaussian processes, and are popular models for speech and music modelling (Bar-Shalom and Fortmann, 1988). 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. See also O’Hagan (1978).

The idea of replacing supervised neural networks by Gaussian processes was first explored by Williams and Rasmussen (1996) and Neal (1997b). A thorough comparison of Gaussian processes with other methods such as neural networks and MARS was made by Rasmussen (1996). Methods for reducing the complexity of data modelling with Gaussian processes remain an active research area (Poggio and Girosi, 1990; Luo and Wahba, 1997; Tresp, 2000; Williams and Seeger, 2001; Smola and Bartlett, 2001; Rasmussen, 2002; Seeger *et al.*, 2003; Opper and Winther, 2000).

A longer review of Gaussian processes is in (MacKay, 1998b). A review paper on regression with complexity control using hierarchical Bayesian models is (MacKay, 1992a).

Gaussian processes and *support vector learning machines* (Scholkopf *et al.*, 1995; Vapnik, 1995) have a lot in common. Both are kernel-based predictors, the kernel being another name for the covariance function. A Bayesian version of support vectors, exploiting this connection, can be found in (Chu *et al.*, 2001; Chu *et al.*, 2002; Chu *et al.*, 2003b; Chu *et al.*, 2003a).