

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

Section 2.5.1

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

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 $\phi(\mathbf{x})$, the kernel function is given by the relation

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

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.

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

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 $k(\mathbf{x}, \mathbf{x}') = k(\|\mathbf{x} - \mathbf{x}'\|)$.

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

Chapter 7

Section 12.3

Section 6.3

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

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

where $\lambda \geq 0$. 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 $\phi(\mathbf{x}_n)$, with coefficients that are functions of \mathbf{w} , of the form

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

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

$$a_n = -\frac{1}{\lambda} \{ \mathbf{w}^T \phi(\mathbf{x}_n) - t_n \}. \quad (6.4)$$

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 $\mathbf{w} = \Phi^T \mathbf{a}$ into $J(\mathbf{w})$, we obtain

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

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

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

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

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

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

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

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

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

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 $\phi(\mathbf{x})$, 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 $\phi(\mathbf{x})$, which allows us implicitly to use feature spaces of high, even infinite, dimensionality.

Exercise 6.2

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.

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 $\phi(\mathbf{x})$ 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

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

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

$$k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T \mathbf{z})^2. \quad (6.11)$$

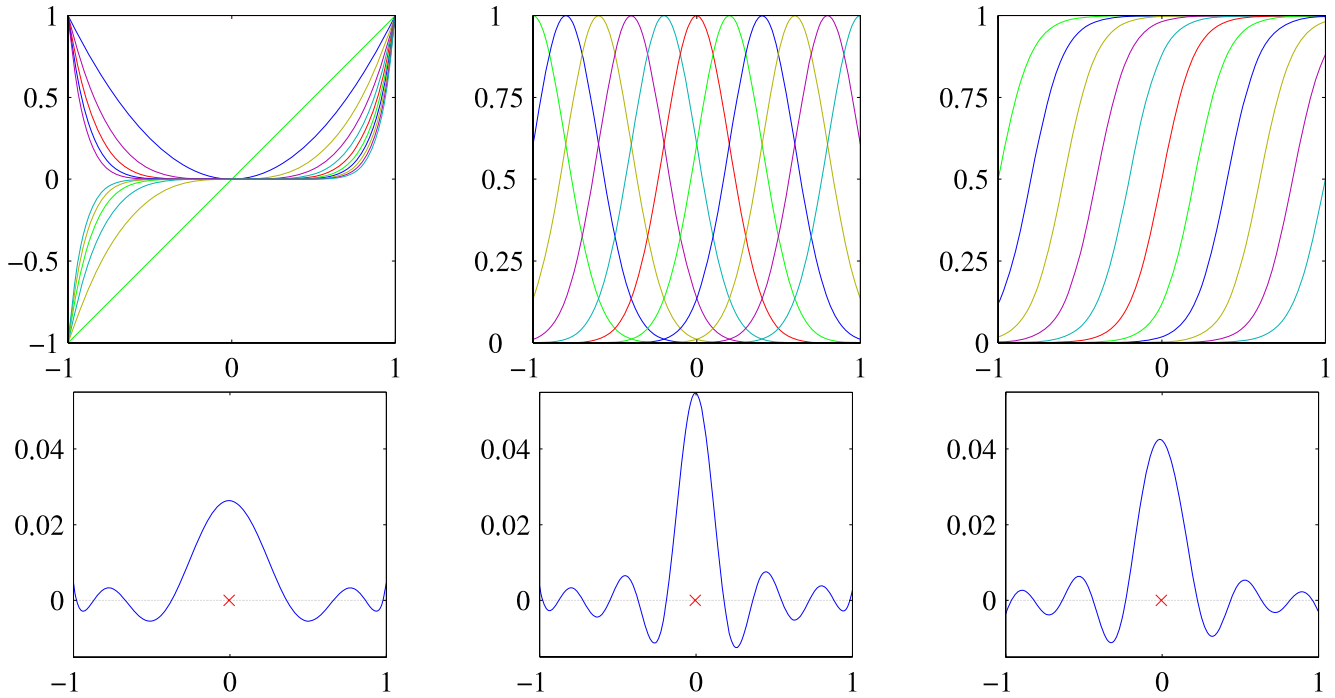


Figure 6.1 Illustration of the construction of kernel functions starting from a corresponding set of basis functions. In each column the lower plot shows the kernel function $k(x, x')$ defined by (6.10) plotted as a function of x for $x' = 0$, while the upper plot shows the corresponding basis functions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

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

We see that the feature mapping takes the form $\phi(\mathbf{x}) = (x_1^2, \sqrt{2}x_1 x_2, x_2^2)^T$ 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 $\phi(\mathbf{x})$ 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:

$$k(\mathbf{x}, \mathbf{x}') = ck_1(\mathbf{x}, \mathbf{x}') \quad (6.13)$$

$$k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})k_1(\mathbf{x}, \mathbf{x}')f(\mathbf{x}') \quad (6.14)$$

$$k(\mathbf{x}, \mathbf{x}') = q(k_1(\mathbf{x}, \mathbf{x}')) \quad (6.15)$$

$$k(\mathbf{x}, \mathbf{x}') = \exp(k_1(\mathbf{x}, \mathbf{x}')) \quad (6.16)$$

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') \quad (6.17)$$

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}') \quad (6.18)$$

$$k(\mathbf{x}, \mathbf{x}') = k_3(\phi(\mathbf{x}), \phi(\mathbf{x}')) \quad (6.19)$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{A} \mathbf{x}' \quad (6.20)$$

$$k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x}_a, \mathbf{x}'_a) + k_b(\mathbf{x}_b, \mathbf{x}'_b) \quad (6.21)$$

$$k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x}_a, \mathbf{x}'_a)k_b(\mathbf{x}_b, \mathbf{x}'_b) \quad (6.22)$$

where $c > 0$ is a constant, $f(\cdot)$ is any function, $q(\cdot)$ is a polynomial with nonnegative coefficients, $\phi(\mathbf{x})$ 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 $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}')^2$ contains only terms of degree two. If we consider the slightly generalized kernel $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}' + c)^2$ with $c > 0$, then the corresponding feature mapping $\phi(\mathbf{x})$ contains constant and linear terms as well as terms of order two. Similarly, $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}')^M$ 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 $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}' + c)^M$ 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

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

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

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

to give

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

and then making use of (6.14) and (6.16), together with the validity of the linear kernel $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'$. 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 $\mathbf{x}^T \mathbf{x}'$ with a nonlinear kernel $\kappa(\mathbf{x}, \mathbf{x}')$, we obtain

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

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

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

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

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

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

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

Section 9.2

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

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

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

Section 13.2

Now suppose that our data consists of ordered sequences of length L so that an observation is given by $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_L\}$. 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 $\mathbf{Z} = \{\mathbf{z}_1, \dots, \mathbf{z}_L\}$. 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

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

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*

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

from which the Fisher kernel is defined by

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

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

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

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 $\boldsymbol{\theta} \rightarrow \boldsymbol{\psi}(\boldsymbol{\theta})$.

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

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

Section 12.1.3

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

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

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

$$k(\mathbf{x}, \mathbf{x}') = \tanh(a\mathbf{x}^T \mathbf{x}' + b) \quad (6.37)$$

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 $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ along with corresponding target values $\{t_1, \dots, t_N\}$, 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 $n = 1, \dots, N$. This is achieved by expressing $f(\mathbf{x})$ as a linear combination of radial basis functions, one centred on every data point

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

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