3

Kernel-Induced Feature Spaces

The limited computational power of linear learning machines was highlighted in the 1960s by Minsky and Papert. In general, complex real-world applications require more expressive hypothesis spaces than linear functions. Another way of viewing this problem is that frequently the target concept cannot be expressed as a simple linear combination of the given attributes, but in general requires that more abstract features of the data be exploited. Multiple layers of thresholded linear functions were proposed as a solution to this problem, and this approach led to the development of multi-layer neural networks and learning algorithms such as back-propagation for training such systems.

Kernel representations offer an alternative solution by projecting the data into a high dimensional feature space to increase the computational power of the linear learning machines of Chapter 2. The use of linear machines in the dual representation makes it possible to perform this step implicitly. As noted in Chapter 2, the training examples never appear isolated but always in the form of inner products between pairs of examples. The advantage of using the machines in the dual representation derives from the fact that in this representation the number of tunable parameters does not depend on the number of attributes being used. By replacing the inner product with an appropriately chosen 'kernel' function, one can implicitly perform a non-linear mapping to a high dimensional feature space without increasing the number of tunable parameters, provided the kernel computes the inner product of the feature vectors corresponding to the two inputs.

In this chapter we will introduce the kernel technique, which provides one of the main building blocks of Support Vector Machines. One of the remarkable features of SVMs is that to a certain extent the approximation-theoretic issues are independent of the learning-theoretic ones. One can therefore study the properties of the kernel representations in a general and self-contained way, and use them with different learning theories, as we will see in Chapter 4.

Another attraction of the kernel method is that the learning algorithms and theory can largely be decoupled from the specifics of the application area, which must simply be encoded into the design of an appropriate kernel function. Hence, the problem of choosing an architecture for a neural network application is replaced by the problem of choosing a suitable kernel for a Support Vector Machine. In this chapter we will describe some well-known kernels and show how more complicated kernels can be constructed by combining simpler ones. We will also mention kernels

that have been developed for discrete structures such as text, showing that the approach is not restricted only to input spaces that are subsets of Euclidean space, and hence can be applied even when we were unable to define linear functions over the input space.

As we will see in Chapters 4 and 7, the use of kernels can overcome the curse of dimensionality in both computation and generalisation.

3.1 Learning in Feature Space

The complexity of the target function to be learned depends on the way it is represented, and the difficulty of the learning task can vary accordingly. Ideally a representation that matches the specific learning problem should be chosen. So one common preprocessing strategy in machine learning involves changing the representation of the data:

$$\mathbf{x} = (x_1, \dots, x_n) \longmapsto \boldsymbol{\phi}(\mathbf{x}) = (\phi_1(\mathbf{x}), \dots, \phi_N(\mathbf{x})).$$

This step is equivalent to mapping the input space X into a new space, $F = \{\boldsymbol{\phi}(\mathbf{x}) | \mathbf{x} \in X\}$.

Example 3.1 Consider the target function

$$f(m_1, m_2, r) = C\frac{m_1 m_2}{r^2},$$

giving Newton's law of gravitation, expressing the gravitational force between two bodies with masses m_1, m_2 and separation r. This law is expressed in terms of the observable quantities, mass and distance. A linear machine such as those described in Chapter 2 could not represent it as written, but a simple change of coordinates

$$(m_1, m_2, r) \longmapsto (x, y, z) = (\ln m_1, \ln m_2, \ln r)$$

gives the representation

$$g(x, y, z) = \ln f(m_1, m_2, r) = \ln C + \ln m_1 + \ln m_2 - 2\ln r = c + x + y - 2z,$$

which could be learned by a linear machine.

The fact that simply mapping the data into another space can greatly simplify the task has been known for a long time in machine learning, and has given rise to a number of techniques for selecting the best representation of data. The quantities introduced to describe the data are usually called *features*, while the original quantities are sometimes called *attributes*. The task of choosing the most suitable representation is known as *feature selection*. The space X is referred to as the input space, while $F = \{\boldsymbol{\phi}(\mathbf{x}) : \mathbf{x} \in X\}$ is called the *feature space*.

Figure 3.1 shows an example of a feature mapping from a two dimensional input space to a two dimensional feature space, where the data cannot be

https://doi.org/10.1017/CBO9780511801389.005 Published online by Cambridge University Press

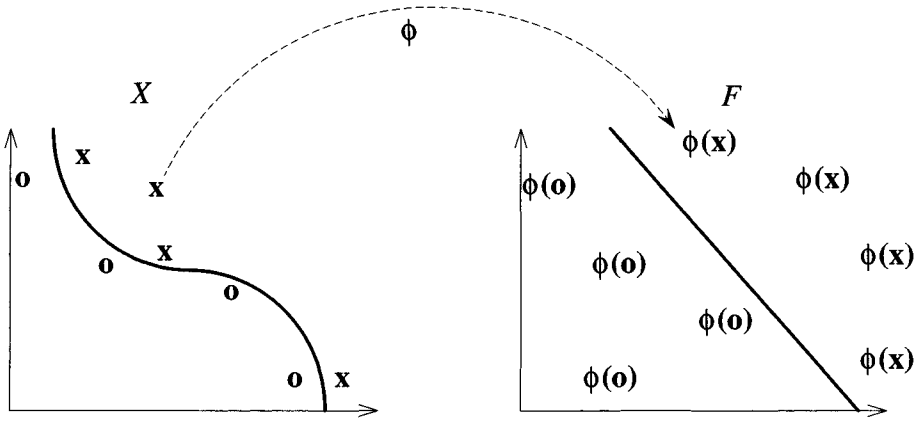

Figure 3.1: A feature map can simplify the classification task

separated by a linear function in the input space, but can be in the feature space. The aim of this chapter is to show how such mappings can be made into very high dimensional spaces where linear separation becomes much easier.

Different approaches to feature selection exist. Frequently one seeks to identify the smallest set of features that still conveys the essential information contained in the original attributes. This is known as *dimensionality reduction*,

$$\mathbf{x} = (x_1, \ldots, x_n) \longmapsto \boldsymbol{\phi}(\mathbf{x}) = (\phi_1(\mathbf{x}), \ldots, \phi_d(\mathbf{x})), \quad d < n,$$

and can be very beneficial as both computational and generalisation performance can degrade as the number of features grows, a phenomenon sometimes referred to as the *curse of dimensionality*. The difficulties with high dimensional feature spaces are unfortunate, since the larger the set of (possibly redundant) features, the more likely that the function to be learned can be represented using a standardised learning machine. We will show how this degradation of performance can be avoided in the Support Vector Machine.

Example 3.2 Again consider the gravitation law for two bodies, but suppose the attributes are now the three components of their positions together with their masses:

$$\mathbf{x} = (p_1^x, p_1^y, p_1^z, p_2^x, p_2^y, p_2^z, m_1, m_2).$$

One way to reduce the dimensionality of the problem would be the following mapping $\boldsymbol{\phi} : \mathbb{R}^8 \longmapsto \mathbb{R}^3$:

$$\mathbf{x} = (p_1^x, p_1^y, p_1^z, p_2^x, p_2^y, p_2^z, m_1, m_2) \mapsto \boldsymbol{\phi}(\mathbf{x}) = \left(\sqrt{\sum_{i \in \{x,y,z\}} (p_1^i - p_2^i)^2}, m_1, m_2 \right),$$

which would retain the essential information.

https://doi.org/10.1017/CBO9780511801389.005 Published online by Cambridge University Press

Another very different feature selection task is the detection of *irrelevant features* and their subsequent elimination. In our example, an irrelevant feature would be the colour of the two bodies, or their temperature, since neither quantity affects the target output value.

The use of principal components analysis provides a mapping of the data to a feature space in which the new features are linear functions of the original attributes and are sorted by the amount of variance that the data exhibit in each direction. Dimensionality reduction can sometimes be performed by simply removing features corresponding to directions in which the data have low variance, though there is no guarantee that these features are not essential for performing the target classification. We now give an example where additional feature dimensions can be useful.

Example 3.3 Consider the case of a two dimensional input space, and assume our prior knowledge about the problem suggests that relevant information is encoded in the form of monomials of degree 2. Hence we want to represent the problem in a feature space where such information is made explicit, and is ready for the learning machine to use. A possible mapping is the following:

$$(x_1, x_2) \longmapsto \phi(x_1, x_2) = (x_1^2, x_2^2, x_1 x_2).$$

In the same way we might want to use features of degree d, giving a feature space of $\binom{n+d-1}{d}$ dimensions, a number that soon becomes computationally infeasible for reasonable numbers of attributes and feature degrees. The use of this type of feature space will require a special technique, introduced in Section 3.2, involving an 'implicit mapping' into the feature space.

The computational problems are not the only ones connected with the size of the feature space we are using. Another source of difficulties is the generalisation of the learning machine, which can be sensitive to the dimensionality of the representation for standard function classes of hypotheses.

It is evident from the previous examples that feature selection should be viewed as a part of the learning process itself, and should be automated as much as possible. On the other hand, it is a somewhat arbitrary step, which reflects our prior expectations on the underlying target function. The theoretical models of learning should also take account of this step: using too large a set of features can create overfitting problems, unless the generalisation can be controlled in some way. It is for this reason that research has frequently concentrated on dimensionality reduction techniques. However, we will see in Chapter 4 that a deeper understanding of generalisation means that we can even afford to use infinite dimensional feature spaces. The generalisation problems will be avoided by using learning machines based on this understanding, while computational problems are avoided by means of the 'implicit mapping' described in the next section.

3.2 The Implicit Mapping into Feature Space

In order to learn non-linear relations with a linear machine, we need to select a set of non-linear features and to rewrite the data in the new representation. This is equivalent to applying a fixed non-linear mapping of the data to a feature space, in which the linear machine can be used. Hence, the set of hypotheses we consider will be functions of the type

$$f(\mathbf{x}) = \sum_{i=1}^{N} w_i \phi_i(\mathbf{x}) + b,$$

where $\phi : X \to F$ is a non-linear map from the input space to some feature space. This means that we will build non-linear machines in two steps: first a fixed non-linear mapping transforms the data into a feature space F, and then a linear machine is used to classify them in the feature space.

As shown in Chapter 2, one important property of linear learning machines is that they can be expressed in a dual representation. This means that the hypothesis can be expressed as a linear combination of the training points, so that the decision rule can be evaluated using just inner products between the test point and the training points:

$$f(\mathbf{x}) = \sum_{i=1}^{\ell} \alpha_i y_i \langle \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}) \rangle + b.$$

If we have a way of computing the inner product $\langle \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}) \rangle$ in feature space directly as a function of the original input points, it becomes possible to merge the two steps needed to build a non-linear learning machine. We call such a direct computation method a *kernel* function.

Definition 3.4 A *kernel* is a function K, such that for all $\mathbf{x}, \mathbf{z} \in X$

$$K(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}) \cdot \phi(\mathbf{z}) \rangle,$$

where ϕ is a mapping from X to an (inner product) feature space F.

The name 'kernel' is derived from integral operator theory, which underpins much of the theory of the relation between kernels and their corresponding feature spaces. An important consequence of the dual representation is that the dimension of the feature space need not affect the computation. As one does not represent the feature vectors explicitly, the number of operations required to compute the inner product by evaluating the kernel function is not necessarily proportional to the number of features. The use of kernels makes it possible to map the data implicitly into a feature space and to train a linear machine in such a space, potentially side-stepping the computational problems inherent in evaluating the feature map. The only information used about the training examples is their Gram matrix (see Remark 2.11) in the feature space. This matrix is also referred to as the *kernel matrix*, and in this context we will use

the symbol **K** to denote it. The key to this approach is finding a kernel function that can be evaluated efficiently. Once we have such a function the decision rule can be evaluated by at most ℓ evaluations of the kernel:

$$f(\mathbf{x}) = \sum_{i=1}^{\ell} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b.$$

One of the curious facts about using a kernel is that we do not need to know the underlying feature map in order to be able to learn in the feature space! The rest of this chapter will be concerned with the problem of creating such kernel functions. We will consider the properties that they must satisfy as well as some of the more recent methods developed for their construction. The concept of a kernel is central to the development of the book, but it is not an immediately intuitive idea. First note that the idea of a kernel generalises the standard inner product in the input space. It is clear that this inner product provides an example of a kernel by making the feature map the identity

$$K(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x} \cdot \mathbf{z} \rangle.$$

We can also take the feature map to be any fixed linear transformation $\mathbf{x} \mapsto \mathbf{Ax}$, for some matrix **A**. In this case the kernel function is given by

$$K(\mathbf{x}, \mathbf{z}) = \langle \mathbf{Ax} \cdot \mathbf{Az} \rangle = \mathbf{x}'\mathbf{A}'\mathbf{Az} = \mathbf{x}'\mathbf{Bz},$$

where by construction $\mathbf{B} = \mathbf{A}'\mathbf{A}$ is a square symmetric positive semi-definite matrix. As discussed in the introduction the aim is to introduce non-linearity into the feature space map. We therefore move to a simple but illustrative example of such a non-linear map obtained by considering the following relation:

$$\begin{aligned} \langle \mathbf{x} \cdot \mathbf{z} \rangle^2 &= \left(\sum_{i=1}^{n} x_i z_i \right)^2 = \left(\sum_{i=1}^{n} x_i z_i \right) \left(\sum_{j=1}^{n} x_j z_j \right) \\ &= \sum_{i=1}^{n} \sum_{j=1}^{n} x_i x_j z_i z_j = \sum_{(i,j)=(1,1)}^{(n,n)} (x_i x_j)(z_i z_j), \end{aligned}$$

which is equivalent to an inner product between the feature vectors

$$\phi(\mathbf{x}) = (x_i x_j)_{(i,j)=(1,1)}^{(n,n)}.$$

In this case the features are all the monomials of degree 2 considered in Example 3.3, though note that when $i \neq j$ the feature $x_i x_j$ occurs twice, giving it double the weight of the features x_i^2. A more general feature space is obtained by

https://doi.org/10.1017/CBO9780511801389.005 Published online by Cambridge University Press

considering the kernel

$$\begin{aligned}\left(\langle \mathbf{x} \cdot \mathbf{z} \rangle + c\right)^2 &= \left(\sum_{i=1}^{n} x_i z_i + c\right)\left(\sum_{j=1}^{n} x_j z_j + c\right) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j z_i z_j + 2c\sum_{i=1}^{n} x_i z_i + c^2 \\ &= \sum_{(i,j)=(1,1)}^{(n,n)} \left(x_i x_j\right)\left(z_i z_j\right) + \sum_{i=1}^{n}\left(\sqrt{2c}x_i\right)\left(\sqrt{2c}z_i\right) + c^2,\end{aligned}$$

whose $\binom{n+1}{2} + n + 1 = \binom{n+2}{2}$ features are all the monomials of degree up to 2, but with the relative weightings between the degree 1 and 2 features controlled by the parameter c, which also determines the strength of the degree 0 or constant feature. Similar derivations can be made for the kernel functions

$$K(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x} \cdot \mathbf{z} \rangle^d \text{ and } K(\mathbf{x}, \mathbf{z}) = \left(\langle \mathbf{x} \cdot \mathbf{z} \rangle + c\right)^d,$$

for $d \geq 2$. For the first kernel the $\binom{n+d-1}{d}$ distinct features are all the monomials of degree d, though again the weighting will vary according to the structure of the exponents. For the second kernel there are $\binom{n+d}{d}$ distinct features, being all the monomials up to and including degree d. The decision boundary in the input space corresponding to a hyperplane in these feature spaces is a polynomial curve of degree d, so these kernels are frequently called *polynomial kernels.*

More complex kernels are possible, and the next section is concerned with the important problem of the construction of kernels. A significant aspect of this problem is the mathematical characterisation of functions $K(\mathbf{x}, \mathbf{z})$ that constitute kernels. We will see that a theorem from functional analysis provides an answer to this question.

3.3 Making Kernels

The use of a kernel function is an attractive computational short-cut. If we wish to use this approach, there appears to be a need to first create a complicated feature space, then work out what the inner product in that space would be, and finally find a direct method of computing that value in terms of the original inputs. In practice the approach taken is to define a kernel function directly, hence implicitly defining the feature space. In this way, we avoid the feature space not only in the computation of inner products, but also in the design of the learning machine itself. We will argue that defining a kernel function for an input space is frequently more natural than creating a complicated feature space. Before we can follow this route, however, we must first determine what properties of a function $K(\mathbf{x}, \mathbf{z})$ are necessary to ensure that it is a kernel for some feature space. Clearly, the function must be symmetric,

$$K(\mathbf{x}, \mathbf{z}) = \langle \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{z}) \rangle = \langle \boldsymbol{\phi}(\mathbf{z}) \cdot \boldsymbol{\phi}(\mathbf{x}) \rangle = K(\mathbf{z}, \mathbf{x}),$$

and satisfy the inequalities that follow from the Cauchy-Schwarz inequality,

$$\begin{aligned} K(\mathbf{x},\mathbf{z})^2 &= \langle \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{z}) \rangle^2 \leq \|\boldsymbol{\phi}(\mathbf{x})\|^2 \, \|\boldsymbol{\phi}(\mathbf{z})\|^2 \\ &= \langle \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{x}) \rangle \, \langle \boldsymbol{\phi}(\mathbf{z}) \cdot \boldsymbol{\phi}(\mathbf{z}) \rangle = K(\mathbf{x},\mathbf{x}) K(\mathbf{z},\mathbf{z}). \end{aligned}$$

These conditions are, however, not sufficient to guarantee the existence of a feature space.

3.3.1 Characterisation of Kernels

Mercer's Theorem

In this subsubsection we will introduce Mercer's theorem, which provides a characterisation of when a function $K(\mathbf{x},\mathbf{z})$ is a kernel. We begin by considering a simple case in order to motivate the result. Consider a finite input space $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, and suppose $K(\mathbf{x},\mathbf{z})$ is a symmetric function on X. Consider the matrix

$$\mathbf{K} = \left(K(\mathbf{x}_i,\mathbf{x}_j)\right)_{i,j=1}^n.$$

Since $\mathbf{K}$ is symmetric there is an orthogonal matrix $\mathbf{V}$ such that $\mathbf{K} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}'$, where $\boldsymbol{\Lambda}$ is a diagonal matrix containing the eigenvalues λ_t of $\mathbf{K}$, with corresponding eigenvectors $\mathbf{v}_t = (v_{ti})_{i=1}^n$ the columns of $\mathbf{V}$. Now assume that all the eigenvalues are non-negative and consider the feature mapping

$$\boldsymbol{\phi} : \mathbf{x}_i \mapsto \left(\sqrt{\lambda_t} v_{ti}\right)_{t=1}^n \in \mathbb{R}^n, \; i = 1, \ldots, n.$$

We now have that

$$\left\langle \boldsymbol{\phi}(\mathbf{x}_i) \cdot \boldsymbol{\phi}(\mathbf{x}_j) \right\rangle = \sum_{t=1}^n \lambda_t v_{ti} v_{tj} = \left(\mathbf{V}\boldsymbol{\Lambda}\mathbf{V}'\right)_{ij} = \mathbf{K}_{ij} = K(\mathbf{x}_i,\mathbf{x}_j),$$

implying that $K(\mathbf{x},\mathbf{z})$ is indeed a kernel function corresponding to the feature mapping $\boldsymbol{\phi}$. The requirement that the eigenvalues of $\mathbf{K}$ be non-negative is necessary since if we have a negative eigenvalue λ_s with eigenvector $\mathbf{v}_s$, the point

$$\mathbf{z} = \sum_{i=1}^n v_{si} \boldsymbol{\phi}(\mathbf{x}_i) = \sqrt{\boldsymbol{\Lambda}}\mathbf{V}'\mathbf{v}_s$$

in the feature space would have norm squared

$$\|\mathbf{z}\|^2 = \langle \mathbf{z} \cdot \mathbf{z} \rangle = \mathbf{v}_s'\mathbf{V}\sqrt{\boldsymbol{\Lambda}}\sqrt{\boldsymbol{\Lambda}}\mathbf{V}'\mathbf{v}_s = \mathbf{v}_s'\mathbf{V}\boldsymbol{\Lambda}\mathbf{V}'\mathbf{v}_s = \mathbf{v}_s'\mathbf{K}\mathbf{v}_s = \lambda_s < 0,$$

contradicting the geometry of that space. Hence, we have proved by contradiction the following proposition.

Proposition 3.5 *Let X be a finite input space with $K(\mathbf{x},\mathbf{z})$ a symmetric function on X. Then $K(\mathbf{x},\mathbf{z})$ is a kernel function if and only if the matrix*

$$\mathbf{K} = \left(K(\mathbf{x}_i,\mathbf{x}_j)\right)_{i,j=1}^n,$$

is positive semi-definite (has non-negative eigenvalues).

https://doi.org/10.1017/CBO9780511801389.005 Published online by Cambridge University Press

Motivated by this simple example, we will allow a slight generalisation of an inner product in a Hilbert space by introducing a weighting λ_i for each dimension,

$$\langle \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{z}) \rangle = \sum_{i=1}^{\infty} \lambda_i \phi_i(\mathbf{x}) \phi_i(\mathbf{z}) = K(\mathbf{x}, \mathbf{z}),$$

so that the feature vector becomes

$$\boldsymbol{\phi}(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \dots, \phi_n(\mathbf{x}), \dots).$$

Mercer's theorem gives necessary and sufficient conditions for a continuous symmetric function $K(\mathbf{x}, \mathbf{z})$ to admit such a representation

$$K(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{\infty} \lambda_i \phi_i(\mathbf{x}) \phi_i(\mathbf{z}),$$

with non-negative λ_i, which is equivalent to $K(\mathbf{x}, \mathbf{z})$ being an inner product in the feature space $F \supseteq \boldsymbol{\phi}(X)$, where F is the l_2 space of all sequences

$$\boldsymbol{\psi} = (\psi_1, \psi_2, \dots, \psi_i, \dots)$$

for which

$$\sum_{i=1}^{\infty} \lambda_i \psi_i^2 < \infty.$$

This will implicitly induce a space defined by the feature vector and as a consequence a linear function in F like those described in Chapter 2 will be represented by

$$f(\mathbf{x}) = \sum_{i=1}^{\infty} \lambda_i \psi_i \phi_i(\mathbf{x}) + b = \sum_{j=1}^{\ell} \alpha_j y_j K(\mathbf{x}, \mathbf{x}_j) + b,$$

where the first expression is the primal representation of the function, and the second is the dual, the relation between the two being given by

$$\boldsymbol{\psi} = \sum_{j=1}^{\ell} \alpha_j y_j \boldsymbol{\phi}(\mathbf{x}_j).$$

Note that in the primal representation the number of terms in the summation is equal to the dimensionality of the feature space, while in the dual there are ℓ terms (the sample size). According to the size of the feature space being considered, one or other of the representations can be more convenient. It is clear from its form that this function is non-linear whenever the kernel function is non-linear.

The analogy with the finite case is very close. The contribution from functional analysis comes from studying the eigenvalue problem for integral equations of the form

$$\int_X K(\mathbf{x}, \mathbf{z})\phi(\mathbf{z})d\mathbf{z} = \lambda\phi(\mathbf{x}),$$

where $K(\mathbf{x}, \mathbf{z})$ is a bounded, symmetric, and positive kernel function and X is a compact space. We will not go into the details of the analysis but simply quote the theorem (see Appendix B.3 for the notation and definitions).

Theorem 3.6 *(Mercer) Let X be a compact subset of $\mathbb{R}^n$. Suppose K is a continuous symmetric function such that the integral operator $T_K : L_2(X) \to L_2(X)$,*

$$(T_K f)(\cdot) = \int_X K(\cdot, \mathbf{x})f(\mathbf{x})d\mathbf{x},$$

is positive, that is

$$\int_{X\times X} K(\mathbf{x}, \mathbf{z})f(\mathbf{x})f(\mathbf{z})d\mathbf{x}d\mathbf{z} \geq 0,$$

for all $f \in L_2(X)$. Then we can expand $K(\mathbf{x}, \mathbf{z})$ in a uniformly convergent series (on $X \times X$) in terms of T_K's eigen-functions $\phi_j \in L_2(X)$, normalised in such a way that $\|\phi_j\|_{L_2} = 1$, and positive associated eigenvalues $\lambda_j \geq 0$,

$$K(\mathbf{x}, \mathbf{z}) = \sum_{j=1}^{\infty} \lambda_j \phi_j(\mathbf{x})\phi_j(\mathbf{z}).$$

Remark 3.7 The positivity condition

$$\int_{X\times X} K(\mathbf{x}, \mathbf{z})f(\mathbf{x})f(\mathbf{z})d\mathbf{x}d\mathbf{z} \geq 0, \forall f \in L_2(X),$$

corresponds to the positive semi-definite condition in the finite case. The finite matrix condition on a set of points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ is recovered by choosing f to be the weighted sums of delta functions at each $\mathbf{x}_i$. Since such functions are limits of functions in $L_2(X)$, the above condition implies that for any finite subset of X the corresponding matrix is positive semi-definite. The converse is also true since if the positivity condition does not hold for some function f, we can approximate the integral with a finite sum over a mesh of inputs, which if chosen sufficiently finely will still give a negative value. The values of f on the chosen mesh $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ form a vector $\mathbf{v}$, for which the corresponding kernel matrix $\mathbf{K}$ satisfies

$$\mathbf{v}'\mathbf{K}\mathbf{v} < 0,$$

showing that $\mathbf{K}$ fails to be positive semi-definite. The conditions for Mercer's theorem are therefore equivalent to requiring that for any finite subset of X, the

corresponding matrix is positive semi-definite. This gives a second characterisation of a kernel function, one that will prove most useful when we come to constructing kernels. We use the term kernel to refer to functions satisfying this property, but in the literature these are often called Mercer kernels.

Remark 3.8 Following the relaxing of the definition of inner product after Proposition 3.5, the theorem suggests the feature mapping

$$\mathbf{x} = (x_1, \ldots, x_n) \longmapsto \boldsymbol{\phi}(\mathbf{x}) = (\phi_1(\mathbf{x}), \ldots, \phi_j(\mathbf{x}), \ldots),$$

into the Hilbert space defined by the weighted inner product given by

$$\left\langle \boldsymbol{\psi} \cdot \tilde{\boldsymbol{\psi}} \right\rangle = \sum_{j=1}^{\infty} \lambda_j \psi_j \tilde{\psi}_j,$$

since the inner product of two feature vectors then satisfies

$$\langle \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{z}) \rangle = \sum_{j=1}^{\infty} \lambda_j \phi_j(\mathbf{x}) \phi_j(\mathbf{z}) = K(\mathbf{x}, \mathbf{z}),$$

These features have the special property that they are orthonormal functions in $L_2(X)$. This choice of mapping is sometimes referred to as *Mercer features*. The compactness of the input domain is required in order to ensure that the spectrum is a countable set. Note that we do not need the features to form an orthonormal set. In the finite input space example given above they have been rescaled by the square root of the eigenvalues. In general we can rescale each coordinate,

$$\mathbf{x} = (x_1, \ldots, x_n) \longmapsto \boldsymbol{\phi}(\mathbf{x}) = (b_1 \phi_1(\mathbf{x}), \ldots, b_j \phi_j(\mathbf{x}), \ldots),$$

into the Hilbert space defined by the weighted inner product given by

$$\left\langle \boldsymbol{\psi} \cdot \tilde{\boldsymbol{\psi}} \right\rangle = \sum_{j=1}^{\infty} \frac{\lambda_j}{b_j^2} \psi_j \tilde{\psi}_j,$$

since again the inner product of two feature vectors then satisfies

$$\langle \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{z}) \rangle = \sum_{j=1}^{\infty} \frac{\lambda_j}{b_j^2} b_j \phi_j(\mathbf{x}) b_j \phi_j(\mathbf{z}) = K(\mathbf{x}, \mathbf{z}).$$

In this case and in the finite input space example the features are still orthogonal. Orthogonality is, however, also not required. For instance in the case of polynomial kernels, the features given in Example 3.3 will not in general be orthogonal. The features could be chosen to be orthogonal, but the particular polynomials needed would depend on the subset of $\mathbb{R}^2$ taken as the input domain and the measure of integration chosen.

Mercer features provide a representation of the input points by means of their image in the feature space with the inner product defined by the potentially infinite number of eigenvalues of the kernel. The sub-manifold formed by the image of the input space is defined by the eigenvectors of the kernel operator. Although the images of the points $\phi(\mathbf{x}) \in F$ will not be computed, their inner products can be calculated using the kernel function.

Example 3.9 Consider the kernel function $K(\mathbf{x}, \mathbf{z}) = K(\mathbf{x} - \mathbf{z})$. Such a kernel is said to be translation invariant, since the inner product of two inputs is unchanged if both are translated by the same vector. Consider the one dimensional case in which K is defined on the interval $[0, 2\pi]$ in such a way that $K(u)$ can be extended to a continuous, symmetric, periodic function on $\mathbb{R}$. Such a function can be expanded in a uniformly convergent Fourier series:

$$K(u) = \sum_{n=0}^{\infty} a_n \cos(nu).$$

In this case we can expand $K(x - z)$ as follows:

$$K(x - z) = a_0 + \sum_{n=1}^{\infty} a_n \sin(nx)\sin(nz) + \sum_{n=1}^{\infty} a_n \cos(nx)\cos(nz).$$

Provided the a_n are all positive this shows $K(x, z)$ as the inner product in the feature space defined by the orthogonal features

$$\{\phi_i(x)\}_{i=0}^{\infty} = (1, \sin(x), \cos(x), \sin(2x), \cos(2x), \dots, \sin(nx), \cos(nx), \dots),$$

since the functions, 1, $\cos(nu)$ and $\sin(nu)$ form a set of orthogonal functions on the interval $[0, 2\pi]$. Hence, normalising them will provide a set of Mercer features. Note that the embedding is defined independently of the a_n, which subsequently control the geometry of the feature space.

Example 3.9 provides some useful insight into the role that the choice of kernel can play. The parameters a_n in the expansion of $K(u)$ are its Fourier coefficients. If for some n, we have $a_n = 0$, the corresponding features are removed from the feature space. Similarly, small values of a_n mean that the feature is given low weighting and so will have less influence on the choice of hyperplane. Hence, the choice of kernel can be seen as choosing a filter with a particular spectral characteristic, the effect of which is to control the influence of the different frequencies in determining the optimal separation. In the next subsection we introduce a view of the feature space that will elucidate its role in encoding a learning bias.

Given a feature space representation with countably many linearly independent features (not necessarily orthogonal or of unit norm) given by the mapping

$$\mathbf{x} = (x_1, \dots, x_n) \longmapsto \phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \dots, \phi_j(\mathbf{x}), \dots),$$

into the l_2 space F defined by the weighted inner product

$$\left\langle \psi \cdot \tilde{\psi} \right\rangle = \sum_{j=1}^{\infty} \mu_j \psi_j \tilde{\psi}_j,$$

we can define a space of functions $\mathscr{H}$ on the input space to be the image of F under the mapping

$$T : \psi \longmapsto \sum_{j=1}^{\infty} \psi_j \phi_j(\mathbf{x}). \tag{3.1}$$

Note that if F is finite dimensional, $\mathscr{H}$ is the function class on the input space we are effectively using by applying linear functions in the feature space since it is the set of all linear combinations of the basis functions. For infinite feature spaces, $\mathscr{H}$ may not contain all the possible hypothesis functions, as they may be images of points that do not have a finite norm in F, or equally $\mathscr{H}$ may have too many functions. In the next subsection we consider a particular choice of the feature mapping that ensures $\mathscr{H}$ does contain exactly the set of hypotheses and at the same time has a number of additional special properties.

Reproducing Kernel Hilbert Spaces

Assume we have a feature space given by the map

$$\mathbf{x} = (x_1, \ldots, x_n) \longmapsto \boldsymbol{\phi}(\mathbf{x}) = (\phi_1(\mathbf{x}), \ldots, \phi_j(\mathbf{x}), \ldots),$$

into the l_2 space F defined by the weighted inner product

$$\left\langle \psi \cdot \tilde{\psi} \right\rangle = \sum_{j=1}^{\infty} \mu_j \psi_j \tilde{\psi}_j,$$

where

$$K(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{\infty} \mu_i \phi_i(\mathbf{x}) \phi_i(\mathbf{z}).$$

Now consider introducing a weighting proportional to the factor μ_i, so that the image has the form

$$\mathbf{x} = (x_1, \ldots, x_n) \longmapsto \boldsymbol{\psi}(\mathbf{x}) = (\psi_1(\mathbf{x}), \ldots, \psi_j(\mathbf{x}), \ldots) = (\mu_1 \phi_1(\mathbf{x}), \ldots, \mu_j \phi_j(\mathbf{x}), \ldots).$$

The appropriate weighted inner product is then given by

$$\left\langle \psi \cdot \tilde{\psi} \right\rangle_{\mu} = \sum_{j=1}^{\infty} \frac{\psi_j \tilde{\psi}_j}{\mu_j},$$

https://doi.org/10.1017/CBO9780511801389.005 Published online by Cambridge University Press

so that

$$\|\psi(\mathbf{x})\|_\mu^2 = K(\mathbf{x}, \mathbf{x}).$$

We denote with $\mathscr{H}$ the image of F under the mapping T defined by equation (3.1). This particular weighting has a number of special properties. For two functions

$$f(\mathbf{x}) = \sum_{j=1}^{\infty} \psi_j \phi_j(\mathbf{x}) \text{ and } g(\mathbf{x}) = \sum_{j=1}^{\infty} \tilde{\psi}_j \phi_j(\mathbf{x}),$$

we define an inner product in $\mathscr{H}$ by

$$\langle f(\cdot) \cdot g(\cdot) \rangle_{\mathscr{H}} = \sum_{j=1}^{\infty} \frac{\psi_j \tilde{\psi}_j}{\mu_j}$$

hence making the map T an isometry. First observe that if we map an input point to the feature space and then apply T to its image, we obtain

$$T(\psi(\mathbf{z})) = \sum_{j=1}^{\infty} \psi_j(\mathbf{z}) \phi_j(\mathbf{x}) = \sum_{j=1}^{\infty} \mu_j \phi_j(\mathbf{z}) \phi_j(\mathbf{x}) = K(\mathbf{z}, \mathbf{x}),$$

so that $K(\mathbf{z}, \cdot) \in \mathscr{H}$. As a consequence functions in the dual representation

$$f(\mathbf{x}) = \sum_{i=1}^{\ell} \alpha_i K(\mathbf{x}_i, \mathbf{x})$$

are in the space $\mathscr{H}$. Furthermore, if we take the inner product of a general function $f(\mathbf{x}) = \sum_{j=1}^{\infty} a_j \phi_j(\mathbf{x}) \in \mathscr{H}$ with $K(\mathbf{z}, \mathbf{x})$, we obtain

$$\begin{aligned} \langle f(\cdot) \cdot K(\mathbf{z}, \cdot) \rangle_{\mathscr{H}} &= \left\langle (a_j)_{j=1}^{\infty} \cdot (\mu_j \phi_j(\mathbf{z}))_{j=1}^{\infty} \right\rangle_\mu \\ &= \sum_{j=1}^{\infty} \frac{a_j \mu_j \phi_j(\mathbf{z})}{\mu_j} = \sum_{j=1}^{\infty} a_j \phi_j(\mathbf{z}) = f(\mathbf{z}), \end{aligned}$$

known as the reproducing property of the kernel K. This also implies that $\mathscr{H}$ coincides with the closure of the subspace

$$\mathscr{K} = \left\{ \sum_{i=1}^{\ell} \alpha_i K(\mathbf{x}_i, \mathbf{x}) : \ell \in \mathbb{N},\ (\mathbf{x}_1, \ldots, \mathbf{x}_\ell) \in X^\ell,\ \alpha_i \in \mathbb{R} \right\},$$

since if $f \in \mathscr{H}$ satisfies $f \perp \mathscr{K}$ then for all $\mathbf{z} \in X$,

$$f(\mathbf{z}) = \langle f(\cdot) \cdot K(\mathbf{z}, \cdot) \rangle_{\mathscr{H}} = 0,$$

implying $f = 0$. Hence $\mathscr{H}$ is contained in the closure of $\mathscr{K}$ and so does not contain functions that cannot be arbitrarily well approximated in the dual representation. For two functions $f(\mathbf{x}) = \sum_{i=1}^{\ell} \alpha_i K(\mathbf{x}_i, \mathbf{x})$ and $g(\mathbf{x}) = \sum_{j=1}^{\hat{\ell}} \hat{\alpha}_j K(\hat{\mathbf{x}}_j, \mathbf{x})$, in the dual representation the inner product is given by

$$\begin{aligned}
\langle f(\cdot) \cdot g(\cdot) \rangle_{\mathscr{H}} &= \left\langle \sum_{i=1}^{\ell} \alpha_i K(\mathbf{x}_i, \cdot) \cdot \sum_{j=1}^{\hat{\ell}} \hat{\alpha}_j K(\hat{\mathbf{x}}_j, \cdot) \right\rangle_{\mathscr{H}} \\
&= \sum_{i=1}^{\ell} \alpha_i \sum_{j=1}^{\hat{\ell}} \hat{\alpha}_j \langle K(\mathbf{x}_i, \cdot) \cdot K(\hat{\mathbf{x}}_j, \cdot) \rangle_{\mathscr{H}} \\
&= \sum_{i=1}^{\ell} \alpha_i \sum_{j=1}^{\hat{\ell}} \hat{\alpha}_j K(\mathbf{x}_i, \hat{\mathbf{x}}_j) \\
&= \sum_{i=1}^{\ell} \alpha_i g(\mathbf{x}_i) = \sum_{j=1}^{\hat{\ell}} \hat{\alpha}_j f(\hat{\mathbf{x}}_j), \qquad (3.2)
\end{aligned}$$

showing that the definition of the inner product is independent of the particular representation of the function (changing the representation of g does not change the value of $g(\mathbf{x}_i)$). In addition we also obtain that $\|f\|_{\mathscr{H}}^2 = \sum_{i=1}^{\ell} \alpha_i f(\mathbf{x}_i)$, showing that for f to have a bounded norm, it must have bounded value and coefficients. Finally, note that the reproducing property $\langle f(\cdot) \cdot K(\mathbf{z}, \cdot) \rangle_{\mathscr{H}} = f(\mathbf{z})$ implies that the evaluation functionals defined by $F_{\mathbf{x}}[f] = f(\mathbf{x})$ $(\forall f \in \mathscr{H})$ are linear and bounded, that is there exist $U_{\mathbf{z}} = \|K(\mathbf{z}, \cdot)\|_{\mathscr{H}} \in \mathbb{R}^+$ such that by the Cauchy-Schwarz inequality

$$|F_{\mathbf{z}}[f]| = |f(\mathbf{z})| = \langle f(\cdot) \cdot K(\mathbf{z}, \cdot) \rangle_{\mathscr{H}} \leq U_{\mathbf{z}} \|f\|_{\mathscr{H}}$$

for all $f \in \mathscr{H}$.

For a Hilbert space $\mathscr{H}$ of functions defined over the input domain $X \subset \mathbb{R}^d$, the bounded linearity of the evaluation functionals is the defining property for a *reproducing kernel Hilbert space (RKHS)*. Hence, we have demonstrated the following result.

Theorem 3.10 *For every Mercer kernel $K(\mathbf{x}, \mathbf{z})$ defined over the domain $X \subset \mathbb{R}^d$, there exists an RKHS $\mathscr{H}$ of functions defined over X for which K is the reproducing kernel.*

Remarkably the converse of this theorem also holds. That is, for any Hilbert space of functions in which the evaluation functionals are bounded and linear, there exists a reproducing kernel function. That a reproducing kernel is also a

Mercer kernel follows from the fact that for $f(\mathbf{x}) = \sum_{i=1}^{\ell} \alpha_i K(\mathbf{x}_i, \mathbf{x})$,

$$\begin{aligned} 0 \leq \|f\|_{\mathcal{H}}^2 &= \left\langle \sum_{i=1}^{\ell} \alpha_i K(\mathbf{x}_i, \cdot) \cdot \sum_{i=1}^{\ell} \alpha_i K(\mathbf{x}_i, \cdot) \right\rangle_{\mathcal{H}} \\ &= \sum_{i=1}^{\ell} \alpha_i \sum_{j=1}^{\ell} \alpha_j \langle K(\mathbf{x}_i, \cdot) \cdot K(\mathbf{x}_j, \cdot) \rangle_{\mathcal{H}} \\ &= \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j), \end{aligned}$$

implying K is positive semi-definite for every finite subset of X, sufficient to imply the positivity of the operator K by Remark 3.7.

The following example suggests some of the power and insight that the RKHS construction affords.

Example 3.11 Suppose we wish to perform regression based on as set of training points

$$S = ((\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_\ell, y_\ell)) \subset (X \times Y)^\ell \subset (\mathbb{R}^n \times \mathbb{R})^\ell,$$

generated from the target function $t(\mathbf{x})$. If we assume a dual representation of the form

$$f(\mathbf{x}) = \sum_{i=1}^{\ell} \alpha_i K(\mathbf{x}_i, \mathbf{x}),$$

we can seek to minimise the norm

$$\begin{aligned} \|f - t\|_{\mathcal{H}}^2 &= \left\langle \sum_{i=1}^{\ell} \alpha_i K(\mathbf{x}_i, \mathbf{x}) - t(\mathbf{x}) \cdot \sum_{i=1}^{\ell} \alpha_i K(\mathbf{x}_i, \mathbf{x}) - t(\mathbf{x}) \right\rangle_{\mathcal{H}} \\ &= -2 \left\langle t(\mathbf{x}) \cdot \sum_{i=1}^{\ell} \alpha_i K(\mathbf{x}_i, \mathbf{x}) \right\rangle_{\mathcal{H}} + \|f\|_{\mathcal{H}}^2 + \|t\|_{\mathcal{H}}^2 \\ &= -2 \sum_{i=1}^{\ell} \alpha_i \langle t(\mathbf{x}) \cdot K(\mathbf{x}_i, \mathbf{x}) \rangle_{\mathcal{H}} + \|f\|_{\mathcal{H}}^2 + \|t\|_{\mathcal{H}}^2 \\ &= -2 \sum_{i=1}^{\ell} \alpha_i y_i + \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) + \|t\|_{\mathcal{H}}^2, \end{aligned}$$

which can be solved by setting the derivatives with respect to the parameters equal to zero, since $\|t\|_{\mathcal{H}}^2$ does not depend on them. We will discuss this further in Chapter 5 in Example 5.6. Notice again how the Gram matrix in the feature space appears if we view K as a Mercer kernel.

https://doi.org/10.1017/CBO9780511801389.005 Published online by Cambridge University Press

3.3.2 Making Kernels from Kernels

The key to verifying that a new symmetric function is a kernel will be the conditions outlined in Remark 3.7, that is the requirement that the matrix defined by restricting the function to any finite set of points is positive semi-definite. We apply this criterion to confirm that a number of new kernels can be created. The following proposition can be viewed as showing that kernels satisfy a number of closure properties, allowing us to create more complicated kernels from simple building blocks.

Proposition 3.12 *Let K_1 and K_2 be kernels over $X \times X$, $X \subseteq \mathbb{R}^n$, $a \in \mathbb{R}^+$, $f(\cdot)$ a real-valued function on X,*

$$\phi : X \longrightarrow \mathbb{R}^m$$

with K_3 a kernel over $\mathbb{R}^m \times \mathbb{R}^m$, and $\mathbf{B}$ a symmetric positive semi-definite $n \times n$ matrix. Then the following functions are kernels:

1. $K(\mathbf{x}, \mathbf{z}) = K_1(\mathbf{x}, \mathbf{z}) + K_2(\mathbf{x}, \mathbf{z})$,
2. $K(\mathbf{x}, \mathbf{z}) = aK_1(\mathbf{x}, \mathbf{z})$,
3. $K(\mathbf{x}, \mathbf{z}) = K_1(\mathbf{x}, \mathbf{z})K_2(\mathbf{x}, \mathbf{z})$,
4. $K(\mathbf{x}, \mathbf{z}) = f(\mathbf{x})f(\mathbf{z})$,
5. $K(\mathbf{x}, \mathbf{z}) = K_3(\phi(\mathbf{x}), \phi(\mathbf{z}))$,
6. $K(\mathbf{x}, \mathbf{z}) = \mathbf{x}'\mathbf{B}\mathbf{z}$.

Proof Fix a finite set of points $\{\mathbf{x}_1, \ldots, \mathbf{x}_\ell\}$, and let $\mathbf{K}_1$ and $\mathbf{K}_2$, be the corresponding matrices obtained by restricting K_1 and K_2 to these points. Consider any vector $\boldsymbol{\alpha} \in \mathbb{R}^\ell$. Recall that a matrix $\mathbf{K}$ is positive semi-definite if and only if $\boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha} \geq 0$, for all $\boldsymbol{\alpha}$.

1. We have

$$\boldsymbol{\alpha}'(\mathbf{K}_1 + \mathbf{K}_2)\boldsymbol{\alpha} = \boldsymbol{\alpha}'\mathbf{K}_1\boldsymbol{\alpha} + \boldsymbol{\alpha}'\mathbf{K}_2\boldsymbol{\alpha} \geq 0,$$

   and so $\mathbf{K}_1 + \mathbf{K}_2$ is positive semi-definite and $K_1 + K_2$ a kernel function.

2. Similarly $\boldsymbol{\alpha}' a\mathbf{K}_1\boldsymbol{\alpha} = a\boldsymbol{\alpha}'\mathbf{K}_1\boldsymbol{\alpha} \geq 0$, verifying that aK_1 is a kernel.

3. Let

$$\mathbf{K} = \mathbf{K}_1 \bigotimes \mathbf{K}_2$$

   be the tensor product of the matrices $\mathbf{K}_1$ and $\mathbf{K}_2$. The tensor product of two positive semi-definite matrices is positive semi-definite since the eigenvalues of the product are all pairs of products of the eigenvalues of the two components. The matrix corresponding to the function K_1K_2 is

known as the Schur product $\mathbf{H}$ of $\mathbf{K}_1$ and $\mathbf{K}_2$ with entries the products of the corresponding entries in the two components. The matrix $\mathbf{H}$ is a principal submatrix of $\mathbf{K}$ defined by a set of columns and the same set of rows. Hence for any $\boldsymbol{\alpha} \in \mathbb{R}^\ell$, there is a corresponding $\boldsymbol{\alpha}_1 \in \mathbb{R}^{\ell^2}$, such that

$$\boldsymbol{\alpha}'\mathbf{H}\boldsymbol{\alpha} = \boldsymbol{\alpha}_1'\mathbf{K}\boldsymbol{\alpha}_1 \geq 0,$$

and so $\mathbf{H}$ is positive semi-definite as required.

4. We can rearrange the bilinear form as follows:

$$\begin{aligned}\sum_{i=1}^{\ell}\sum_{j=1}^{\ell}\alpha_i\alpha_j K(\mathbf{x}_i,\mathbf{x}_j) &= \sum_{i=1}^{\ell}\sum_{j=1}^{\ell}\alpha_i\alpha_j f(\mathbf{x}_i)f(\mathbf{x}_j)\\ &= \sum_{i=1}^{\ell}\alpha_i f(\mathbf{x}_i)\sum_{j=1}^{\ell}\alpha_j f(\mathbf{x}_j)\\ &= \left(\sum_{i=1}^{\ell}\alpha_i f(\mathbf{x}_i)\right)^2 \geq 0,\end{aligned}$$

as required.

5. Since K_3 is a kernel, the matrix obtained by restricting K_3 to the points $\boldsymbol{\phi}(\mathbf{x}_1),\ldots,\boldsymbol{\phi}(\mathbf{x}_\ell)$ is positive semi-definite as required.

6. Consider the diagonalisation of $\mathbf{B} = \mathbf{V}'\Lambda\mathbf{V}$ by an orthogonal matrix $\mathbf{V}$, where Λ is the diagonal matrix containing the non-negative eigenvalues. Let $\sqrt{\Lambda}$ be the diagonal matrix with the square roots of the eigenvalues and set $\mathbf{A} = \sqrt{\Lambda}\mathbf{V}$. We therefore have

$$K(\mathbf{x},\mathbf{z}) = \mathbf{x}'\mathbf{B}\mathbf{z} = \mathbf{x}'\mathbf{V}'\Lambda\mathbf{V}\mathbf{z} = \mathbf{x}'\mathbf{V}'\sqrt{\Lambda}\sqrt{\Lambda}\mathbf{V}\mathbf{z} = \mathbf{x}'\mathbf{A}'\mathbf{A}\mathbf{z} = \langle \mathbf{A}\mathbf{x}\cdot\mathbf{A}\mathbf{z}\rangle,$$

the inner product using the feature mapping $\mathbf{A}$.

□

Corollary 3.13 Let $K_1(\mathbf{x},\mathbf{z})$ be a kernel over $X \times X$, $\mathbf{x}, \mathbf{z} \in X$, and $p(x)$ a polynomial with positive coefficients. Then the following functions are also kernels:

1. $K(\mathbf{x},\mathbf{z}) = p(K_1(\mathbf{x},\mathbf{z}))$,
2. $K(\mathbf{x},\mathbf{z}) = \exp(K(\mathbf{x},\mathbf{z}))$,
3. $K(\mathbf{x},\mathbf{z}) = \exp(-\|\mathbf{x}-\mathbf{z}\|^2/\sigma^2)$.

Proof We take the three parts in turn:

1. For a polynomial the result is immediate by combining the parts of the proposition. Note that the constant is covered by part 4 of the proposition.

https://doi.org/10.1017/CBO9780511801389.005 Published online by Cambridge University Press

2. The exponential function can be arbitrarily closely approximated by polynomials with positive coefficients and hence is a limit of kernels. Since kernels are clearly closed under taking pointwise limits, the result follows.

3. We can decompose the Gaussian function as follows:

$$\exp(-\|\mathbf{x}-\mathbf{z}\|^2/\sigma^2) = \exp(-\|\mathbf{x}\|^2/\sigma^2)\exp(-\|\mathbf{z}\|^2/\sigma^2)\exp(2\langle \mathbf{x}\cdot\mathbf{z}\rangle/\sigma^2).$$

The first two factors together form a kernel by part 4 of the proposition, while the third factor is a kernel by part 2 of this corollary.

□

Remark 3.14 The final kernel of the corollary is known as the Gaussian kernel. This function forms the core of a radial basis function network and hence, using this kernel will mean the hypotheses are radial basis function networks.

3.3.3 Making Kernels from Features

Another way to obtain a kernel is of course to start from the features, and to obtain it by working out their inner product. In this case there is no need to check for positive semi-definiteness since this will follow automatically from the definition as an inner product. The first example we gave of polynomial kernels followed this route. We now give a more unusual example of a kernel over a discrete space, in this case finite strings in order to illustrate the potential of the method for non-Euclidean spaces.

Example 3.15 *(String subsequence kernels)* Let Σ be a finite alphabet. A string is a finite sequence of characters from Σ, including the empty sequence. For strings s, t, we denote by $|s|$ the length of the string $s = s_1 \dots s_{|s|}$, and by st the string obtained by concatenating the strings s and t. The string $s[i:j]$ is the substring $s_i \dots s_j$ of s. We say that u is a subsequence of s, if there exist indices $\mathbf{i} = (i_1, \dots, i_{|u|})$, with $1 \le i_1 < \dots < i_{|u|} \le |s|$, such that $u_j = s_{i_j}$, for $j = 1, \dots, |u|$, or $u = s[\mathbf{i}]$ for short. The length $l(\mathbf{i})$ of the subsequence in s is $i_{|u|} - i_1 + 1$. We denote by Σ^n the set of all finite strings of length n, and by Σ^* the set of all strings

$$\Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n.$$

We now define feature spaces $F_n = \mathbb{R}^{\Sigma^n}$. The feature mapping $\boldsymbol{\phi}$ for a string s is given by defining the u coordinate $\phi_u(s)$ for each $u \in \Sigma^n$. We define

$$\phi_u(s) = \sum_{\mathbf{i}:u=s[\mathbf{i}]} \lambda^{l(\mathbf{i})},$$

https://doi.org/10.1017/CBO9780511801389.005 Published online by Cambridge University Press

for some $\lambda \leq 1$. These features measure the number of occurrences of subsequences in the string s weighting them according to their lengths. Hence, the inner product of the feature vectors for two strings s and t give a sum over all common subsequences weighted according to their frequency of occurrence and lengths

$$\begin{aligned} K_n(s,t) &= \sum_{u\in\Sigma^n} \langle \phi_u(s) \cdot \phi_u(t) \rangle \\ &= \sum_{u\in\Sigma^n} \sum_{\mathbf{i}:u=s[\mathbf{i}]} \lambda^{l(\mathbf{i})} \sum_{\mathbf{j}:u=t[\mathbf{j}]} \lambda^{l(\mathbf{j})} \\ &= \sum_{u\in\Sigma^n} \sum_{\mathbf{i}:u=s[\mathbf{i}]} \sum_{\mathbf{j}:u=t[\mathbf{j}]} \lambda^{l(\mathbf{i})+l(\mathbf{j})}. \end{aligned}$$

Such a kernel could clearly be very useful for text classification, but appears at first sight too expensive to compute for practical purposes. We introduce an additional function which will aid in defining a recursive computation for this kernel. Let

$$K'_i(s,t) = \sum_{u\in\Sigma^i} \sum_{\mathbf{i}:u=s[\mathbf{i}]} \sum_{\mathbf{j}:u=t[\mathbf{j}]} \lambda^{|s|+|t|-i_1-j_1+2}, \; i = 1,\ldots,n-1,$$

that is counting the length to the end of the strings s and t instead of just $l(\mathbf{i})$ and $l(\mathbf{j})$. We can now define a recursive computation for K'_i and hence compute K_n,

$$\begin{aligned} K'_0(s,t) &= 1, \text{ for all } s,t, \\ K'_i(s,t) &= 0, \text{ if } \min\left(|s|,|t|\right) < i, \\ K_i(s,t) &= 0, \text{ if } \min\left(|s|,|t|\right) < i, \\ K'_i(sx,t) &= \lambda K'_i(s,t) + \sum_{j:t_j=x} K'_{i-1}(s,t[1:j-1])\lambda^{|t|-j+2}, \; i = 1,\ldots,n-1, \\ K_n(sx,t) &= K_n(s,t) + \sum_{j:t_j=x} K'_{n-1}(s,t[1:j-1])\lambda^2. \end{aligned}$$

The correctness of this recursion follows from observing how the length of the strings has increased, incurring a factor of λ for each extra character, until the full length of n characters has been attained. It is quite surprising that the recursive formulation is sufficient to compute the kernel in time proportional to $n|s||t|$. If we wished to compute $K_n(s,t)$ for a range of values of n, we would simply perform the computation of $K'_i(s,t)$ up to one less than the largest n required, and then apply the last recursion for each $K_n(s,t)$ that is needed using the stored values of $K'_i(s,t)$. We can of course use parts 1 and 2 of Proposition 3.12 to create a kernel $K(s,t)$ that combines the different $K_n(s,t)$ giving different weightings for each n.

3.4 Working in Feature Space

Since the embedding given by the feature mapping

$$\mathbf{x} = (x_1, \ldots, x_n) \longmapsto \boldsymbol{\phi}(\mathbf{x}) = (\phi_1(\mathbf{x}), \ldots, \phi_j(\mathbf{x}), \ldots)$$

is non-linear and hence in general defines an n dimensional sub-manifold of the feature space, linear combinations of images often do not correspond to images of any input point. Nevertheless, we are able to represent such points using the dual representation, and provided we are only concerned with distances and inner products the kernel can be used without computing an explicit feature vector. In this section, we briefly review how such calculations can be performed. Let $\boldsymbol{\phi}(X)$ be the image of the input space under the feature mapping, and define $F = \mathrm{co}(\boldsymbol{\phi}(X))$ to be the space of all finite linear combinations of points from $\boldsymbol{\phi}(X)$. We represent a generic point

$$P = \sum_{i=1}^{\ell} \alpha_i \boldsymbol{\phi}(\mathbf{x}_i)$$

in F, by a sequence of pairs

$$P = (\alpha_i, \mathbf{x}_i)_{i=1}^{\ell}.$$

Consider a second such point $Q = (\beta_i, \mathbf{z}_i)_{i=1}^{s}$. The sum and difference of two such points are defined by taking the union of the set of points and adding or subtracting corresponding coefficients. For example if $\mathbf{x}_i$, $i = 1, \ldots, \ell$, are a set of positive examples, and $\mathbf{z}_i$, $i = 1, \ldots, s$, a set of negative examples, then the weight vector of the hyperplane oriented from the centre of mass of the negative points towards the centre of mass of the positives is given by

$$\frac{1}{\ell}\sum_{i=1}^{\ell}(1, \mathbf{x}_i) - \frac{1}{s}\sum_{j=1}^{s}(1, \mathbf{z}_j) = \left(\frac{1}{\ell}, \mathbf{x}_i\right)_{i=1}^{\ell} \cup \left(-\frac{1}{s}, \mathbf{z}_j\right)_{j=1}^{s}.$$

The inner product with a second point $Q = (\beta_i, \mathbf{z}_i)_{i=1}^{s}$ is given by

$$\langle P \cdot Q \rangle_F = \sum_{i,j} \alpha_i \beta_j K(\mathbf{x}_i, \mathbf{z}_j).$$

Note that this corresponds exactly to the inner product $\langle f \cdot g \rangle_{\mathscr{H}}$ of the two functions

$$f(\mathbf{x}) = \sum_{i=1}^{\ell} \alpha_i K(\mathbf{x}_i, \mathbf{x}) \quad \text{and} \quad g(\mathbf{x}) = \sum_{i=1}^{s} \beta_i K(\mathbf{z}_i, \mathbf{x}),$$

in the corresponding RKHS $\mathscr{H}$. Hence, we can also view the sequences of pairs as representing functions in the RKHS with the inner product giving the corresponding RKHS inner product.

https://doi.org/10.1017/CBO9780511801389.005 Published online by Cambridge University Press

For a point $\mathbf{x} \in X$ the norm squared of the feature vector is $K(\mathbf{x}, \mathbf{x})$. Note that for Gaussian kernels, this is equal to 1 for all inputs, showing that the input space is embedded onto the surface of the unit ball in this case. Similarly, the square of the distance between P and Q can be computed as

$$\begin{aligned} \|P - Q\|_F^2 &= \langle P - Q \cdot P - Q \rangle_F \\ &= \sum_{i,j} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) - 2 \sum_{i,j} \alpha_i \beta_j K(\mathbf{x}_i, \mathbf{z}_j) + \sum_{i,j} \beta_i \beta_j K(\mathbf{z}_i, \mathbf{z}_j). \end{aligned}$$

As an example of how distances can be computed, the conditions that P must satisfy to be at the centre of the smallest sphere containing the images $(1, \mathbf{x}_i)$ of the points $\mathbf{x}_i$, $i = 1, \ldots, \ell$, are

$$\begin{aligned} P &= (\alpha_i, \mathbf{x}_i)_{i=1}^{\ell}, \text{ where} \\ \alpha &= \operatorname*{argmin}_{\alpha} \left(\sum_{i,j=1}^{\ell} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) + \max_{1 \le k \le \ell} \left(K(\mathbf{x}_k, \mathbf{x}_k) - 2 \sum_{i=1}^{\ell} \alpha_i K(\mathbf{x}_i, \mathbf{x}_k) \right) \right), \end{aligned}$$

since the squared distance from P to $(1, \mathbf{x}_k)$ is

$$\begin{aligned} \langle P - (1, \mathbf{x}_k) \cdot P - (1, \mathbf{x}_k) \rangle_F &= \sum_{i,j=1}^{\ell} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \\ &\quad - 2 \sum_{i=1}^{\ell} \alpha_i K(\mathbf{x}_i, \mathbf{x}_k) + K(\mathbf{x}_k, \mathbf{x}_k), \end{aligned}$$

and the first term on the right hand side is independent of k.

Following this type of approach it is also possible to perform principal components analysis in feature space using dual representations of the feature vectors.

Remark 3.16 There is an interesting view of the feature space, not necessary for the main development of the book, that sheds light on the structure of the image manifold of the input space. The distances we have considered above are all distances in the full feature space, which is potentially very high dimensional. The image $\phi(X)$ of the input space is a possibly contorted submanifold whose dimension is that of the input space. It is also possible to analyse distances within this image. The measurement of distance along the surface requires the use of a Riemannian metric defined by an appropriate tensor. Differential geometry studies the metric induced on a space by such a tensor. Metric tensors are symmetric and positive definite. Choosing a kernel induces a corresponding tensor g, which can be determined by computing the first three terms of a Taylor series for the squared distance between two points $\mathbf{x}$ and $\mathbf{z}$ as a function of the point $\mathbf{x} = \mathbf{z} + d\mathbf{x}$ on the surface. Since the first two terms are zero we obtain the

bilinear form or quadratic term

$$\begin{aligned} ds^2 &= \langle (1,\mathbf{x}) - (1,\mathbf{z}) \cdot (1,\mathbf{x}) - (1,\mathbf{z}) \rangle_F \\ &= K(\mathbf{x},\mathbf{x}) - 2K(\mathbf{x},\mathbf{z}) + K(\mathbf{z},\mathbf{z}) \\ &= \frac{1}{2}\sum_{i=1}^{\infty}\sum_{j=1}^{\infty}\left(\frac{\partial^2 K(\mathbf{x},\mathbf{x})}{\partial x_i \partial x_j}\right)_{\mathbf{x}=\mathbf{z}} dx_i dx_j - \sum_{i=1}^{\infty}\sum_{j=1}^{\infty}\left(\frac{\partial^2 K(\mathbf{x},\mathbf{z})}{\partial x_i \partial x_j}\right)_{\mathbf{x}=\mathbf{z}} dx_i dx_j \\ &= \sum_{i=1}^{\infty}\sum_{j=1}^{\infty}\left(\frac{1}{2}\frac{\partial^2 K(\mathbf{x},\mathbf{x})}{\partial x_i \partial x_j} - \frac{\partial^2 K(\mathbf{x},\mathbf{z})}{\partial x_i \partial x_j}\right)_{\mathbf{x}=\mathbf{z}} dx_i dx_j, \end{aligned}$$

giving the tensor $\mathbf{g}$ components

$$g_{ij}(\mathbf{z}) = \left(\frac{1}{2}\frac{\partial^2 K(\mathbf{x},\mathbf{x})}{\partial x_i \partial x_j} - \frac{\partial^2 K(\mathbf{x},\mathbf{z})}{\partial x_i \partial x_j}\right)_{\mathbf{x}=\mathbf{z}}.$$

Hence, the Riemannian metric tensor can also be computed from the kernel function and may give additional insight into the structure of the feature space and the sub-manifold corresponding to the image of the input space.

3.5 Kernels and Gaussian Processes

If we consider the output of a function $f(\mathbf{x})$, for fixed $\mathbf{x} \in X$, as f is chosen according to some distribution $\mathscr{D}$ defined over a class of real-valued functions $\mathscr{F}$, we may view the output value as a random variable, and hence

$$\{f(\mathbf{x}) : \mathbf{x} \in X\}$$

as a collection of potentially correlated random variables. Such a collection is known as a stochastic process. The distribution over the function class $\mathscr{F}$ can be regarded as our prior belief in the likelihood that the different functions will provide the solution to our learning problem. Such a prior is characteristic of a Bayesian perspective on learning. We will return to discuss this approach further in the next chapter and in Chapter 6 discuss how to make predictions using Gaussian processes. At this point we wish to highlight the connection between a particular form of prior commonly used in Bayesian learning and the kernel functions we have introduced for Support Vector Machines. Many of the computations required by a Bayesian analysis are greatly simplified if the prior distributions are assumed to be Gaussian distributions. For a finite set of variables $S = (\mathbf{x}_1, \ldots, \mathbf{x}_\ell)$, a Gaussian distribution (with zero mean) is specified by a symmetric positive definite covariance matrix $\Sigma = \Sigma(\mathbf{x}_1, \ldots, \mathbf{x}_\ell)$ with the corresponding distribution given by

$$P_{f\sim\mathscr{D}}[(f(\mathbf{x}_1), \ldots, f(\mathbf{x}_\ell)) = (y_1, \ldots, y_\ell)] \propto \exp\left(-\frac{1}{2}\mathbf{y}'\Sigma^{-1}\mathbf{y}\right).$$

A Gaussian process is a stochastic process for which the marginal distribution for any finite set of variables is zero mean Gaussian. The (i, j) entry of Σ measures the

correlation between $f(\mathbf{x}_i)$ and $f(\mathbf{x}_j)$, that is the expectation $E_{f\sim\mathscr{D}}[f(\mathbf{x}_i)f(\mathbf{x}_j)]$, and hence depends only on $\mathbf{x}_i$ and $\mathbf{x}_j$. There therefore exists a symmetric covariance function $K(\mathbf{x},\mathbf{z})$ such that $\Sigma(\mathbf{x}_1,\ldots,\mathbf{x}_\ell)_{ij} = K(\mathbf{x}_i,\mathbf{x}_j)$. The requirement that the covariance matrix is positive definite for all finite sets of input points is precisely the defining property of a Mercer kernel given in Remark 3.7, and hence we see that defining a Gaussian process over a set of variables indexed by a space X is equivalent to defining a Mercer kernel on $X \times X$. The definition of a Gaussian process by specifying the covariance function avoids explicit definition of the function class $\mathscr{F}$, and the prior over the functions in $\mathscr{F}$. Hence, in much the same way that the feature space is defined implicitly by the kernel in Support Vector Machines, the function class and prior are defined implicitly by the Gaussian process covariance function. It is possible to define a function class and prior for which the kernel is the corresponding covariance function in much the same way that the feature space can be explicitly computed for a kernel. Indeed one choice of function space is the class of linear functions in the space F of Mercer features

$$\mathbf{x} = (x_1,\ldots,x_n) \longmapsto \boldsymbol{\phi}(\mathbf{x}) = (\phi_1(\mathbf{x}),\ldots,\phi_j(\mathbf{x}),\ldots),$$

in the l_2 space defined by the weighted inner product given by

$$\left\langle \boldsymbol{\psi}\cdot\tilde{\boldsymbol{\psi}}\right\rangle = \sum_{j=0}^{\infty}\lambda_j\psi_j\tilde{\psi}_j.$$

The prior distribution $\mathscr{D}$ over the weight vector $\boldsymbol{\psi}$ is chosen to be an independent zero mean Gaussian in each coordinate with variance in coordinate i equal to $\sqrt{\lambda_i}$. With this prior we can compute the correlation $C(\mathbf{x},\mathbf{z})$ between the outputs of two inputs $\mathbf{x}$ and $\mathbf{z}$, as follows:

$$\begin{aligned}
C(\mathbf{x},\mathbf{z}) &= E_{\boldsymbol{\psi}\sim\mathscr{D}}\left[\langle\boldsymbol{\psi}\cdot\boldsymbol{\phi}(\mathbf{x})\rangle\langle\boldsymbol{\psi}\cdot\boldsymbol{\phi}(\mathbf{z})\rangle\right]\\
&= \int_{\mathscr{F}}\langle\boldsymbol{\psi}\cdot\boldsymbol{\phi}(\mathbf{x})\rangle\langle\boldsymbol{\psi}\cdot\boldsymbol{\phi}(\mathbf{z})\rangle\, d\mathscr{D}(\boldsymbol{\psi})\\
&= \sum_{i=0}^{\infty}\sum_{j=0}^{\infty}\phi_i(\mathbf{x})\phi_j(\mathbf{z})\int_{\mathscr{F}}\psi_i\psi_j d\mathscr{D}(\boldsymbol{\psi})\\
&= \sum_{i=0}^{\infty}\sum_{j=0}^{\infty}\phi_i(\mathbf{x})\phi_j(\mathbf{z})\delta_{ij}\lambda_i\\
&= \sum_{i=0}^{\infty}\phi_i(\mathbf{x})\phi_i(\mathbf{z})\lambda_i = K(\mathbf{x},\mathbf{z}),
\end{aligned}$$

as promised.

3.6 Exercises

1. Working in feature space, find the centre of mass and the mean of the squared distances of the points from it.

https://doi.org/10.1017/CBO9780511801389.005 Published online by Cambridge University Press

2. Let $K(\mathbf{x},\mathbf{z}) = \exp(-\|\mathbf{x}-\mathbf{z}\|^2/\sigma^2)$ be the Gaussian kernel of Remark 3.14, which can be applied in any Euclidean or l_2 space. Now consider any kernel $K_1(\mathbf{x},\mathbf{z})$ over $X \times X$ for an input space X. Show how you can compute a Gaussian kernel of the features defined implicitly by K_1 and hence use it as a kernel over $X \times X$.

3. Consider Example 3.15. Using the same feature space construct a kernel for the feature map for which $\phi_u(s)$ counts the number of occurrences of u as a substring of s.

3.7 Further Reading and Advanced Topics

The use of Mercer's theorem for interpreting kernels as inner products in a feature space was introduced into machine learning in 1964 by the work of Aizermann, Bravermann and Rozoener on the method of potential functions [1], but its possibilities were not fully understood until it was first used in the article by Boser, Guyon and Vapnik that introduced the Support Vector method [19].

The theory of kernels is, however, older: Mercer's theorem dates back to 1909 [95], and the study of reproducing kernel Hilbert spaces was developed by Aronszajn in the 1940s [7]. This theory was used in approximation and regularisation theory, see for example the book of Wahba [171] and her 1999 survey [172]. The first use of polynomial kernels was by Poggio in 1975 [115]. Reproducing kernels were extensively used in machine learning and neural networks by Poggio and Girosi, see for example their 1990 paper on radial basis function networks [116].

The theory of positive definite functions was also developed in the context of covariance and correlation functions, so that work in Gaussian processes is closely related [180]. In fact that literature builds on the older results in [172]. Saitoh [123] shows the connection between positivity and the positive semi-definiteness of all finite set kernel matrices mentioned in Remark 3.7.

Techniques for 'making kernels' can be found in many papers, for example by Micchelli [97], MacKay [81], Evgeniou et al. [39], Schölkopf et al. [136], Haussler [58], and Watkins [174]. The discussion about RKHSs draws on the paper of Haussler [58], while Example 3.15 is based on Watkins's paper [176]. The one dimensional shift invariant kernels of Example 3.9 is taken from Girosi [51]. The differential geometric description of the feature space has been provided by Burges [132], along with some necessary conditions for a kernel to satisfy Mercer's theorem.

Building on an observation of Schölkopf [129], Watkins [175] and Haussler [58] have greatly extended the use of kernels, showing that they can in fact be defined on general sets, which do not need to be Euclidean spaces, paving the way for their use in a swathe of new real-world applications, on input spaces as diverse as biological sequences, text, and images. These kernels generalise the idea of recursive ANOVA kernels described in Vapnik [159].

Joachims [67] and Dumais et al. [36] used sparse vectors to encode text features. Jaakkola and Haussler proposed to use a hidden Markov model in

order to evaluate a kernel between biosequences [65], where the feature vector is the Fisher score of the distribution. This is described in more detail in Subsection 8.4.1. Watkins proposed to use probabilistic context free grammars to build kernels between sequences [174]. Also Haussler proposed the use of special kernels for biosequences [58].

An interesting direction of research is to learn the kernel directly from the data. The papers of Jaakkola [65] and Cristianini et al. [31] deal with this issue. An interesting paper by Amari and Wu [3] describes a method for directly acting on the kernel in order to affect the geometry in the input space, so that the separability of the data is improved.

The use of kernels as a general technique for using linear machines in a non-linear fashion can be *exported* to other learning systems, such as nearest neighbour (using the techniques of Section 3.4) or less trivially to PCA as demonstrated by Schölkopf, Smola and Mueller [134]. A technique that uses a specific kernel in order to deal with noisy and non-separable data was introduced by Shawe-Taylor and Cristianini, and will be described in Chapter 6 (see also [140]).

These references are also given on the website **www.support-vector.net**, which will be kept up to date with new work, pointers to software and papers that are available on-line.