6 Kernel Methods

In Chapters 3 and 4, we considered linear parametric models for regression and classification in which the form of the mapping $y(\mathbf{x}, \mathbf{w})$ from input $\mathbf{x}$ to output y is governed by a vector $\mathbf{w}$ of adaptive parameters. During the learning phase, a set of training data is used either to obtain a point estimate of the parameter vector or to determine a posterior distribution over this vector. The training data is then discarded, and predictions for new inputs are based purely on the learned parameter vector $\mathbf{w}$. This approach is also used in nonlinear parametric models such as neural networks.

Chapter 5

However, there is a class of pattern recognition techniques, in which the training data points, or a subset of them, are kept and used also during the prediction phase. For instance, the Parzen probability density model comprised a linear combination of 'kernel' functions each one centred on one of the training data points. Similarly, in Section 2.5.2 we introduced a simple technique for classification called nearest neighbours, which involved assigning to each new test vector the same label as the

Section 2.5.1

closest example from the training set. These are examples of *memory-based* methods that involve storing the entire training set in order to make predictions for future data points. They typically require a metric to be defined that measures the similarity of any two vectors in input space, and are generally fast to 'train' but slow at making predictions for test data points.

Many linear parametric models can be re-cast into an equivalent 'dual representation' in which the predictions are also based on linear combinations of a *kernel function* evaluated at the training data points. As we shall see, for models which are based on a fixed nonlinear *feature space* mapping $\boldsymbol{\phi}(\mathbf{x})$, the kernel function is given by the relation

$$k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}'). \tag{6.1}$$

From this definition, we see that the kernel is a symmetric function of its arguments so that $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$. The kernel concept was introduced into the field of pattern recognition by Aizerman *et al.* (1964) in the context of the method of potential functions, so-called because of an analogy with electrostatics. Although neglected for many years, it was re-introduced into machine learning in the context of large-margin classifiers by Boser *et al.* (1992) giving rise to the technique of *support vector machines*. Since then, there has been considerable interest in this topic, both in terms of theory and applications. One of the most significant developments has been the extension of kernels to handle symbolic objects, thereby greatly expanding the range of problems that can be addressed.

Chapter 7

The simplest example of a kernel function is obtained by considering the identity mapping for the feature space in (6.1) so that $\boldsymbol{\phi}(\mathbf{x}) = \mathbf{x}$, in which case $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^{\mathrm{T}}\mathbf{x}'$. We shall refer to this as the linear kernel.

The concept of a kernel formulated as an inner product in a feature space allows us to build interesting extensions of many well-known algorithms by making use of the *kernel trick*, also known as *kernel substitution*. The general idea is that, if we have an algorithm formulated in such a way that the input vector $\mathbf{x}$ enters only in the form of scalar products, then we can replace that scalar product with some other choice of kernel. For instance, the technique of kernel substitution can be applied to principal component analysis in order to develop a nonlinear variant of PCA (Schölkopf *et al.*, 1998). Other examples of kernel substitution include nearest-neighbour classifiers and the kernel Fisher discriminant (Mika *et al.*, 1999; Roth and Steinhage, 2000; Baudat and Anouar, 2000).

Section 12.3

There are numerous forms of kernel functions in common use, and we shall encounter several examples in this chapter. Many have the property of being a function only of the difference between the arguments, so that $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} - \mathbf{x}')$, which are known as *stationary* kernels because they are invariant to translations in input space. A further specialization involves *homogeneous* kernels, also known as *radial basis functions*, which depend only on the magnitude of the distance (typically Euclidean) between the arguments so that $k(\mathbf{x}, \mathbf{x}') = k(\|\mathbf{x} - \mathbf{x}'\|)$.

Section 6.3

For recent textbooks on kernel methods, see Schölkopf and Smola (2002), Herbrich (2002), and Shawe-Taylor and Cristianini (2004).

6.1. Dual Representations

Many linear models for regression and classification can be reformulated in terms of a dual representation in which the kernel function arises naturally. This concept will play an important role when we consider support vector machines in the next chapter. Here we consider a linear regression model whose parameters are determined by minimizing a regularized sum-of-squares error function given by

$$J(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left\{\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) - t_n\right\}^2 + \frac{\lambda}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} \tag{6.2}$$

where $\lambda \geqslant 0$. If we set the gradient of $J(\mathbf{w})$ with respect to $\mathbf{w}$ equal to zero, we see that the solution for $\mathbf{w}$ takes the form of a linear combination of the vectors $\boldsymbol{\phi}(\mathbf{x}_n)$, with coefficients that are functions of $\mathbf{w}$, of the form

$$\mathbf{w} = -\frac{1}{\lambda}\sum_{n=1}^{N}\left\{\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) - t_n\right\}\boldsymbol{\phi}(\mathbf{x}_n) = \sum_{n=1}^{N} a_n\boldsymbol{\phi}(\mathbf{x}_n) = \boldsymbol{\Phi}^{\mathrm{T}}\mathbf{a} \tag{6.3}$$

where $\boldsymbol{\Phi}$ is the design matrix, whose n^{th} row is given by $\boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}}$. Here the vector $\mathbf{a} = (a_1, \ldots, a_N)^{\mathrm{T}}$, and we have defined

$$a_n = -\frac{1}{\lambda}\left\{\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) - t_n\right\}. \tag{6.4}$$

Instead of working with the parameter vector $\mathbf{w}$, we can now reformulate the least-squares algorithm in terms of the parameter vector $\mathbf{a}$, giving rise to a *dual representation*. If we substitute $\mathbf{w} = \boldsymbol{\Phi}^{\mathrm{T}}\mathbf{a}$ into $J(\mathbf{w})$, we obtain

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{a} - \mathbf{a}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t} + \frac{1}{2}\mathbf{t}^{\mathrm{T}}\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{a} \tag{6.5}$$

where $\mathbf{t} = (t_1, \ldots, t_N)^{\mathrm{T}}$. We now define the *Gram* matrix $\mathbf{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}$, which is an $N \times N$ symmetric matrix with elements

$$K_{nm} = \boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m) \tag{6.6}$$

where we have introduced the *kernel function* $k(\mathbf{x}, \mathbf{x}')$ defined by (6.1). In terms of the Gram matrix, the sum-of-squares error function can be written as

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^{\mathrm{T}}\mathbf{K}\mathbf{K}\mathbf{a} - \mathbf{a}^{\mathrm{T}}\mathbf{K}\mathbf{t} + \frac{1}{2}\mathbf{t}^{\mathrm{T}}\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^{\mathrm{T}}\mathbf{K}\mathbf{a}. \tag{6.7}$$

Setting the gradient of $J(\mathbf{a})$ with respect to $\mathbf{a}$ to zero, we obtain the following solution

$$\mathbf{a} = (\mathbf{K} + \lambda\mathbf{I}_N)^{-1}\mathbf{t}. \tag{6.8}$$

If we substitute this back into the linear regression model, we obtain the following prediction for a new input $\mathbf{x}$

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}) = \mathbf{a}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\phi}(\mathbf{x}) = \mathbf{k}(\mathbf{x})^{\mathrm{T}}\left(\mathbf{K} + \lambda\mathbf{I}_N\right)^{-1}\mathbf{t} \tag{6.9}$$

where we have defined the vector $\mathbf{k}(\mathbf{x})$ with elements $k_n(\mathbf{x}) = k(\mathbf{x}_n, \mathbf{x})$. Thus we see that the dual formulation allows the solution to the least-squares problem to be expressed entirely in terms of the kernel function $k(\mathbf{x}, \mathbf{x}')$. This is known as a dual formulation because, by noting that the solution for $\mathbf{a}$ can be expressed as a linear combination of the elements of $\boldsymbol{\phi}(\mathbf{x})$, we recover the original formulation in terms of the parameter vector $\mathbf{w}$. Note that the prediction at $\mathbf{x}$ is given by a linear combination of the target values from the training set. In fact, we have already obtained this result, using a slightly different notation, in Section 3.3.3.

Exercise 6.1

In the dual formulation, we determine the parameter vector $\mathbf{a}$ by inverting an $N \times N$ matrix, whereas in the original parameter space formulation we had to invert an $M \times M$ matrix in order to determine $\mathbf{w}$. Because N is typically much larger than M, the dual formulation does not seem to be particularly useful. However, the advantage of the dual formulation, as we shall see, is that it is expressed entirely in terms of the kernel function $k(\mathbf{x}, \mathbf{x}')$. We can therefore work directly in terms of kernels and avoid the explicit introduction of the feature vector $\boldsymbol{\phi}(\mathbf{x})$, which allows us implicitly to use feature spaces of high, even infinite, dimensionality.

The existence of a dual representation based on the Gram matrix is a property of many linear models, including the perceptron. In Section 6.4, we will develop a duality between probabilistic linear models for regression and the technique of Gaussian processes. Duality will also play an important role when we discuss support vector machines in Chapter 7.

Exercise 6.2

6.2. Constructing Kernels

In order to exploit kernel substitution, we need to be able to construct valid kernel functions. One approach is to choose a feature space mapping $\boldsymbol{\phi}(\mathbf{x})$ and then use this to find the corresponding kernel, as is illustrated in Figure 6.1. Here the kernel function is defined for a one-dimensional input space by

$$k(x, x') = \boldsymbol{\phi}(x)^{\mathrm{T}}\boldsymbol{\phi}(x') = \sum_{i=1}^{M} \phi_i(x)\phi_i(x') \tag{6.10}$$

where $\phi_i(x)$ are the basis functions.

An alternative approach is to construct kernel functions directly. In this case, we must ensure that the function we choose is a valid kernel, in other words that it corresponds to a scalar product in some (perhaps infinite dimensional) feature space. As a simple example, consider a kernel function given by

$$k(\mathbf{x}, \mathbf{z}) = \left(\mathbf{x}^{\mathrm{T}}\mathbf{z}\right)^2. \tag{6.11}$$

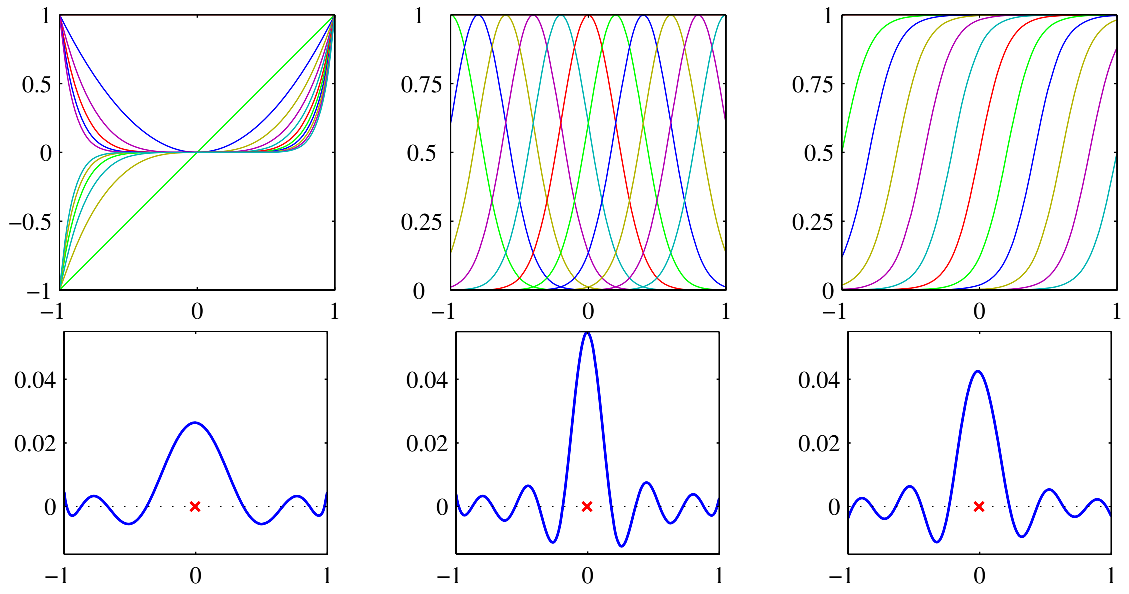

Figure 6.1 Illustration of the construction of kernel functions starting from a corresponding set of basis functions. In each column the lower plot shows the kernel function $k(x, x')$ defined by (6.10) plotted as a function of x for $x' = 0$, while the upper plot shows the corresponding basis functions given by polynomials (left column), 'Gaussians' (centre column), and logistic sigmoids (right column).

If we take the particular case of a two-dimensional input space $\mathbf{x} = (x_1, x_2)$ we can expand out the terms and thereby identify the corresponding nonlinear feature mapping

$$\begin{aligned} k(\mathbf{x}, \mathbf{z}) &= \left(\mathbf{x}^{\mathrm{T}}\mathbf{z}\right)^2 = (x_1 z_1 + x_2 z_2)^2 \\ &= x_1^2 z_1^2 + 2x_1 z_1 x_2 z_2 + x_2^2 z_2^2 \\ &= (x_1^2, \sqrt{2}x_1 x_2, x_2^2)(z_1^2, \sqrt{2}z_1 z_2, z_2^2)^{\mathrm{T}} \\ &= \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{z}). \end{aligned} \tag{6.12}$$

We see that the feature mapping takes the form $\boldsymbol{\phi}(\mathbf{x}) = (x_1^2, \sqrt{2}x_1 x_2, x_2^2)^{\mathrm{T}}$ and therefore comprises all possible second order terms, with a specific weighting between them.

More generally, however, we need a simple way to test whether a function constitutes a valid kernel without having to construct the function $\boldsymbol{\phi}(\mathbf{x})$ explicitly. A necessary and sufficient condition for a function $k(\mathbf{x}, \mathbf{x}')$ to be a valid kernel (Shawe-Taylor and Cristianini, 2004) is that the Gram matrix $\mathbf{K}$, whose elements are given by $k(\mathbf{x}_n, \mathbf{x}_m)$, should be positive semidefinite for all possible choices of the set $\{\mathbf{x}_n\}$. Note that a positive semidefinite matrix is not the same thing as a matrix whose elements are nonnegative.

Appendix C

One powerful technique for constructing new kernels is to build them out of simpler kernels as building blocks. This can be done using the following properties:

Techniques for Constructing New Kernels.

Given valid kernels $k_1(\mathbf{x}, \mathbf{x}')$ and $k_2(\mathbf{x}, \mathbf{x}')$, the following new kernels will also be valid:

$$\begin{aligned}
k(\mathbf{x}, \mathbf{x}') &= ck_1(\mathbf{x}, \mathbf{x}') && (6.13)\\
k(\mathbf{x}, \mathbf{x}') &= f(\mathbf{x})k_1(\mathbf{x}, \mathbf{x}')f(\mathbf{x}') && (6.14)\\
k(\mathbf{x}, \mathbf{x}') &= q\left(k_1(\mathbf{x}, \mathbf{x}')\right) && (6.15)\\
k(\mathbf{x}, \mathbf{x}') &= \exp\left(k_1(\mathbf{x}, \mathbf{x}')\right) && (6.16)\\
k(\mathbf{x}, \mathbf{x}') &= k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') && (6.17)\\
k(\mathbf{x}, \mathbf{x}') &= k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}') && (6.18)\\
k(\mathbf{x}, \mathbf{x}') &= k_3\left(\boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}')\right) && (6.19)\\
k(\mathbf{x}, \mathbf{x}') &= \mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x}' && (6.20)\\
k(\mathbf{x}, \mathbf{x}') &= k_a(\mathbf{x}_a, \mathbf{x}'_a) + k_b(\mathbf{x}_b, \mathbf{x}'_b) && (6.21)\\
k(\mathbf{x}, \mathbf{x}') &= k_a(\mathbf{x}_a, \mathbf{x}'_a)k_b(\mathbf{x}_b, \mathbf{x}'_b) && (6.22)
\end{aligned}$$

where $c > 0$ is a constant, $f(\cdot)$ is any function, $q(\cdot)$ is a polynomial with nonnegative coefficients, $\boldsymbol{\phi}(\mathbf{x})$ is a function from $\mathbf{x}$ to $\mathbb{R}^M$, $k_3(\cdot, \cdot)$ is a valid kernel in $\mathbb{R}^M$, $\mathbf{A}$ is a symmetric positive semidefinite matrix, $\mathbf{x}_a$ and $\mathbf{x}_b$ are variables (not necessarily disjoint) with $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$, and k_a and k_b are valid kernel functions over their respective spaces.

Equipped with these properties, we can now embark on the construction of more complex kernels appropriate to specific applications. We require that the kernel $k(\mathbf{x}, \mathbf{x}')$ be symmetric and positive semidefinite and that it expresses the appropriate form of similarity between $\mathbf{x}$ and $\mathbf{x}'$ according to the intended application. Here we consider a few common examples of kernel functions. For a more extensive discussion of 'kernel engineering', see Shawe-Taylor and Cristianini (2004).

We saw that the simple polynomial kernel $k(\mathbf{x}, \mathbf{x}') = \left(\mathbf{x}^{\mathrm{T}}\mathbf{x}'\right)^2$ contains only terms of degree two. If we consider the slightly generalized kernel $k(\mathbf{x}, \mathbf{x}') = \left(\mathbf{x}^{\mathrm{T}}\mathbf{x}' + c\right)^2$ with $c > 0$, then the corresponding feature mapping $\boldsymbol{\phi}(\mathbf{x})$ contains constant and linear terms as well as terms of order two. Similarly, $k(\mathbf{x}, \mathbf{x}') = \left(\mathbf{x}^{\mathrm{T}}\mathbf{x}'\right)^M$ contains all monomials of order M. For instance, if $\mathbf{x}$ and $\mathbf{x}'$ are two images, then the kernel represents a particular weighted sum of all possible products of M pixels in the first image with M pixels in the second image. This can similarly be generalized to include all terms up to degree M by considering $k(\mathbf{x}, \mathbf{x}') = \left(\mathbf{x}^{\mathrm{T}}\mathbf{x}' + c\right)^M$ with $c > 0$. Using the results (6.17) and (6.18) for combining kernels we see that these will all be valid kernel functions.

Another commonly used kernel takes the form

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\|\mathbf{x} - \mathbf{x}'\|^2/2\sigma^2\right) \tag{6.23}$$

and is often called a 'Gaussian' kernel. Note, however, that in this context it is not interpreted as a probability density, and hence the normalization coefficient is omitted. We can see that this is a valid kernel by expanding the square

$$\|\mathbf{x} - \mathbf{x}'\|^2 = \mathbf{x}^{\mathrm{T}}\mathbf{x} + (\mathbf{x}')^{\mathrm{T}}\mathbf{x}' - 2\mathbf{x}^{\mathrm{T}}\mathbf{x}' \tag{6.24}$$

to give

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\mathbf{x}^{\mathrm{T}}\mathbf{x}/2\sigma^2\right) \exp\left(\mathbf{x}^{\mathrm{T}}\mathbf{x}'/\sigma^2\right) \exp\left(-(\mathbf{x}')^{\mathrm{T}}\mathbf{x}'/2\sigma^2\right) \tag{6.25}$$

and then making use of (6.14) and (6.16), together with the validity of the linear kernel $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^{\mathrm{T}}\mathbf{x}'$. Note that the feature vector that corresponds to the Gaussian kernel has infinite dimensionality.

Exercise 6.11

The Gaussian kernel is not restricted to the use of Euclidean distance. If we use kernel substitution in (6.24) to replace $\mathbf{x}^{\mathrm{T}}\mathbf{x}'$ with a nonlinear kernel $\kappa(\mathbf{x}, \mathbf{x}')$, we obtain

$$k(\mathbf{x}, \mathbf{x}') = \exp\left\{-\frac{1}{2\sigma^2}\left(\kappa(\mathbf{x}, \mathbf{x}) + \kappa(\mathbf{x}', \mathbf{x}') - 2\kappa(\mathbf{x}, \mathbf{x}')\right)\right\}. \tag{6.26}$$

An important contribution to arise from the kernel viewpoint has been the extension to inputs that are symbolic, rather than simply vectors of real numbers. Kernel functions can be defined over objects as diverse as graphs, sets, strings, and text documents. Consider, for instance, a fixed set and define a nonvectorial space consisting of all possible subsets of this set. If A_1 and A_2 are two such subsets then one simple choice of kernel would be

$$k(A_1, A_2) = 2^{|A_1 \cap A_2|} \tag{6.27}$$

where $A_1 \cap A_2$ denotes the intersection of sets A_1 and A_2, and $|A|$ denotes the number of subsets in A. This is a valid kernel function because it can be shown to correspond to an inner product in a feature space.

Exercise 6.12

One powerful approach to the construction of kernels starts from a probabilistic generative model (Haussler, 1999), which allows us to apply generative models in a discriminative setting. Generative models can deal naturally with missing data and in the case of hidden Markov models can handle sequences of varying length. By contrast, discriminative models generally give better performance on discriminative tasks than generative models. It is therefore of some interest to combine these two approaches (Lasserre *et al.*, 2006). One way to combine them is to use a generative model to define a kernel, and then use this kernel in a discriminative approach.

Given a generative model $p(\mathbf{x})$ we can define a kernel by

$$k(\mathbf{x}, \mathbf{x}') = p(\mathbf{x})p(\mathbf{x}'). \tag{6.28}$$

This is clearly a valid kernel function because we can interpret it as an inner product in the one-dimensional feature space defined by the mapping $p(\mathbf{x})$. It says that two inputs $\mathbf{x}$ and $\mathbf{x}'$ are similar if they both have high probabilities. We can use (6.13) and (6.17) to extend this class of kernels by considering sums over products of different probability distributions, with positive weighting coefficients $p(i)$, of the form

$$k(\mathbf{x}, \mathbf{x}') = \sum_i p(\mathbf{x}|i)p(\mathbf{x}'|i)p(i). \tag{6.29}$$

This is equivalent, up to an overall multiplicative constant, to a mixture distribution in which the components factorize, with the index i playing the role of a 'latent' variable. Two inputs $\mathbf{x}$ and $\mathbf{x}'$ will give a large value for the kernel function, and hence appear similar, if they have significant probability under a range of different components. Taking the limit of an infinite sum, we can also consider kernels of the form

Section 9.2

$$k(\mathbf{x}, \mathbf{x}') = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{x}'|\mathbf{z})p(\mathbf{z})\,\mathrm{d}\mathbf{z} \tag{6.30}$$

where $\mathbf{z}$ is a continuous latent variable.

Now suppose that our data consists of ordered sequences of length L so that an observation is given by $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_L\}$. A popular generative model for sequences is the hidden Markov model, which expresses the distribution $p(\mathbf{X})$ as a marginalization over a corresponding sequence of hidden states $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_L\}$. We can use this approach to define a kernel function measuring the similarity of two sequences $\mathbf{X}$ and $\mathbf{X}'$ by extending the mixture representation (6.29) to give

Section 13.2

$$k(\mathbf{X}, \mathbf{X}') = \sum_{\mathbf{Z}} p(\mathbf{X}|\mathbf{Z})p(\mathbf{X}'|\mathbf{Z})p(\mathbf{Z}) \tag{6.31}$$

so that both observed sequences are generated by the same hidden sequence $\mathbf{Z}$. This model can easily be extended to allow sequences of differing length to be compared.

An alternative technique for using generative models to define kernel functions is known as the *Fisher kernel* (Jaakkola and Haussler, 1999). Consider a parametric generative model $p(\mathbf{x}|\boldsymbol{\theta})$ where $\boldsymbol{\theta}$ denotes the vector of parameters. The goal is to find a kernel that measures the similarity of two input vectors $\mathbf{x}$ and $\mathbf{x}'$ induced by the generative model. Jaakkola and Haussler (1999) consider the gradient with respect to $\boldsymbol{\theta}$, which defines a vector in a 'feature' space having the same dimensionality as $\boldsymbol{\theta}$. In particular, they consider the *Fisher score*

$$\mathbf{g}(\boldsymbol{\theta}, \mathbf{x}) = \nabla_{\boldsymbol{\theta}} \ln p(\mathbf{x}|\boldsymbol{\theta}) \tag{6.32}$$

from which the Fisher kernel is defined by

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{g}(\boldsymbol{\theta}, \mathbf{x})^{\mathrm{T}}\mathbf{F}^{-1}\mathbf{g}(\boldsymbol{\theta}, \mathbf{x}'). \tag{6.33}$$

Here $\mathbf{F}$ is the *Fisher information matrix*, given by

$$\mathbf{F} = \mathbb{E}_{\mathbf{x}}\left[\mathbf{g}(\boldsymbol{\theta}, \mathbf{x})\mathbf{g}(\boldsymbol{\theta}, \mathbf{x})^{\mathrm{T}}\right] \tag{6.34}$$

where the expectation is with respect to $\mathbf{x}$ under the distribution $p(\mathbf{x}|\boldsymbol{\theta})$. This can be motivated from the perspective of *information geometry* (Amari, 1998), which considers the differential geometry of the space of model parameters. Here we simply note that the presence of the Fisher information matrix causes this kernel to be invariant under a nonlinear re-parameterization of the density model $\boldsymbol{\theta} \to \boldsymbol{\psi}(\boldsymbol{\theta})$.

Exercise 6.13

In practice, it is often infeasible to evaluate the Fisher information matrix. One approach is simply to replace the expectation in the definition of the Fisher information with the sample average, giving

$$\mathbf{F} \simeq \frac{1}{N}\sum_{n=1}^{N} \mathbf{g}(\boldsymbol{\theta}, \mathbf{x}_n)\mathbf{g}(\boldsymbol{\theta}, \mathbf{x}_n)^{\mathrm{T}}. \tag{6.35}$$

This is the covariance matrix of the Fisher scores, and so the Fisher kernel corresponds to a whitening of these scores. More simply, we can just omit the Fisher information matrix altogether and use the noninvariant kernel

Section 12.1.3

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{g}(\boldsymbol{\theta}, \mathbf{x})^{\mathrm{T}}\mathbf{g}(\boldsymbol{\theta}, \mathbf{x}'). \tag{6.36}$$

An application of Fisher kernels to document retrieval is given by Hofmann (2000).

A final example of a kernel function is the sigmoidal kernel given by

$$k(\mathbf{x}, \mathbf{x}') = \tanh\left(a\mathbf{x}^{\mathrm{T}}\mathbf{x}' + b\right) \tag{6.37}$$

whose Gram matrix in general is not positive semidefinite. This form of kernel has, however, been used in practice (Vapnik, 1995), possibly because it gives kernel expansions such as the support vector machine a superficial resemblance to neural network models. As we shall see, in the limit of an infinite number of basis functions, a Bayesian neural network with an appropriate prior reduces to a Gaussian process, thereby providing a deeper link between neural networks and kernel methods.

Section 6.4.7

6.3. Radial Basis Function Networks

In Chapter 3, we discussed regression models based on linear combinations of fixed basis functions, although we did not discuss in detail what form those basis functions might take. One choice that has been widely used is that of *radial basis functions*, which have the property that each basis function depends only on the radial distance (typically Euclidean) from a centre $\boldsymbol{\mu}_j$, so that $\phi_j(\mathbf{x}) = h(\|\mathbf{x} - \boldsymbol{\mu}_j\|)$.

Historically, radial basis functions were introduced for the purpose of exact function interpolation (Powell, 1987). Given a set of input vectors $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ along with corresponding target values $\{t_1, \ldots, t_N\}$, the goal is to find a smooth function $f(\mathbf{x})$ that fits every target value exactly, so that $f(\mathbf{x}_n) = t_n$ for $n = 1, \ldots, N$. This is achieved by expressing $f(\mathbf{x})$ as a linear combination of radial basis functions, one centred on every data point

$$f(\mathbf{x}) = \sum_{n=1}^{N} w_n h(\|\mathbf{x} - \mathbf{x}_n\|). \tag{6.38}$$

The values of the coefficients $\{w_n\}$ are found by least squares, and because there are the same number of coefficients as there are constraints, the result is a function that fits every target value exactly. In pattern recognition applications, however, the target values are generally noisy, and exact interpolation is undesirable because this corresponds to an over-fitted solution.

Expansions in radial basis functions also arise from regularization theory (Poggio and Girosi, 1990; Bishop, 1995a). For a sum-of-squares error function with a regularizer defined in terms of a differential operator, the optimal solution is given by an expansion in the *Green's functions* of the operator (which are analogous to the eigenvectors of a discrete matrix), again with one basis function centred on each data point. If the differential operator is isotropic then the Green's functions depend only on the radial distance from the corresponding data point. Due to the presence of the regularizer, the solution no longer interpolates the training data exactly.

Another motivation for radial basis functions comes from a consideration of the interpolation problem when the input (rather than the target) variables are noisy (Webb, 1994; Bishop, 1995a). If the noise on the input variable $\mathbf{x}$ is described by a variable $\boldsymbol{\xi}$ having a distribution $\nu(\boldsymbol{\xi})$, then the sum-of-squares error function becomes

$$E = \frac{1}{2}\sum_{n=1}^{N}\int \left\{y(\mathbf{x}_n + \boldsymbol{\xi}) - t_n\right\}^2 \nu(\boldsymbol{\xi})\,\mathrm{d}\boldsymbol{\xi}. \tag{6.39}$$

Appendix D
Exercise 6.17

Using the calculus of variations, we can optimize with respect to the function $f(\mathbf{x})$ to give

$$y(\mathbf{x}_n) = \sum_{n=1}^{N} t_n h(\mathbf{x} - \mathbf{x}_n) \tag{6.40}$$

where the basis functions are given by

$$h(\mathbf{x} - \mathbf{x}_n) = \frac{\nu(\mathbf{x} - \mathbf{x}_n)}{\displaystyle\sum_{n=1}^{N} \nu(\mathbf{x} - \mathbf{x}_n)}. \tag{6.41}$$

We see that there is one basis function centred on every data point. This is known as the *Nadaraya-Watson* model and will be derived again from a different perspective in Section 6.3.1. If the noise distribution $\nu(\boldsymbol{\xi})$ is isotropic, so that it is a function only of $\|\boldsymbol{\xi}\|$, then the basis functions will be radial.

Note that the basis functions (6.41) are normalized, so that $\sum_n h(\mathbf{x} - \mathbf{x}_n) = 1$ for any value of $\mathbf{x}$. The effect of such normalization is shown in Figure 6.2. Normalization is sometimes used in practice as it avoids having regions of input space where all of the basis functions take small values, which would necessarily lead to predictions in such regions that are either small or controlled purely by the bias parameter.

Another situation in which expansions in normalized radial basis functions arise is in the application of kernel density estimation to the problem of regression, as we shall discuss in Section 6.3.1.

Because there is one basis function associated with every data point, the corresponding model can be computationally costly to evaluate when making predictions for new data points. Models have therefore been proposed (Broomhead and Lowe, 1988; Moody and Darken, 1989; Poggio and Girosi, 1990), which retain the expansion in radial basis functions but where the number M of basis functions is smaller than the number N of data points. Typically, the number of basis functions, and the locations $\boldsymbol{\mu}_i$ of their centres, are determined based on the input data $\{\mathbf{x}_n\}$ alone. The basis functions are then kept fixed and the coefficients $\{w_i\}$ are determined by least squares by solving the usual set of linear equations, as discussed in Section 3.1.1.

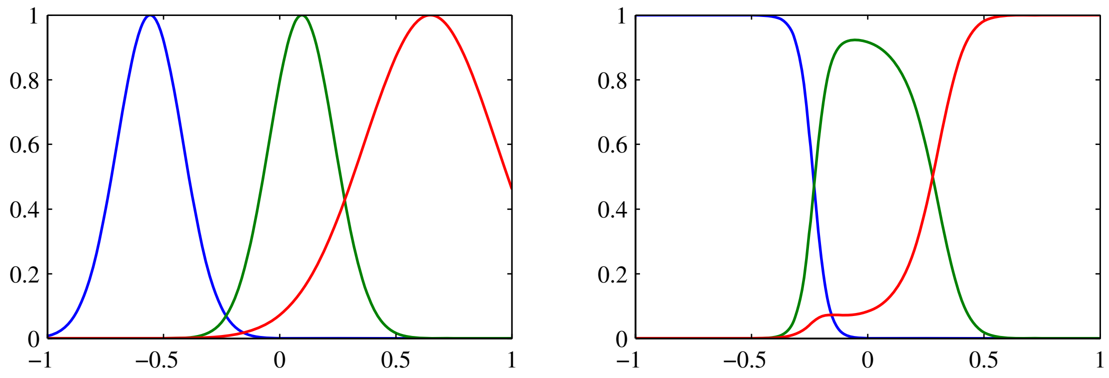

Figure 6.2 Plot of a set of Gaussian basis functions on the left, together with the corresponding normalized basis functions on the right.

One of the simplest ways of choosing basis function centres is to use a randomly chosen subset of the data points. A more systematic approach is called *orthogonal least squares* (Chen *et al.*, 1991). This is a sequential selection process in which at each step the next data point to be chosen as a basis function centre corresponds to the one that gives the greatest reduction in the sum-of-squares error. Values for the expansion coefficients are determined as part of the algorithm. Clustering algorithms such as K-means have also been used, which give a set of basis function centres that no longer coincide with training data points.

Section 9.1

6.3.1 Nadaraya-Watson model

In Section 3.3.3, we saw that the prediction of a linear regression model for a new input $\mathbf{x}$ takes the form of a linear combination of the training set target values with coefficients given by the 'equivalent kernel' (3.62) where the equivalent kernel satisfies the summation constraint (3.64).

We can motivate the kernel regression model (3.61) from a different perspective, starting with kernel density estimation. Suppose we have a training set $\{\mathbf{x}_n, t_n\}$ and we use a Parzen density estimator to model the joint distribution $p(\mathbf{x}, t)$, so that

Section 2.5.1

$$p(\mathbf{x}, t) = \frac{1}{N} \sum_{n=1}^{N} f(\mathbf{x} - \mathbf{x}_n, t - t_n) \tag{6.42}$$

where $f(\mathbf{x}, t)$ is the component density function, and there is one such component centred on each data point. We now find an expression for the regression function $y(\mathbf{x})$, corresponding to the conditional average of the target variable conditioned on the input variable, which is given by

$$\begin{aligned} y(\mathbf{x}) &= \mathbb{E}[t|\mathbf{x}] = \int_{-\infty}^{\infty} tp(t|\mathbf{x})\,\mathrm{d}t \\ &= \frac{\displaystyle\int tp(\mathbf{x},t)\,\mathrm{d}t}{\displaystyle\int p(\mathbf{x},t)\,\mathrm{d}t} \\ &= \frac{\displaystyle\sum_n \int tf(\mathbf{x}-\mathbf{x}_n, t-t_n)\,\mathrm{d}t}{\displaystyle\sum_m \int f(\mathbf{x}-\mathbf{x}_m, t-t_m)\,\mathrm{d}t}. \end{aligned} \tag{6.43}$$

We now assume for simplicity that the component density functions have zero mean so that

$$\int_{-\infty}^{\infty} f(\mathbf{x},t)t\,\mathrm{d}t = 0 \tag{6.44}$$

for all values of $\mathbf{x}$. Using a simple change of variable, we then obtain

$$\begin{aligned} y(\mathbf{x}) &= \frac{\displaystyle\sum_n g(\mathbf{x}-\mathbf{x}_n)t_n}{\displaystyle\sum_m g(\mathbf{x}-\mathbf{x}_m)} \\ &= \sum_n k(\mathbf{x},\mathbf{x}_n)t_n \end{aligned} \tag{6.45}$$

where $n, m = 1, \ldots, N$ and the kernel function $k(\mathbf{x}, \mathbf{x}_n)$ is given by

$$k(\mathbf{x},\mathbf{x}_n) = \frac{g(\mathbf{x}-\mathbf{x}_n)}{\displaystyle\sum_m g(\mathbf{x}-\mathbf{x}_m)} \tag{6.46}$$

and we have defined

$$g(\mathbf{x}) = \int_{-\infty}^{\infty} f(\mathbf{x},t)\,\mathrm{d}t. \tag{6.47}$$

The result (6.45) is known as the *Nadaraya-Watson* model, or *kernel regression* (Nadaraya, 1964; Watson, 1964). For a localized kernel function, it has the property of giving more weight to the data points $\mathbf{x}_n$ that are close to $\mathbf{x}$. Note that the kernel (6.46) satisfies the summation constraint

$$\sum_{n=1}^{N} k(\mathbf{x},\mathbf{x}_n) = 1.$$

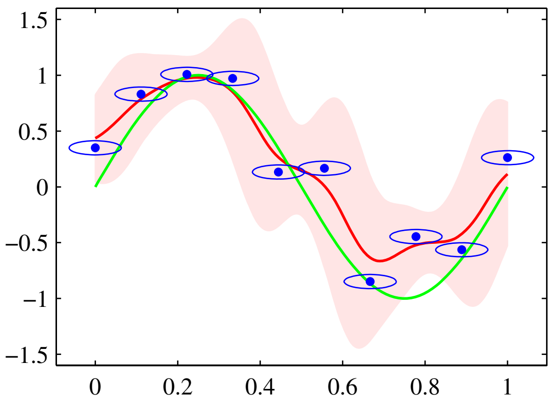

Figure 6.3 Illustration of the Nadaraya-Watson kernel regression model using isotropic Gaussian kernels, for the sinusoidal data set. The original sine function is shown by the green curve, the data points are shown in blue, and each is the centre of an isotropic Gaussian kernel. The resulting regression function, given by the conditional mean, is shown by the red line, along with the two-standard-deviation region for the conditional distribution $p(t|x)$ shown by the red shading. The blue ellipse around each data point shows one standard deviation contour for the corresponding kernel. These appear noncircular due to the different scales on the horizontal and vertical axes.

In fact, this model defines not only a conditional expectation but also a full conditional distribution given by

$$p(t|\mathbf{x}) = \frac{p(t, \mathbf{x})}{\int p(t, \mathbf{x})\,\mathrm{d}t} = \frac{\sum_n f(\mathbf{x} - \mathbf{x}_n, t - t_n)}{\sum_m \int f(\mathbf{x} - \mathbf{x}_m, t - t_m)\,\mathrm{d}t} \tag{6.48}$$

from which other expectations can be evaluated.

As an illustration we consider the case of a single input variable x in which $f(x, t)$ is given by a zero-mean isotropic Gaussian over the variable $\mathbf{z} = (x, t)$ with variance σ^2. The corresponding conditional distribution (6.48) is given by a Gaussian mixture, and is shown, together with the conditional mean, for the sinusoidal synthetic data set in Figure 6.3.

Exercise 6.18

An obvious extension of this model is to allow for more flexible forms of Gaussian components, for instance having different variance parameters for the input and target variables. More generally, we could model the joint distribution $p(t, \mathbf{x})$ using a Gaussian mixture model, trained using techniques discussed in Chapter 9 (Ghahramani and Jordan, 1994), and then find the corresponding conditional distribution $p(t|\mathbf{x})$. In this latter case we no longer have a representation in terms of kernel functions evaluated at the training set data points. However, the number of components in the mixture model can be smaller than the number of training set points, resulting in a model that is faster to evaluate for test data points. We have thereby accepted an increased computational cost during the training phase in order to have a model that is faster at making predictions.

6.4. Gaussian Processes

In Section 6.1, we introduced kernels by applying the concept of duality to a non-probabilistic model for regression. Here we extend the role of kernels to probabilistic discriminative models, leading to the framework of Gaussian processes. We shall thereby see how kernels arise naturally in a Bayesian setting.

In Chapter 3, we considered linear regression models of the form $y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})$ in which $\mathbf{w}$ is a vector of parameters and $\boldsymbol{\phi}(\mathbf{x})$ is a vector of fixed nonlinear basis functions that depend on the input vector $\mathbf{x}$. We showed that a prior distribution over $\mathbf{w}$ induced a corresponding prior distribution over functions $y(\mathbf{x}, \mathbf{w})$. Given a training data set, we then evaluated the posterior distribution over $\mathbf{w}$ and thereby obtained the corresponding posterior distribution over regression functions, which in turn (with the addition of noise) implies a predictive distribution $p(t|\mathbf{x})$ for new input vectors $\mathbf{x}$.

In the Gaussian process viewpoint, we dispense with the parametric model and instead define a prior probability distribution over functions directly. At first sight, it might seem difficult to work with a distribution over the uncountably infinite space of functions. However, as we shall see, for a finite training set we only need to consider the values of the function at the discrete set of input values $\mathbf{x}_n$ corresponding to the training set and test set data points, and so in practice we can work in a finite space.

Models equivalent to Gaussian processes have been widely studied in many different fields. For instance, in the geostatistics literature Gaussian process regression is known as *kriging* (Cressie, 1993). Similarly, ARMA (autoregressive moving average) models, Kalman filters, and radial basis function networks can all be viewed as forms of Gaussian process models. Reviews of Gaussian processes from a machine learning perspective can be found in MacKay (1998), Williams (1999), and MacKay (2003), and a comparison of Gaussian process models with alternative approaches is given in Rasmussen (1996). See also Rasmussen and Williams (2006) for a recent textbook on Gaussian processes.

6.4.1 Linear regression revisited

In order to motivate the Gaussian process viewpoint, let us return to the linear regression example and re-derive the predictive distribution by working in terms of distributions over functions $y(\mathbf{x}, \mathbf{w})$. This will provide a specific example of a Gaussian process.

Consider a model defined in terms of a linear combination of M fixed basis functions given by the elements of the vector $\boldsymbol{\phi}(\mathbf{x})$ so that

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}) \tag{6.49}$$

where $\mathbf{x}$ is the input vector and $\mathbf{w}$ is the M-dimensional weight vector. Now consider a prior distribution over $\mathbf{w}$ given by an isotropic Gaussian of the form

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I}) \tag{6.50}$$

governed by the hyperparameter α, which represents the precision (inverse variance) of the distribution. For any given value of $\mathbf{w}$, the definition (6.49) defines a particular function of $\mathbf{x}$. The probability distribution over $\mathbf{w}$ defined by (6.50) therefore induces a probability distribution over functions $y(\mathbf{x})$. In practice, we wish to evaluate this function at specific values of $\mathbf{x}$, for example at the training data points

$\mathbf{x}_1, \ldots, \mathbf{x}_N$. We are therefore interested in the joint distribution of the function values $y(\mathbf{x}_1), \ldots, y(\mathbf{x}_N)$, which we denote by the vector $\mathbf{y}$ with elements $y_n = y(\mathbf{x}_n)$ for $n = 1, \ldots, N$. From (6.49), this vector is given by

$$\mathbf{y} = \mathbf{\Phi}\mathbf{w} \tag{6.51}$$

where $\mathbf{\Phi}$ is the design matrix with elements $\Phi_{nk} = \phi_k(\mathbf{x}_n)$. We can find the probability distribution of $\mathbf{y}$ as follows. First of all we note that $\mathbf{y}$ is a linear combination of Gaussian distributed variables given by the elements of $\mathbf{w}$ and hence is itself Gaussian. We therefore need only to find its mean and covariance, which are given from (6.50) by

Exercise 2.31

$$\mathbb{E}[\mathbf{y}] = \mathbf{\Phi}\mathbb{E}[\mathbf{w}] = \mathbf{0} \tag{6.52}$$

$$\operatorname{cov}[\mathbf{y}] = \mathbb{E}\left[\mathbf{y}\mathbf{y}^{\mathrm{T}}\right] = \mathbf{\Phi}\mathbb{E}\left[\mathbf{w}\mathbf{w}^{\mathrm{T}}\right]\mathbf{\Phi}^{\mathrm{T}} = \frac{1}{\alpha}\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}} = \mathbf{K} \tag{6.53}$$

where $\mathbf{K}$ is the Gram matrix with elements

$$K_{nm} = k(\mathbf{x}_n, \mathbf{x}_m) = \frac{1}{\alpha}\boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_m) \tag{6.54}$$

and $k(\mathbf{x}, \mathbf{x}')$ is the kernel function.

This model provides us with a particular example of a Gaussian process. In general, a Gaussian process is defined as a probability distribution over functions $y(\mathbf{x})$ such that the set of values of $y(\mathbf{x})$ evaluated at an arbitrary set of points $\mathbf{x}_1, \ldots, \mathbf{x}_N$ jointly have a Gaussian distribution. In cases where the input vector $\mathbf{x}$ is two dimensional, this may also be known as a *Gaussian random field*. More generally, a *stochastic process* $y(\mathbf{x})$ is specified by giving the joint probability distribution for any finite set of values $y(\mathbf{x}_1), \ldots, y(\mathbf{x}_N)$ in a consistent manner.

A key point about Gaussian stochastic processes is that the joint distribution over N variables $y_1, \ldots, y_N$ is specified completely by the second-order statistics, namely the mean and the covariance. In most applications, we will not have any prior knowledge about the mean of $y(\mathbf{x})$ and so by symmetry we take it to be zero. This is equivalent to choosing the mean of the prior over weight values $p(\mathbf{w}|\alpha)$ to be zero in the basis function viewpoint. The specification of the Gaussian process is then completed by giving the covariance of $y(\mathbf{x})$ evaluated at any two values of $\mathbf{x}$, which is given by the kernel function

$$\mathbb{E}\left[y(\mathbf{x}_n)y(\mathbf{x}_m)\right] = k(\mathbf{x}_n, \mathbf{x}_m). \tag{6.55}$$

For the specific case of a Gaussian process defined by the linear regression model (6.49) with a weight prior (6.50), the kernel function is given by (6.54).

We can also define the kernel function directly, rather than indirectly through a choice of basis function. Figure 6.4 shows samples of functions drawn from Gaussian processes for two different choices of kernel function. The first of these is a 'Gaussian' kernel of the form (6.23), and the second is the exponential kernel given by

$$k(x, x') = \exp\left(-\theta\,|x - x'|\right) \tag{6.56}$$

which corresponds to the *Ornstein-Uhlenbeck process* originally introduced by Uhlenbeck and Ornstein (1930) to describe Brownian motion.

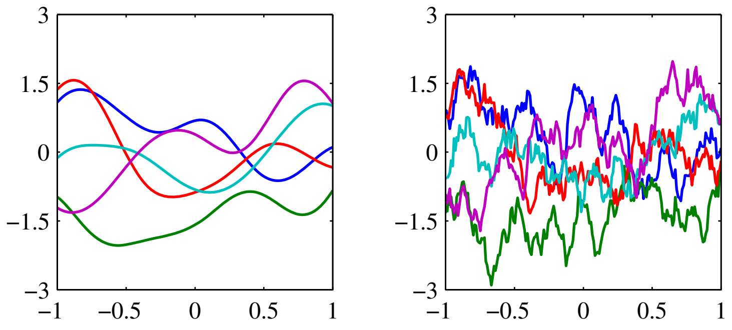

Figure 6.4 Samples from Gaussian processes for a 'Gaussian' kernel (left) and an exponential kernel (right).

6.4.2 Gaussian processes for regression

In order to apply Gaussian process models to the problem of regression, we need to take account of the noise on the observed target values, which are given by

$$t_n = y_n + \epsilon_n \tag{6.57}$$

where $y_n = y(\mathbf{x}_n)$, and ϵ_n is a random noise variable whose value is chosen independently for each observation n. Here we shall consider noise processes that have a Gaussian distribution, so that

$$p(t_n|y_n) = \mathcal{N}(t_n|y_n, \beta^{-1}) \tag{6.58}$$

where β is a hyperparameter representing the precision of the noise. Because the noise is independent for each data point, the joint distribution of the target values $\mathbf{t} = (t_1, \ldots, t_N)^{\mathrm{T}}$ conditioned on the values of $\mathbf{y} = (y_1, \ldots, y_N)^{\mathrm{T}}$ is given by an isotropic Gaussian of the form

$$p(\mathbf{t}|\mathbf{y}) = \mathcal{N}(\mathbf{t}|\mathbf{y}, \beta^{-1}\mathbf{I}_N) \tag{6.59}$$

where $\mathbf{I}_N$ denotes the $N \times N$ unit matrix. From the definition of a Gaussian process, the marginal distribution $p(\mathbf{y})$ is given by a Gaussian whose mean is zero and whose covariance is defined by a Gram matrix $\mathbf{K}$ so that

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K}). \tag{6.60}$$

The kernel function that determines $\mathbf{K}$ is typically chosen to express the property that, for points $\mathbf{x}_n$ and $\mathbf{x}_m$ that are similar, the corresponding values $y(\mathbf{x}_n)$ and $y(\mathbf{x}_m)$ will be more strongly correlated than for dissimilar points. Here the notion of similarity will depend on the application.

In order to find the marginal distribution $p(\mathbf{t})$, conditioned on the input values $\mathbf{x}_1, \ldots, \mathbf{x}_N$, we need to integrate over $\mathbf{y}$. This can be done by making use of the results from Section 2.3.3 for the linear-Gaussian model. Using (2.115), we see that the marginal distribution of $\mathbf{t}$ is given by

$$p(\mathbf{t}) = \int p(\mathbf{t}|\mathbf{y})p(\mathbf{y})\,\mathrm{d}\mathbf{y} = \mathcal{N}(\mathbf{t}|\mathbf{0}, \mathbf{C}) \tag{6.61}$$

where the covariance matrix $\mathbf{C}$ has elements

$$C(\mathbf{x}_n, \mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m) + \beta^{-1}\delta_{nm}. \tag{6.62}$$

This result reflects the fact that the two Gaussian sources of randomness, namely that associated with $y(\mathbf{x})$ and that associated with ϵ, are independent and so their covariances simply add.

One widely used kernel function for Gaussian process regression is given by the exponential of a quadratic form, with the addition of constant and linear terms to give

$$k(\mathbf{x}_n, \mathbf{x}_m) = \theta_0 \exp\left\{-\frac{\theta_1}{2}\|\mathbf{x}_n - \mathbf{x}_m\|^2\right\} + \theta_2 + \theta_3 \mathbf{x}_n^{\mathrm{T}}\mathbf{x}_m. \tag{6.63}$$

Note that the term involving θ_3 corresponds to a parametric model that is a linear function of the input variables. Samples from this prior are plotted for various values of the parameters $\theta_0, \ldots, \theta_3$ in Figure 6.5, and Figure 6.6 shows a set of points sampled from the joint distribution (6.60) along with the corresponding values defined by (6.61).

So far, we have used the Gaussian process viewpoint to build a model of the joint distribution over sets of data points. Our goal in regression, however, is to make predictions of the target variables for new inputs, given a set of training data. Let us suppose that $\mathbf{t}_N = (t_1, \ldots, t_N)^{\mathrm{T}}$, corresponding to input values $\mathbf{x}_1, \ldots, \mathbf{x}_N$, comprise the observed training set, and our goal is to predict the target variable t_{N+1} for a new input vector $\mathbf{x}_{N+1}$. This requires that we evaluate the predictive distribution $p(t_{N+1}|\mathbf{t}_N)$. Note that this distribution is conditioned also on the variables $\mathbf{x}_1, \ldots, \mathbf{x}_N$ and $\mathbf{x}_{N+1}$. However, to keep the notation simple we will not show these conditioning variables explicitly.

To find the conditional distribution $p(t_{N+1}|\mathbf{t})$, we begin by writing down the joint distribution $p(\mathbf{t}_{N+1})$, where $\mathbf{t}_{N+1}$ denotes the vector $(t_1, \ldots, t_N, t_{N+1})^{\mathrm{T}}$. We then apply the results from Section 2.3.1 to obtain the required conditional distribution, as illustrated in Figure 6.7.

From (6.61), the joint distribution over $t_1, \ldots, t_{N+1}$ will be given by

$$p(\mathbf{t}_{N+1}) = \mathcal{N}(\mathbf{t}_{N+1}|\mathbf{0}, \mathbf{C}_{N+1}) \tag{6.64}$$

where $\mathbf{C}_{N+1}$ is an $(N+1) \times (N+1)$ covariance matrix with elements given by (6.62). Because this joint distribution is Gaussian, we can apply the results from Section 2.3.1 to find the conditional Gaussian distribution. To do this, we partition the covariance matrix as follows

$$\mathbf{C}_{N+1} = \begin{pmatrix} \mathbf{C}_N & \mathbf{k} \\ \mathbf{k}^{\mathrm{T}} & c \end{pmatrix} \tag{6.65}$$

where $\mathbf{C}_N$ is the $N \times N$ covariance matrix with elements given by (6.62) for $n, m = 1, \ldots, N$, the vector $\mathbf{k}$ has elements $k(\mathbf{x}_n, \mathbf{x}_{N+1})$ for $n = 1, \ldots, N$, and the scalar

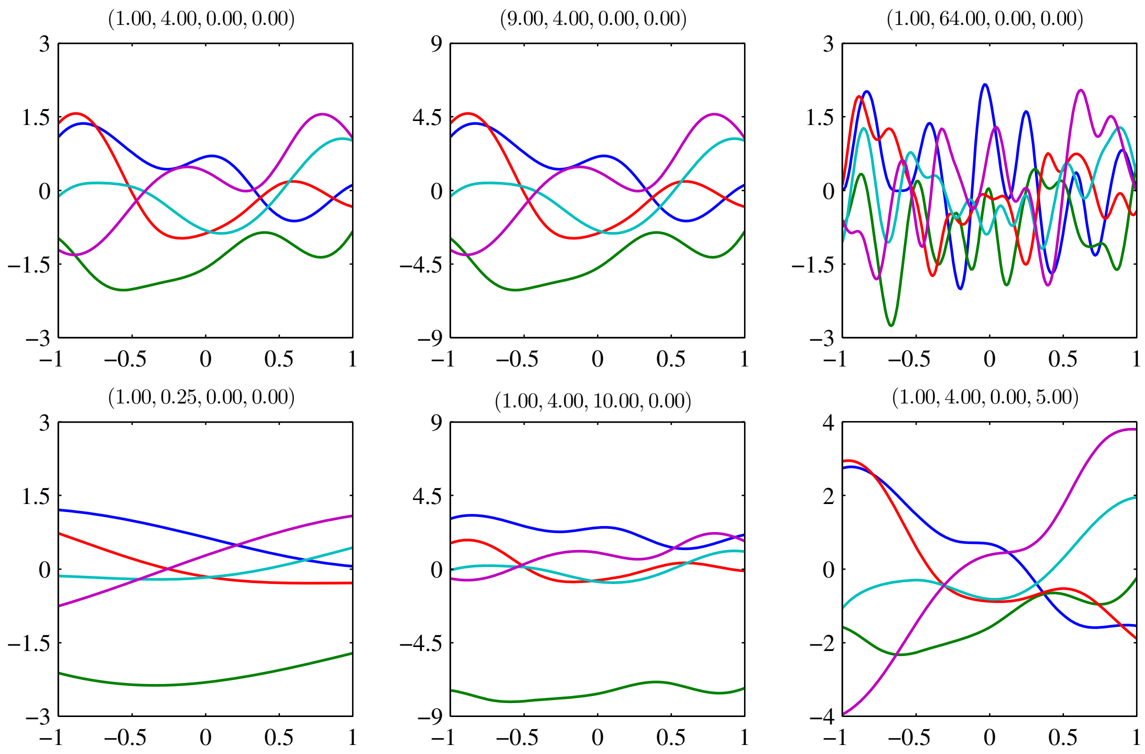

Figure 6.5 Samples from a Gaussian process prior defined by the covariance function (6.63). The title above each plot denotes $(\theta_0, \theta_1, \theta_2, \theta_3)$.

$c = k(\mathbf{x}_{N+1}, \mathbf{x}_{N+1}) + \beta^{-1}$. Using the results (2.81) and (2.82), we see that the conditional distribution $p(t_{N+1}|\mathbf{t})$ is a Gaussian distribution with mean and covariance given by

$$m(\mathbf{x}_{N+1}) = \mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{t} \tag{6.66}$$

$$\sigma^2(\mathbf{x}_{N+1}) = c - \mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{k}. \tag{6.67}$$

These are the key results that define Gaussian process regression. Because the vector $\mathbf{k}$ is a function of the test point input value $\mathbf{x}_{N+1}$, we see that the predictive distribution is a Gaussian whose mean and variance both depend on $\mathbf{x}_{N+1}$. An example of Gaussian process regression is shown in Figure 6.8.

The only restriction on the kernel function is that the covariance matrix given by (6.62) must be positive definite. If λ_i is an eigenvalue of $\mathbf{K}$, then the corresponding eigenvalue of $\mathbf{C}$ will be $\lambda_i + \beta^{-1}$. It is therefore sufficient that the kernel matrix $k(\mathbf{x}_n, \mathbf{x}_m)$ be positive semidefinite for any pair of points $\mathbf{x}_n$ and $\mathbf{x}_m$, so that $\lambda_i \geqslant 0$, because any eigenvalue λ_i that is zero will still give rise to a positive eigenvalue for $\mathbf{C}$ because $\beta > 0$. This is the same restriction on the kernel function discussed earlier, and so we can again exploit all of the techniques in Section 6.2 to construct

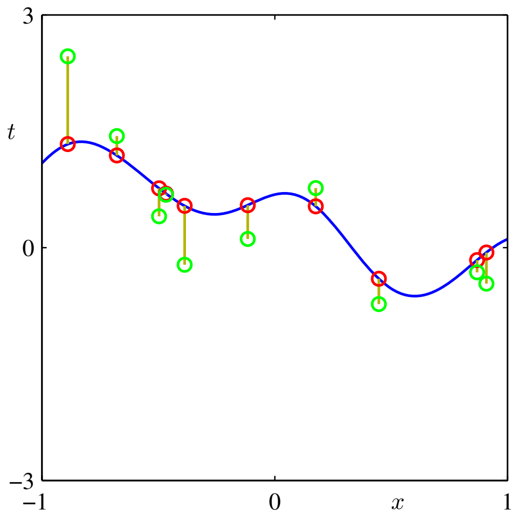

Figure 6.6 Illustration of the sampling of data points $\{t_n\}$ from a Gaussian process. The blue curve shows a sample function from the Gaussian process prior over functions, and the red points show the values of y_n obtained by evaluating the function at a set of input values $\{x_n\}$. The corresponding values of $\{t_n\}$, shown in green, are obtained by adding independent Gaussian noise to each of the $\{y_n\}$.

suitable kernels.

Note that the mean (6.66) of the predictive distribution can be written, as a function of $\mathbf{x}_{N+1}$, in the form

$$m(\mathbf{x}_{N+1}) = \sum_{n=1}^{N} a_n k(\mathbf{x}_n, \mathbf{x}_{N+1}) \tag{6.68}$$

where a_n is the n^{th} component of $\mathbf{C}_N^{-1}\mathbf{t}$. Thus, if the kernel function $k(\mathbf{x}_n, \mathbf{x}_m)$ depends only on the distance $\|\mathbf{x}_n - \mathbf{x}_m\|$, then we obtain an expansion in radial basis functions.

The results (6.66) and (6.67) define the predictive distribution for Gaussian process regression with an arbitrary kernel function $k(\mathbf{x}_n, \mathbf{x}_m)$. In the particular case in which the kernel function $k(\mathbf{x}, \mathbf{x}')$ is defined in terms of a finite set of basis functions, we can derive the results obtained previously in Section 3.3.2 for linear regression starting from the Gaussian process viewpoint.

Exercise 6.21

For such models, we can therefore obtain the predictive distribution either by taking a parameter space viewpoint and using the linear regression result or by taking a function space viewpoint and using the Gaussian process result.

The central computational operation in using Gaussian processes will involve the inversion of a matrix of size $N \times N$, for which standard methods require $O(N^3)$ computations. By contrast, in the basis function model we have to invert a matrix $\mathbf{S}_N$ of size $M \times M$, which has $O(M^3)$ computational complexity. Note that for both viewpoints, the matrix inversion must be performed once for the given training set. For each new test point, both methods require a vector-matrix multiply, which has cost $O(N^2)$ in the Gaussian process case and $O(M^2)$ for the linear basis function model. If the number M of basis functions is smaller than the number N of data points, it will be computationally more efficient to work in the basis function

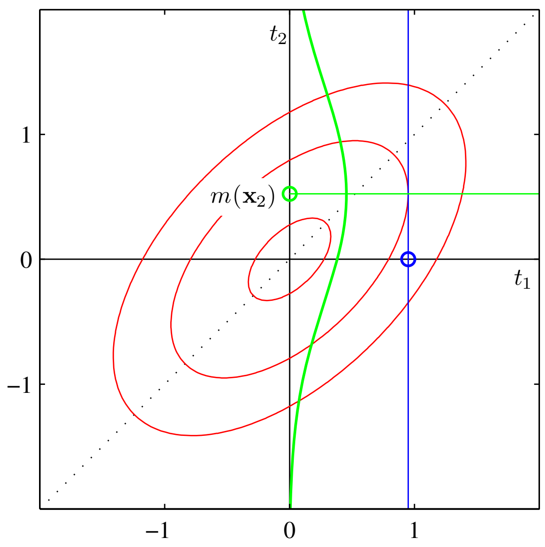

Figure 6.7 Illustration of the mechanism of Gaussian process regression for the case of one training point and one test point, in which the red ellipses show contours of the joint distribution $p(t_1, t_2)$. Here t_1 is the training data point, and conditioning on the value of t_1, corresponding to the vertical blue line, we obtain $p(t_2|t_1)$ shown as a function of t_2 by the green curve.

framework. However, an advantage of a Gaussian processes viewpoint is that we can consider covariance functions that can only be expressed in terms of an infinite number of basis functions.

For large training data sets, however, the direct application of Gaussian process methods can become infeasible, and so a range of approximation schemes have been developed that have better scaling with training set size than the exact approach (Gibbs, 1997; Tresp, 2001; Smola and Bartlett, 2001; Williams and Seeger, 2001; Csató and Opper, 2002; Seeger *et al.*, 2003). Practical issues in the application of Gaussian processes are discussed in Bishop and Nabney (2008).

We have introduced Gaussian process regression for the case of a single target variable. The extension of this formalism to multiple target variables, known as co-kriging (Cressie, 1993), is straightforward. Various other extensions of Gaus- *Exercise 6.23*

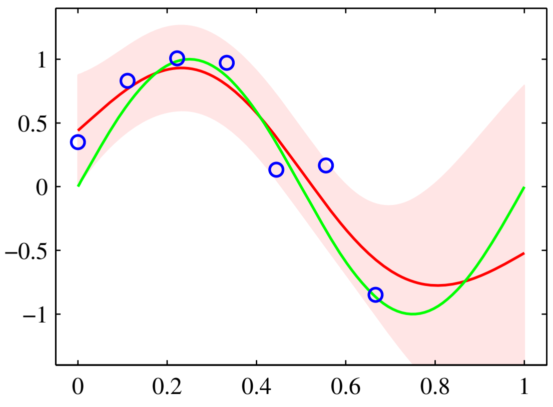

Figure 6.8 Illustration of Gaussian process regression applied to the sinusoidal data set in Figure A.6 in which the three right-most data points have been omitted. The green curve shows the sinusoidal function from which the data points, shown in blue, are obtained by sampling and addition of Gaussian noise. The red line shows the mean of the Gaussian process predictive distribution, and the shaded region corresponds to plus and minus two standard deviations. Notice how the uncertainty increases in the region to the right of the data points.

sian process regression have also been considered, for purposes such as modelling the distribution over low-dimensional manifolds for unsupervised learning (Bishop *et al.*, 1998a) and the solution of stochastic differential equations (Graepel, 2003).

6.4.3 Learning the hyperparameters

The predictions of a Gaussian process model will depend, in part, on the choice of covariance function. In practice, rather than fixing the covariance function, we may prefer to use a parametric family of functions and then infer the parameter values from the data. These parameters govern such things as the length scale of the correlations and the precision of the noise and correspond to the hyperparameters in a standard parametric model.

Techniques for learning the hyperparameters are based on the evaluation of the likelihood function $p(\mathbf{t}|\boldsymbol{\theta})$ where $\boldsymbol{\theta}$ denotes the hyperparameters of the Gaussian process model. The simplest approach is to make a point estimate of $\boldsymbol{\theta}$ by maximizing the log likelihood function. Because $\boldsymbol{\theta}$ represents a set of hyperparameters for the regression problem, this can be viewed as analogous to the type 2 maximum likelihood procedure for linear regression models. Maximization of the log likelihood can be done using efficient gradient-based optimization algorithms such as conjugate gradients (Fletcher, 1987; Nocedal and Wright, 1999; Bishop and Nabney, 2008).

Section 3.5

The log likelihood function for a Gaussian process regression model is easily evaluated using the standard form for a multivariate Gaussian distribution, giving

$$\ln p(\mathbf{t}|\boldsymbol{\theta}) = -\frac{1}{2}\ln|\mathbf{C}_N| - \frac{1}{2}\mathbf{t}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{t} - \frac{N}{2}\ln(2\pi). \tag{6.69}$$

For nonlinear optimization, we also need the gradient of the log likelihood function with respect to the parameter vector $\boldsymbol{\theta}$. We shall assume that evaluation of the derivatives of $\mathbf{C}_N$ is straightforward, as would be the case for the covariance functions considered in this chapter. Making use of the result (C.21) for the derivative of $\mathbf{C}_N^{-1}$, together with the result (C.22) for the derivative of $\ln|\mathbf{C}_N|$, we obtain

$$\frac{\partial}{\partial\theta_i}\ln p(\mathbf{t}|\boldsymbol{\theta}) = -\frac{1}{2}\mathrm{Tr}\left(\mathbf{C}_N^{-1}\frac{\partial\mathbf{C}_N}{\partial\theta_i}\right) + \frac{1}{2}\mathbf{t}^{\mathrm{T}}\mathbf{C}_N^{-1}\frac{\partial\mathbf{C}_N}{\partial\theta_i}\mathbf{C}_N^{-1}\mathbf{t}. \tag{6.70}$$

Because $\ln p(\mathbf{t}|\boldsymbol{\theta})$ will in general be a nonconvex function, it can have multiple maxima.

It is straightforward to introduce a prior over $\boldsymbol{\theta}$ and to maximize the log posterior using gradient-based methods. In a fully Bayesian treatment, we need to evaluate marginals over $\boldsymbol{\theta}$ weighted by the product of the prior $p(\boldsymbol{\theta})$ and the likelihood function $p(\mathbf{t}|\boldsymbol{\theta})$. In general, however, exact marginalization will be intractable, and we must resort to approximations.

The Gaussian process regression model gives a predictive distribution whose mean and variance are functions of the input vector $\mathbf{x}$. However, we have assumed that the contribution to the predictive variance arising from the additive noise, governed by the parameter β, is a constant. For some problems, known as *heteroscedastic*, the noise variance itself will also depend on $\mathbf{x}$. To model this, we can extend the

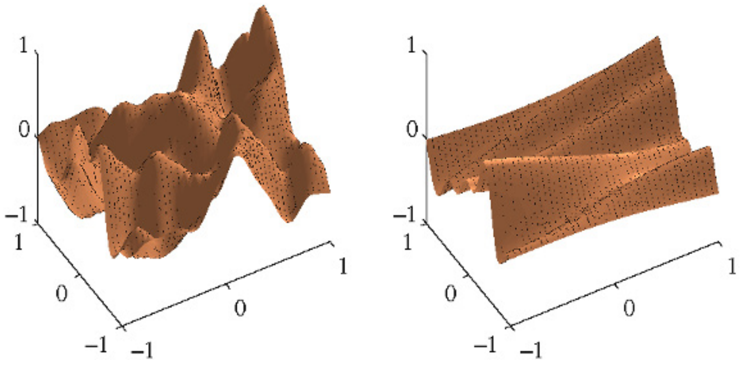

Figure 6.9 Samples from the ARD prior for Gaussian processes, in which the kernel function is given by (6.71). The left plot corresponds to $\eta_1 = \eta_2 = 1$, and the right plot corresponds to $\eta_1 = 1$, $\eta_2 = 0.01$.

Gaussian process framework by introducing a second Gaussian process to represent the dependence of β on the input $\mathbf{x}$ (Goldberg *et al.*, 1998). Because β is a variance, and hence nonnegative, we use the Gaussian process to model $\ln \beta(\mathbf{x})$.

6.4.4 Automatic relevance determination

In the previous section, we saw how maximum likelihood could be used to determine a value for the correlation length-scale parameter in a Gaussian process. This technique can usefully be extended by incorporating a separate parameter for each input variable (Rasmussen and Williams, 2006). The result, as we shall see, is that the optimization of these parameters by maximum likelihood allows the relative importance of different inputs to be inferred from the data. This represents an example in the Gaussian process context of *automatic relevance determination*, or *ARD*, which was originally formulated in the framework of neural networks (MacKay, 1994; Neal, 1996). The mechanism by which appropriate inputs are preferred is discussed in Section 7.2.2.

Consider a Gaussian process with a two-dimensional input space $\mathbf{x} = (x_1, x_2)$, having a kernel function of the form

$$k(\mathbf{x}, \mathbf{x}') = \theta_0 \exp\left\{ -\frac{1}{2} \sum_{i=1}^{2} \eta_i (x_i - x_i')^2 \right\}. \tag{6.71}$$

Samples from the resulting prior over functions $y(\mathbf{x})$ are shown for two different settings of the precision parameters η_i in Figure 6.9. We see that, as a particular parameter η_i becomes small, the function becomes relatively insensitive to the corresponding input variable x_i. By adapting these parameters to a data set using maximum likelihood, it becomes possible to detect input variables that have little effect on the predictive distribution, because the corresponding values of η_i will be small. This can be useful in practice because it allows such inputs to be discarded. ARD is illustrated using a simple synthetic data set having three inputs x_1, x_2 and x_3 (Nabney, 2002) in Figure 6.10. The target variable t, is generated by sampling 100 values of x_1 from a Gaussian, evaluating the function $\sin(2\pi x_1)$, and then adding

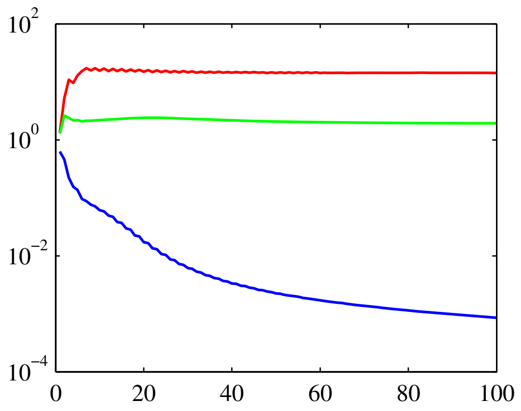

Figure 6.10 Illustration of automatic relevance determination in a Gaussian process for a synthetic problem having three inputs x_1, x_2, and x_3, for which the curves show the corresponding values of the hyperparameters η_1 (red), η_2 (green), and η_3 (blue) as a function of the number of iterations when optimizing the marginal likelihood. Details are given in the text. Note the logarithmic scale on the vertical axis.

Gaussian noise. Values of x_2 are given by copying the corresponding values of x_1 and adding noise, and values of x_3 are sampled from an independent Gaussian distribution. Thus x_1 is a good predictor of t, x_2 is a more noisy predictor of t, and x_3 has only chance correlations with t. The marginal likelihood for a Gaussian process with ARD parameters η_1, η_2, η_3 is optimized using the scaled conjugate gradients algorithm. We see from Figure 6.10 that η_1 converges to a relatively large value, η_2 converges to a much smaller value, and η_3 becomes very small indicating that x_3 is irrelevant for predicting t.

The ARD framework is easily incorporated into the exponential-quadratic kernel (6.63) to give the following form of kernel function, which has been found useful for applications of Gaussian processes to a range of regression problems

$$k(\mathbf{x}_n, \mathbf{x}_m) = \theta_0 \exp\left\{-\frac{1}{2}\sum_{i=1}^{D} \eta_i (x_{ni} - x_{mi})^2\right\} + \theta_2 + \theta_3 \sum_{i=1}^{D} x_{ni} x_{mi} \tag{6.72}$$

where D is the dimensionality of the input space.

6.4.5 Gaussian processes for classification

In a probabilistic approach to classification, our goal is to model the posterior probabilities of the target variable for a new input vector, given a set of training data. These probabilities must lie in the interval $(0, 1)$, whereas a Gaussian process model makes predictions that lie on the entire real axis. However, we can easily adapt Gaussian processes to classification problems by transforming the output of the Gaussian process using an appropriate nonlinear activation function.

Consider first the two-class problem with a target variable $t \in \{0, 1\}$. If we define a Gaussian process over a function $a(\mathbf{x})$ and then transform the function using a logistic sigmoid $y = \sigma(a)$, given by (4.59), then we will obtain a non-Gaussian stochastic process over functions $y(\mathbf{x})$ where $y \in (0, 1)$. This is illustrated for the case of a one-dimensional input space in Figure 6.11 in which the probability distri-

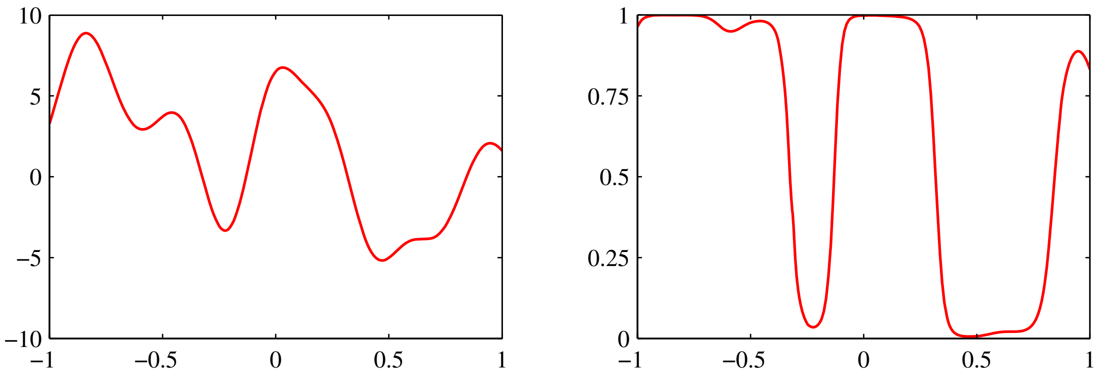

Figure 6.11 The left plot shows a sample from a Gaussian process prior over functions $a(\mathbf{x})$, and the right plot shows the result of transforming this sample using a logistic sigmoid function.

bution over the target variable t is then given by the Bernoulli distribution

$$p(t|a) = \sigma(a)^t (1 - \sigma(a))^{1-t}. \tag{6.73}$$

As usual, we denote the training set inputs by $\mathbf{x}_1, \ldots, \mathbf{x}_N$ with corresponding observed target variables $\mathbf{t} = (t_1, \ldots, t_N)^{\mathrm{T}}$. We also consider a single test point $\mathbf{x}_{N+1}$ with target value t_{N+1}. Our goal is to determine the predictive distribution $p(t_{N+1}|\mathbf{t})$, where we have left the conditioning on the input variables implicit. To do this we introduce a Gaussian process prior over the vector $\mathbf{a}_{N+1}$, which has components $a(\mathbf{x}_1), \ldots, a(\mathbf{x}_{N+1})$. This in turn defines a non-Gaussian process over $\mathbf{t}_{N+1}$, and by conditioning on the training data $\mathbf{t}_N$ we obtain the required predictive distribution. The Gaussian process prior for $\mathbf{a}_{N+1}$ takes the form

$$p(\mathbf{a}_{N+1}) = \mathcal{N}(\mathbf{a}_{N+1}|\mathbf{0}, \mathbf{C}_{N+1}). \tag{6.74}$$

Unlike the regression case, the covariance matrix no longer includes a noise term because we assume that all of the training data points are correctly labelled. However, for numerical reasons it is convenient to introduce a noise-like term governed by a parameter ν that ensures that the covariance matrix is positive definite. Thus the covariance matrix $\mathbf{C}_{N+1}$ has elements given by

$$C(\mathbf{x}_n, \mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m) + \nu\delta_{nm} \tag{6.75}$$

where $k(\mathbf{x}_n, \mathbf{x}_m)$ is any positive semidefinite kernel function of the kind considered in Section 6.2, and the value of ν is typically fixed in advance. We shall assume that the kernel function $k(\mathbf{x}, \mathbf{x}')$ is governed by a vector $\boldsymbol{\theta}$ of parameters, and we shall later discuss how $\boldsymbol{\theta}$ may be learned from the training data.

For two-class problems, it is sufficient to predict $p(t_{N+1} = 1|\mathbf{t}_N)$ because the value of $p(t_{N+1} = 0|\mathbf{t}_N)$ is then given by $1 - p(t_{N+1} = 1|\mathbf{t}_N)$. The required predictive distribution is given by

$$p(t_{N+1} = 1|\mathbf{t}_N) = \int p(t_{N+1} = 1|a_{N+1})p(a_{N+1}|\mathbf{t}_N)\,\mathrm{d}a_{N+1} \tag{6.76}$$

where $p(t_{N+1} = 1|a_{N+1}) = \sigma(a_{N+1})$.

This integral is analytically intractable, and so may be approximated using sampling methods (Neal, 1997). Alternatively, we can consider techniques based on an analytical approximation. In Section 4.5.2, we derived the approximate formula (4.153) for the convolution of a logistic sigmoid with a Gaussian distribution. We can use this result to evaluate the integral in (6.76) provided we have a Gaussian approximation to the posterior distribution $p(a_{N+1}|\mathbf{t}_N)$. The usual justification for a Gaussian approximation to a posterior distribution is that the true posterior will tend to a Gaussian as the number of data points increases as a consequence of the central limit theorem. In the case of Gaussian processes, the number of variables grows with the number of data points, and so this argument does not apply directly. However, if we consider increasing the number of data points falling in a fixed region of $\mathbf{x}$ space, then the corresponding uncertainty in the function $a(\mathbf{x})$ will decrease, again leading asymptotically to a Gaussian (Williams and Barber, 1998).

Section 2.3

Three different approaches to obtaining a Gaussian approximation have been considered. One technique is based on *variational inference* (Gibbs and MacKay, 2000) and makes use of the local variational bound (10.144) on the logistic sigmoid. This allows the product of sigmoid functions to be approximated by a product of Gaussians thereby allowing the marginalization over $\mathbf{a}_N$ to be performed analytically. The approach also yields a lower bound on the likelihood function $p(\mathbf{t}_N|\boldsymbol{\theta})$. The variational framework for Gaussian process classification can also be extended to multiclass ($K > 2$) problems by using a Gaussian approximation to the softmax function (Gibbs, 1997).

Section 10.1

A second approach uses *expectation propagation* (Opper and Winther, 2000b; Minka, 2001b; Seeger, 2003). Because the true posterior distribution is unimodal, as we shall see shortly, the expectation propagation approach can give good results.

Section 10.7

6.4.6 Laplace approximation

The third approach to Gaussian process classification is based on the Laplace approximation, which we now consider in detail. In order to evaluate the predictive distribution (6.76), we seek a Gaussian approximation to the posterior distribution over a_{N+1}, which, using Bayes' theorem, is given by

Section 4.4

$$\begin{aligned} p(a_{N+1}|\mathbf{t}_N) &= \int p(a_{N+1}, \mathbf{a}_N|\mathbf{t}_N)\,\mathrm{d}\mathbf{a}_N \\ &= \frac{1}{p(\mathbf{t}_N)}\int p(a_{N+1}, \mathbf{a}_N)p(\mathbf{t}_N|a_{N+1}, \mathbf{a}_N)\,\mathrm{d}\mathbf{a}_N \\ &= \frac{1}{p(\mathbf{t}_N)}\int p(a_{N+1}|\mathbf{a}_N)p(\mathbf{a}_N)p(\mathbf{t}_N|\mathbf{a}_N)\,\mathrm{d}\mathbf{a}_N \\ &= \int p(a_{N+1}|\mathbf{a}_N)p(\mathbf{a}_N|\mathbf{t}_N)\,\mathrm{d}\mathbf{a}_N \end{aligned} \tag{6.77}$$

where we have used $p(\mathbf{t}_N|a_{N+1}, \mathbf{a}_N) = p(\mathbf{t}_N|\mathbf{a}_N)$. The conditional distribution $p(a_{N+1}|\mathbf{a}_N)$ is obtained by invoking the results (6.66) and (6.67) for Gaussian process regression, to give

$$p(a_{N+1}|\mathbf{a}_N) = \mathcal{N}(a_{N+1}|\mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{a}_N, c - \mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{k}). \tag{6.78}$$

We can therefore evaluate the integral in (6.77) by finding a Laplace approximation for the posterior distribution $p(\mathbf{a}_N|\mathbf{t}_N)$, and then using the standard result for the convolution of two Gaussian distributions.

The prior $p(\mathbf{a}_N)$ is given by a zero-mean Gaussian process with covariance matrix $\mathbf{C}_N$, and the data term (assuming independence of the data points) is given by

$$p(\mathbf{t}_N|\mathbf{a}_N) = \prod_{n=1}^{N} \sigma(a_n)^{t_n}(1-\sigma(a_n))^{1-t_n} = \prod_{n=1}^{N} e^{a_n t_n}\sigma(-a_n). \tag{6.79}$$

We then obtain the Laplace approximation by Taylor expanding the logarithm of $p(\mathbf{a}_N|\mathbf{t}_N)$, which up to an additive normalization constant is given by the quantity

$$\begin{aligned}\Psi(\mathbf{a}_N) &= \ln p(\mathbf{a}_N) + \ln p(\mathbf{t}_N|\mathbf{a}_N) \\ &= -\frac{1}{2}\mathbf{a}_N^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{a}_N - \frac{N}{2}\ln(2\pi) - \frac{1}{2}\ln|\mathbf{C}_N| + \mathbf{t}_N^{\mathrm{T}}\mathbf{a}_N \\ &\quad - \sum_{n=1}^{N}\ln(1+e^{a_n}) + \text{const}. \end{aligned} \tag{6.80}$$

First we need to find the mode of the posterior distribution, and this requires that we evaluate the gradient of $\Psi(\mathbf{a}_N)$, which is given by

$$\nabla\Psi(\mathbf{a}_N) = \mathbf{t}_N - \boldsymbol{\sigma}_N - \mathbf{C}_N^{-1}\mathbf{a}_N \tag{6.81}$$

where $\boldsymbol{\sigma}_N$ is a vector with elements $\sigma(a_n)$. We cannot simply find the mode by setting this gradient to zero, because $\boldsymbol{\sigma}_N$ depends nonlinearly on $\mathbf{a}_N$, and so we resort to an iterative scheme based on the Newton-Raphson method, which gives rise to an iterative reweighted least squares (IRLS) algorithm. This requires the second derivatives of $\Psi(\mathbf{a}_N)$, which we also require for the Laplace approximation anyway, and which are given by

Section 4.3.3

$$\nabla\nabla\Psi(\mathbf{a}_N) = -\mathbf{W}_N - \mathbf{C}_N^{-1} \tag{6.82}$$

where $\mathbf{W}_N$ is a diagonal matrix with elements $\sigma(a_n)(1-\sigma(a_n))$, and we have used the result (4.88) for the derivative of the logistic sigmoid function. Note that these diagonal elements lie in the range $(0, 1/4)$, and hence $\mathbf{W}_N$ is a positive definite matrix. Because $\mathbf{C}_N$ (and hence its inverse) is positive definite by construction, and because the sum of two positive definite matrices is also positive definite, we see that the Hessian matrix $\mathbf{A} = -\nabla\nabla\Psi(\mathbf{a}_N)$ is positive definite and so the posterior distribution $p(\mathbf{a}_N|\mathbf{t}_N)$ is log convex and therefore has a single mode that is the global

Exercise 6.24

maximum. The posterior distribution is not Gaussian, however, because the Hessian is a function of $\mathbf{a}_N$.

Exercise 6.25

Using the Newton-Raphson formula (4.92), the iterative update equation for $\mathbf{a}_N$ is given by

$$\mathbf{a}_N^{\text{new}} = \mathbf{C}_N(\mathbf{I} + \mathbf{W}_N\mathbf{C}_N)^{-1}\left\{\mathbf{t}_N - \boldsymbol{\sigma}_N + \mathbf{W}_N\mathbf{a}_N\right\}. \tag{6.83}$$

These equations are iterated until they converge to the mode which we denote by $\mathbf{a}_N^\star$. At the mode, the gradient $\nabla\Psi(\mathbf{a}_N)$ will vanish, and hence $\mathbf{a}_N^\star$ will satisfy

$$\mathbf{a}_N^\star = \mathbf{C}_N(\mathbf{t}_N - \boldsymbol{\sigma}_N). \tag{6.84}$$

Once we have found the mode $\mathbf{a}_N^\star$ of the posterior, we can evaluate the Hessian matrix given by

$$\mathbf{H} = -\nabla\nabla\Psi(\mathbf{a}_N) = \mathbf{W}_N + \mathbf{C}_N^{-1} \tag{6.85}$$

where the elements of $\mathbf{W}_N$ are evaluated using $\mathbf{a}_N^\star$. This defines our Gaussian approximation to the posterior distribution $p(\mathbf{a}_N|\mathbf{t}_N)$ given by

$$q(\mathbf{a}_N) = \mathcal{N}(\mathbf{a}_N|\mathbf{a}_N^\star, \mathbf{H}^{-1}). \tag{6.86}$$

Exercise 6.26

We can now combine this with (6.78) and hence evaluate the integral (6.77). Because this corresponds to a linear-Gaussian model, we can use the general result (2.115) to give

$$\mathbb{E}[a_{N+1}|\mathbf{t}_N] = \mathbf{k}^{\mathrm{T}}(\mathbf{t}_N - \boldsymbol{\sigma}_N) \tag{6.87}$$

$$\operatorname{var}[a_{N+1}|\mathbf{t}_N] = c - \mathbf{k}^{\mathrm{T}}(\mathbf{W}_N^{-1} + \mathbf{C}_N)^{-1}\mathbf{k}. \tag{6.88}$$

Now that we have a Gaussian distribution for $p(a_{N+1}|\mathbf{t}_N)$, we can approximate the integral (6.76) using the result (4.153). As with the Bayesian logistic regression model of Section 4.5, if we are only interested in the decision boundary corresponding to $p(t_{N+1}|\mathbf{t}_N) = 0.5$, then we need only consider the mean and we can ignore the effect of the variance.

We also need to determine the parameters $\boldsymbol{\theta}$ of the covariance function. One approach is to maximize the likelihood function given by $p(\mathbf{t}_N|\boldsymbol{\theta})$ for which we need expressions for the log likelihood and its gradient. If desired, suitable regularization terms can also be added, leading to a penalized maximum likelihood solution. The likelihood function is defined by

$$p(\mathbf{t}_N|\boldsymbol{\theta}) = \int p(\mathbf{t}_N|\mathbf{a}_N)p(\mathbf{a}_N|\boldsymbol{\theta})\,\mathrm{d}\mathbf{a}_N. \tag{6.89}$$

This integral is analytically intractable, so again we make use of the Laplace approximation. Using the result (4.135), we obtain the following approximation for the log of the likelihood function

$$\ln p(\mathbf{t}_N|\boldsymbol{\theta}) = \Psi(\mathbf{a}_N^\star) - \frac{1}{2}\ln|\mathbf{W}_N + \mathbf{C}_N^{-1}| + \frac{N}{2}\ln(2\pi) \tag{6.90}$$

where $\Psi(\mathbf{a}_N^\star) = \ln p(\mathbf{a}_N^\star|\boldsymbol{\theta}) + \ln p(\mathbf{t}_N|\mathbf{a}_N^\star)$. We also need to evaluate the gradient of $\ln p(\mathbf{t}_N|\boldsymbol{\theta})$ with respect to the parameter vector $\boldsymbol{\theta}$. Note that changes in $\boldsymbol{\theta}$ will cause changes in $\mathbf{a}_N^\star$, leading to additional terms in the gradient. Thus, when we differentiate (6.90) with respect to $\boldsymbol{\theta}$, we obtain two sets of terms, the first arising from the dependence of the covariance matrix $\mathbf{C}_N$ on $\boldsymbol{\theta}$, and the rest arising from dependence of $\mathbf{a}_N^\star$ on $\boldsymbol{\theta}$.

The terms arising from the explicit dependence on $\boldsymbol{\theta}$ can be found by using (6.80) together with the results (C.21) and (C.22), and are given by

$$\begin{aligned}\frac{\partial \ln p(\mathbf{t}_N|\boldsymbol{\theta})}{\partial \theta_j} &= \frac{1}{2}\mathbf{a}_N^{\star\mathrm{T}}\mathbf{C}_N^{-1}\frac{\partial \mathbf{C}_N}{\partial \theta_j}\mathbf{C}_N^{-1}\mathbf{a}_N^\star \\ &\quad -\frac{1}{2}\mathrm{Tr}\left[(\mathbf{I}+\mathbf{C}_N\mathbf{W}_N)^{-1}\mathbf{W}_N\frac{\partial \mathbf{C}_N}{\partial \theta_j}\right].\end{aligned} \tag{6.91}$$

To compute the terms arising from the dependence of $\mathbf{a}_N^\star$ on $\boldsymbol{\theta}$, we note that the Laplace approximation has been constructed such that $\Psi(\mathbf{a}_N)$ has zero gradient at $\mathbf{a}_N = \mathbf{a}_N^\star$, and so $\Psi(\mathbf{a}_N^\star)$ gives no contribution to the gradient as a result of its dependence on $\mathbf{a}_N^\star$. This leaves the following contribution to the derivative with respect to a component θ_j of $\boldsymbol{\theta}$

$$\begin{aligned}&-\frac{1}{2}\sum_{n=1}^{N}\frac{\partial \ln|\mathbf{W}_N+\mathbf{C}_N^{-1}|}{\partial a_n^\star}\frac{\partial a_n^\star}{\partial \theta_j} \\ &\qquad = -\frac{1}{2}\sum_{n=1}^{N}\left[(\mathbf{I}+\mathbf{C}_N\mathbf{W}_N)^{-1}\mathbf{C}_N\right]_{nn}\sigma_n^\star(1-\sigma_n^\star)(1-2\sigma_n^\star)\frac{\partial a_n^\star}{\partial \theta_j}\end{aligned} \tag{6.92}$$

where $\sigma_n^\star = \sigma(a_n^\star)$, and again we have used the result (C.22) together with the definition of $\mathbf{W}_N$. We can evaluate the derivative of $a_N^\star$ with respect to θ_j by differentiating the relation (6.84) with respect to θ_j to give

$$\frac{\partial a_n^\star}{\partial \theta_j} = \frac{\partial \mathbf{C}_N}{\partial \theta_j}(\mathbf{t}_N - \boldsymbol{\sigma}_N) - \mathbf{C}_N\mathbf{W}_N\frac{\partial a_n^\star}{\partial \theta_j}. \tag{6.93}$$

Rearranging then gives

$$\frac{\partial a_n^\star}{\partial \theta_j} = (\mathbf{I}+\mathbf{W}_N\mathbf{C}_N)^{-1}\frac{\partial \mathbf{C}_N}{\partial \theta_j}(\mathbf{t}_N - \boldsymbol{\sigma}_N). \tag{6.94}$$

Combining (6.91), (6.92), and (6.94), we can evaluate the gradient of the log likelihood function, which can be used with standard nonlinear optimization algorithms in order to determine a value for $\boldsymbol{\theta}$.

We can illustrate the application of the Laplace approximation for Gaussian processes using the synthetic two-class data set shown in Figure 6.12. Extension of the Laplace approximation to Gaussian processes involving $K > 2$ classes, using the softmax activation function, is straightforward (Williams and Barber, 1998).

Appendix A

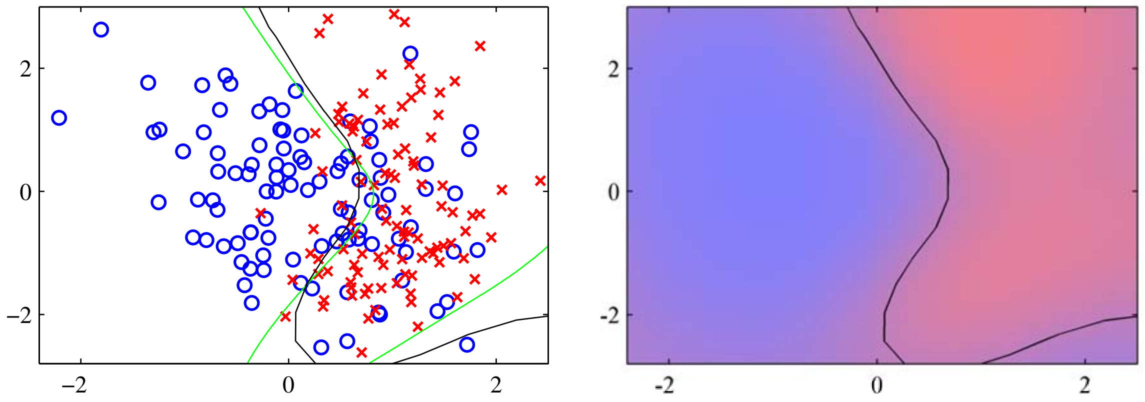

Figure 6.12 Illustration of the use of a Gaussian process for classification, showing the data on the left together with the optimal decision boundary from the true distribution in green, and the decision boundary from the Gaussian process classifier in black. On the right is the predicted posterior probability for the blue and red classes together with the Gaussian process decision boundary.

6.4.7 Connection to neural networks

We have seen that the range of functions which can be represented by a neural network is governed by the number M of hidden units, and that, for sufficiently large M, a two-layer network can approximate any given function with arbitrary accuracy. In the framework of maximum likelihood, the number of hidden units needs to be limited (to a level dependent on the size of the training set) in order to avoid over-fitting. However, from a Bayesian perspective it makes little sense to limit the number of parameters in the network according to the size of the training set.

In a Bayesian neural network, the prior distribution over the parameter vector $\mathbf{w}$, in conjunction with the network function $f(\mathbf{x}, \mathbf{w})$, produces a prior distribution over functions from $y(\mathbf{x})$ where $\mathbf{y}$ is the vector of network outputs. Neal (1996) has shown that, for a broad class of prior distributions over $\mathbf{w}$, the distribution of functions generated by a neural network will tend to a Gaussian process in the limit $M \to \infty$. It should be noted, however, that in this limit the output variables of the neural network become independent. One of the great merits of neural networks is that the outputs share the hidden units and so they can 'borrow statistical strength' from each other, that is, the weights associated with each hidden unit are influenced by all of the output variables not just by one of them. This property is therefore lost in the Gaussian process limit.

We have seen that a Gaussian process is determined by its covariance (kernel) function. Williams (1998) has given explicit forms for the covariance in the case of two specific choices for the hidden unit activation function (probit and Gaussian). These kernel functions $k(\mathbf{x}, \mathbf{x}')$ are nonstationary, i.e. they cannot be expressed as a function of the difference $\mathbf{x} - \mathbf{x}'$, as a consequence of the Gaussian weight prior being centred on zero which breaks translation invariance in weight space.

By working directly with the covariance function we have implicitly marginalized over the distribution of weights. If the weight prior is governed by hyperparameters, then their values will determine the length scales of the distribution over functions, as can be understood by studying the examples in Figure 5.11 for the case of a finite number of hidden units. Note that we cannot marginalize out the hyperparameters analytically, and must instead resort to techniques of the kind discussed in Section 6.4.

Exercises

6.1 $(\star\star)$ **www** Consider the dual formulation of the least squares linear regression problem given in Section 6.1. Show that the solution for the components a_n of the vector $\mathbf{a}$ can be expressed as a linear combination of the elements of the vector $\boldsymbol{\phi}(\mathbf{x}_n)$. Denoting these coefficients by the vector $\mathbf{w}$, show that the dual of the dual formulation is given by the original representation in terms of the parameter vector $\mathbf{w}$.

6.2 $(\star\star)$ In this exercise, we develop a dual formulation of the perceptron learning algorithm. Using the perceptron learning rule (4.55), show that the learned weight vector $\mathbf{w}$ can be written as a linear combination of the vectors $t_n\boldsymbol{\phi}(\mathbf{x}_n)$ where $t_n \in \{-1, +1\}$. Denote the coefficients of this linear combination by α_n and derive a formulation of the perceptron learning algorithm, and the predictive function for the perceptron, in terms of the α_n. Show that the feature vector $\boldsymbol{\phi}(\mathbf{x})$ enters only in the form of the kernel function $k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}')$.

6.3 $(\star)$ The nearest-neighbour classifier (Section 2.5.2) assigns a new input vector $\mathbf{x}$ to the same class as that of the nearest input vector $\mathbf{x}_n$ from the training set, where in the simplest case, the distance is defined by the Euclidean metric $\|\mathbf{x} - \mathbf{x}_n\|^2$. By expressing this rule in terms of scalar products and then making use of kernel substitution, formulate the nearest-neighbour classifier for a general nonlinear kernel.

6.4 $(\star)$ In Appendix C, we give an example of a matrix that has positive elements but that has a negative eigenvalue and hence that is not positive definite. Find an example of the converse property, namely a 2×2 matrix with positive eigenvalues yet that has at least one negative element.

6.5 $(\star)$ **www** Verify the results (6.13) and (6.14) for constructing valid kernels.

6.6 $(\star)$ Verify the results (6.15) and (6.16) for constructing valid kernels.

6.7 $(\star)$ **www** Verify the results (6.17) and (6.18) for constructing valid kernels.

6.8 $(\star)$ Verify the results (6.19) and (6.20) for constructing valid kernels.

6.9 $(\star)$ Verify the results (6.21) and (6.22) for constructing valid kernels.

6.10 $(\star)$ Show that an excellent choice of kernel for learning a function $f(\mathbf{x})$ is given by $k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})f(\mathbf{x}')$ by showing that a linear learning machine based on this kernel will always find a solution proportional to $f(\mathbf{x})$.

6.11 ($\star$) By making use of the expansion (6.25), and then expanding the middle factor as a power series, show that the Gaussian kernel (6.23) can be expressed as the inner product of an infinite-dimensional feature vector.

6.12 ($\star\star$) **www** Consider the space of all possible subsets A of a given fixed set D. Show that the kernel function (6.27) corresponds to an inner product in a feature space of dimensionality $2^{|D|}$ defined by the mapping $\boldsymbol{\phi}(A)$ where A is a subset of D and the element $\phi_U(A)$, indexed by the subset U, is given by

$$\phi_U(A) = \begin{cases} 1, & \text{if } U \subseteq A; \\ 0, & \text{otherwise.} \end{cases} \tag{6.95}$$

Here $U \subseteq A$ denotes that U is either a subset of A or is equal to A.

6.13 ($\star$) Show that the Fisher kernel, defined by (6.33), remains invariant if we make a nonlinear transformation of the parameter vector $\boldsymbol{\theta} \to \boldsymbol{\psi}(\boldsymbol{\theta})$, where the function $\boldsymbol{\psi}(\cdot)$ is invertible and differentiable.

6.14 ($\star$) **www** Write down the form of the Fisher kernel, defined by (6.33), for the case of a distribution $p(\mathbf{x}|\boldsymbol{\mu}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \mathbf{S})$ that is Gaussian with mean $\boldsymbol{\mu}$ and fixed covariance $\mathbf{S}$.

6.15 ($\star$) By considering the determinant of a 2×2 Gram matrix, show that a positive-definite kernel function $k(x, x')$ satisfies the Cauchy-Schwartz inequality

$$k(x_1, x_2)^2 \leqslant k(x_1, x_1)k(x_2, x_2). \tag{6.96}$$

6.16 ($\star\star$) Consider a parametric model governed by the parameter vector $\mathbf{w}$ together with a data set of input values $\mathbf{x}_1, \ldots, \mathbf{x}_N$ and a nonlinear feature mapping $\boldsymbol{\phi}(\mathbf{x})$. Suppose that the dependence of the error function on $\mathbf{w}$ takes the form

$$J(\mathbf{w}) = f(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_1), \ldots, \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_N)) + g(\mathbf{w}^{\mathrm{T}}\mathbf{w}) \tag{6.97}$$

where $g(\cdot)$ is a monotonically increasing function. By writing $\mathbf{w}$ in the form

$$\mathbf{w} = \sum_{n=1}^{N} \alpha_n \boldsymbol{\phi}(\mathbf{x}_n) + \mathbf{w}_{\perp} \tag{6.98}$$

show that the value of $\mathbf{w}$ that minimizes $J(\mathbf{w})$ takes the form of a linear combination of the basis functions $\boldsymbol{\phi}(\mathbf{x}_n)$ for $n = 1, \ldots, N$.

6.17 ($\star\star$) **www** Consider the sum-of-squares error function (6.39) for data having noisy inputs, where $\nu(\boldsymbol{\xi})$ is the distribution of the noise. Use the calculus of variations to minimize this error function with respect to the function $y(\mathbf{x})$, and hence show that the optimal solution is given by an expansion of the form (6.40) in which the basis functions are given by (6.41).

6.18 ($\star$) Consider a Nadaraya-Watson model with one input variable x and one target variable t having Gaussian components with isotropic covariances, so that the covariance matrix is given by $\sigma^2\mathbf{I}$ where $\mathbf{I}$ is the unit matrix. Write down expressions for the conditional density $p(t|x)$ and for the conditional mean $\mathbb{E}[t|x]$ and variance $\text{var}[t|x]$, in terms of the kernel function $k(x, x_n)$.

6.19 ($\star\star$) Another viewpoint on kernel regression comes from a consideration of regression problems in which the input variables as well as the target variables are corrupted with additive noise. Suppose each target value t_n is generated as usual by taking a function $y(\mathbf{z}_n)$ evaluated at a point $\mathbf{z}_n$, and adding Gaussian noise. The value of $\mathbf{z}_n$ is not directly observed, however, but only a noise corrupted version $\mathbf{x}_n = \mathbf{z}_n + \boldsymbol{\xi}_n$ where the random variable $\boldsymbol{\xi}$ is governed by some distribution $g(\boldsymbol{\xi})$. Consider a set of observations $\{\mathbf{x}_n, t_n\}$, where $n = 1, \ldots, N$, together with a corresponding sum-of-squares error function defined by averaging over the distribution of input noise to give

$$E = \frac{1}{2}\sum_{n=1}^{N}\int \{y(\mathbf{x}_n - \boldsymbol{\xi}_n) - t_n\}^2 g(\boldsymbol{\xi}_n)\,\mathrm{d}\boldsymbol{\xi}_n. \tag{6.99}$$

By minimizing E with respect to the function $y(\mathbf{z})$ using the calculus of variations (Appendix D), show that optimal solution for $y(\mathbf{x})$ is given by a Nadaraya-Watson kernel regression solution of the form (6.45) with a kernel of the form (6.46).

6.20 ($\star\star$) **www** Verify the results (6.66) and (6.67).

6.21 ($\star\star$) **www** Consider a Gaussian process regression model in which the kernel function is defined in terms of a fixed set of nonlinear basis functions. Show that the predictive distribution is identical to the result (3.58) obtained in Section 3.3.2 for the Bayesian linear regression model. To do this, note that both models have Gaussian predictive distributions, and so it is only necessary to show that the conditional mean and variance are the same. For the mean, make use of the matrix identity (C.6), and for the variance, make use of the matrix identity (C.7).

6.22 ($\star\star$) Consider a regression problem with N training set input vectors $\mathbf{x}_1, \ldots, \mathbf{x}_N$ and L test set input vectors $\mathbf{x}_{N+1}, \ldots, \mathbf{x}_{N+L}$, and suppose we define a Gaussian process prior over functions $t(\mathbf{x})$. Derive an expression for the joint predictive distribution for $t(\mathbf{x}_{N+1}), \ldots, t(\mathbf{x}_{N+L})$, given the values of $t(\mathbf{x}_1), \ldots, t(\mathbf{x}_N)$. Show the marginal of this distribution for one of the test observations t_j where $N + 1 \leqslant j \leqslant N + L$ is given by the usual Gaussian process regression result (6.66) and (6.67).

6.23 ($\star\star$) **www** Consider a Gaussian process regression model in which the target variable $\mathbf{t}$ has dimensionality D. Write down the conditional distribution of $\mathbf{t}_{N+1}$ for a test input vector $\mathbf{x}_{N+1}$, given a training set of input vectors $\mathbf{x}_1, \ldots, \mathbf{x}_{N+1}$ and corresponding target observations $\mathbf{t}_1, \ldots, \mathbf{t}_N$.

6.24 ($\star$) Show that a diagonal matrix $\mathbf{W}$ whose elements satisfy $0 < W_{ii} < 1$ is positive definite. Show that the sum of two positive definite matrices is itself positive definite.

6.25 ($\star$) **www** Using the Newton-Raphson formula (4.92), derive the iterative update formula (6.83) for finding the mode $\mathbf{a}_N^\star$ of the posterior distribution in the Gaussian process classification model.

6.26 ($\star$) Using the result (2.115), derive the expressions (6.87) and (6.88) for the mean and variance of the posterior distribution $p(a_{N+1}|\mathbf{t}_N)$ in the Gaussian process classification model.

6.27 ($\star\star\star$) Derive the result (6.90) for the log likelihood function in the Laplace approximation framework for Gaussian process classification. Similarly, derive the results (6.91), (6.92), and (6.94) for the terms in the gradient of the log likelihood.