CHAPTER

Kernel Methods and Support Vector Machines

16

John Shawe-Taylor* **and Shiliang Sun†**

*Department of Computer Science, University College London, Gower Street, London, United Kingdom
†Department of Computer Science and Technology, East China Normal University, 500 Dongchuan Road, Shanghai, China

Nomenclature

$\mathbf{X}$	The data matrix with each row as an observation
$\kappa(\mathbf{x}, \mathbf{z})$	The kernel function with input vectors $\mathbf{x}$ and $\mathbf{z}$
$\langle \mathbf{x}, \mathbf{z} \rangle$	The inner product between two vectors $\mathbf{x}$ and $\mathbf{z}$
$\phi(\mathbf{x})$	The mapping of $\mathbf{x}$ to the feature space F
$\mathbf{I}_n$	The $n \times n$ identity matrix
K	The kernel matrix with entry K_{ij} being the kernel function value for the ith and jth inputs
$\|\mathbf{w}\|$	The Euclidean norm of the vector $\mathbf{w}$
$\text{trace}(K)$	The trace of the matrix K
$\text{cov}(x, u)$	The covariance between two random scalar variable x and u
$\text{var}(x)$	The variance of the random scalar variable x
$\mathbf{x} \succeq (\preceq) \mathbf{z}$	The vector $\mathbf{x}$ is larger (less) than the vector $\mathbf{z}$ elementwise
$K \succeq 0$	The matrix K is positive semidefinite

1.16.1 Introduction

Data often possess some intrinsic regularities which, if revealed, can facilitate people to understand data themselves or make predictions about new data from the same source. These regularities are called patterns, and pattern analysis, which has been studied broadly such as in statistics, artificial intelligence and signal processing, deals with the automatic detection of patterns in data.

The development of pattern analysis algorithms can be summarized with three important stages [1]. In the 1950s and 1960s, efficient algorithms such as the perceptron [2] were used. They are well understood and effective for detecting linear patterns, though were shown to be limited in complexity. In the 1980s, with the introduction of both the backpropagation algorithm for multi-layer networks

Academic Press Library in Signal Processing. http://dx.doi.org/10.1016/B978-0-12-396502-8.00016-4

[3] and decision trees [4,5], pattern analysis underwent a nonlinear revolution. These methods made a high impact to efficiently and reliably detect nonlinear patterns, though they are largely heuristical with limited statistical analysis and often get trapped with local minima. In the 1990s, the emerging of kernel methods [1,6] for which support vector machines (SVMs) [7,8] are the earliest and foremost influential finally enabled people to deal with nonlinear patterns in the input space via linear patterns in high dimensional spaces. This third generation of pattern analysis algorithms are well-founded just like their linear counterparts, but wipe off the drawbacks of local minima and limited statistical analysis which are typical for multi-layer neural networks and decision trees. Since the 1990s, the algorithms and application scopes of kernel methods have been extended rapidly, from classification to regression, to clustering and many other machine learning tasks.

The approach of kernel methods has four key aspects: (i) Data are embedded into a Euclidean feature space; (ii) Linear relations are sought in the feature space; (iii) Algorithms are implemented so that only inner products between vectors in the feature space are required; (iv) The products can be directly computed from the original data by an efficient “short-cut" known as a kernel function (or kernel for short). This is also known as the kernel trick. The idea of using kernel functions as inner products in a feature space is not new. It was introduced into machine learning in 1964 with the method of potential functions [9] and this work is mentioned in a footnote of Duda and Hart’s pattern classification book [10]. Through this route, the authors of [7] noticed this idea, combined it with large margin hyperplanes in the later SVMs and thus introduced the notion of kernels into the mainstream of the machine learning literature.

Although basic kernel methods are rather mature techniques, research combining them with other techniques is still going on, e.g., kernels have been successfully applied to multi-view learning, semi-supervised learning and multitask learning problems [11–16]. This forms a continual impetus along the line of research on kernel-based learning methods. More importantly, recent work on multiple kernel learning [17] has promoted the study of kernel methods to a new level. This article reviews both classical and some recent research developments on kernel methods, with emphases on the “plug-and-play” flavor of kernel methods.

The rest of this article is organized as follows. Section 1.16.2 introduces the kernel trick and properties and types of kernels, which constitute the foundations of kernel methods. In addition to the kernel ridge regression method presented in Section 1.16.2, Section 1.16.3 reviews some fundamental kernel methods including kernel principal component analysis, kernel canonical correlation analysis, kernel Fisher discriminant analysis, support vector machines, and Gaussian processes. Section 1.16.4 discusses the computational issues of kernel methods and algorithms towards their efficient implementations. Section 1.16.5 briefly surveys the recent developments on multiple kernel learning. Section 1.16.6 presents some practical applications of kernel methods and SVMs. Finally, open issues and problems are discussed in Section 1.16.7 after a brief concluding summary of the article.

1.16.2 Foundations of kernel methods

In this section, we first illustrate key concepts for kernel methods from kernel ridge regression, and then discuss properties of valid kernels. Finally, kernel design strategies are introduced.

1.16.2.1 The kernel trick: ridge regression as an example

Consider the problem of finding a homogeneous real-valued linear function

$$g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle = \mathbf{x}^\top \mathbf{w} = \sum_{i=1}^{n} w_i x_i, \tag{16.1}$$

that best interpolates a given training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ of points $\mathbf{x}_i \in \mathbb{R}^n$ with corresponding labels $y_i \in \mathbb{R}$. A commonly chosen measure of the discrepancy between a function output and the real observation is

$$f_g((\mathbf{x}, y)) = (g(\mathbf{x}) - y)^2. \tag{16.2}$$

Suppose the m inputs of S are stored in the matrix $\mathbf{X}$ as row vectors, and the corresponding outputs constitute vector $\mathbf{y}$ with $\mathbf{y} = [y_1, \ldots, y_m]^\top$. Hence we can write $\boldsymbol{\xi} = \mathbf{y} - \mathbf{X}\mathbf{w}$ for the vector of differences between $g(\mathbf{x}_i)$ and y_i. Ridge regression corresponds to solving the following optimization with a simple norm regularizer

$$\min_{\mathbf{w}} \mathcal{L}_\lambda(\mathbf{w}, S) = \min_{\mathbf{w}} \lambda \|\mathbf{w}\|^2 + \|\boldsymbol{\xi}\|^2, \tag{16.3}$$

where $\lambda > 0$ defines the relative tradeoff between the norm and loss. Setting the derivative of $\mathcal{L}_\lambda(\mathbf{w}, S)$ with respect to the parameter vector $\mathbf{w}$ equal to 0 gives

$$\mathbf{X}^\top \mathbf{X}\mathbf{w} + \lambda \mathbf{w} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_n)\mathbf{w} = \mathbf{X}^\top \mathbf{y}, \tag{16.4}$$

where $\mathbf{I}_n$ is the $n \times n$ identity matrix. Thus we get the primal solution (referring to the explicit representation) for the weight vector

$$\mathbf{w} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_n)^{-1} \mathbf{X}^\top \mathbf{y}, \tag{16.5}$$

from which the resulting prediction function $g(\mathbf{x})$ can be readily given.

Alternatively, from (16.4) we get

$$\mathbf{w} = \mathbf{X}^\top \frac{1}{\lambda}(\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{X}^\top \boldsymbol{\alpha} = \sum_{i=1}^{m} \alpha_i \mathbf{x}_i, \tag{16.6}$$

where parameters $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_m]^\top \triangleq \lambda^{-1}(\mathbf{y} - \mathbf{X}\mathbf{w})$ are known as the dual variables. Substituting $\mathbf{w} = \mathbf{X}^\top \boldsymbol{\alpha}$ into $\boldsymbol{\alpha} = \lambda^{-1}(\mathbf{y} - \mathbf{X}\mathbf{w})$, we obtain

$$\boldsymbol{\alpha} = (\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}_m)^{-1} \mathbf{y}, \tag{16.7}$$

which is called the dual solution. The dual solution expresses the weight vector $\mathbf{w}$ as a linear combination of the training examples. Denote the term $\mathbf{X}\mathbf{X}^\top$ by $\mathbf{K}$. It follows that $\mathbf{K}_{i,j} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$. Now the resulting prediction function is formulated as

$$g(\mathbf{x}) = \mathbf{x}^\top \mathbf{w} = \mathbf{x}^\top \mathbf{X}^\top \boldsymbol{\alpha} = \left\langle \mathbf{x}, \sum_{i=1}^{m} \alpha_i \mathbf{x}_i \right\rangle = \sum_{i=1}^{m} \alpha_i \langle \mathbf{x}, \mathbf{x}_i \rangle. \tag{16.8}$$

There are two ingredients embedded in the dual form of ridge regression: computing vector $\boldsymbol{\alpha}$ and evaluation of the prediction function. Both operations only involve inner products between data inputs.

Since the computation only involves inner products, we can substitute for all occurrences of $\langle \cdot, \cdot \rangle$ a kernel function κ that computes $\kappa(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$ and we obtain an algorithm for ridge regression in the feature space F defined by the mapping $\phi : \mathbf{x} \mapsto \phi(\mathbf{x}) \in F$. This is an instantiation of the kernel trick for ridge regression and results in the kernel ridge regression algorithm. Through kernel ridge regression we can perform linear regression in very high-dimensional spaces efficiently, which is equivalent to performing non-linear regression in the original input space.

1.16.2.2 Properties of kernels

Definition 1 (Kernel function). A kernel is a function κ that for all $\mathbf{x}$, $\mathbf{z}$ from a nonempty set $\mathcal{X}$ (which need not be a vector space) satisfies

$$\kappa(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle, \tag{16.9}$$

where ϕ is a mapping from the set $\mathcal{X}$ to a Hilbert space F that is usually called the feature space

$$\phi : \mathbf{x} \in \mathcal{X} \mapsto \phi(\mathbf{x}) \in F. \tag{16.10}$$

To verify whether a function is a valid kernel, one approach is to construct a feature space for which the function value for two inputs corresponds to first performing an explicit feature mapping and then computing the inner product between their images. An alternative approach, which is more widely used, is to investigate the finitely positive semidefinite property [1,6,18–20].

Definition 2 (Finitely positive semidefinite function). A function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ satisfies the finitely positive semidefinite property if it is a *symmetric* function for which the kernel matrices K with $K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ formed by restriction to any finite subset of $\mathcal{X}$ are positive semidefinite.

The feasibility of the above property for characterizing kernels is justified by the following theorem [1,6,21].

Theorem 3 (Characterization of kernels). *A function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which is either continuous or has a countable domain, can be decomposed as an inner product in a Hilbert space F by a feature map ϕ applied to both its arguments*

$$\kappa(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle \tag{16.11}$$

if and only if it satisfies the finitely positive semidefinite property.

A Hilbert space F is defined as an inner product space that is complete where completeness means that every Cauchy sequence of elements of F converges to an element in this space. If the separability property is further added to the definition of a Hilbert space, where a space is separable if there is a countable set of elements from this space such that the distance between each element of this space and some element of this countable set is less than any predefined threshold, the existence of "kernels are continuous or the domain is countable" in Theorem 3 is then necessary.

The Hilbert space constructed in proving Theorem 3 is called the reproducing kernel Hilbert space (RKHS) because the following reproducing property of the kernel resulting from the defined inner product holds

$$\langle f_F(\,\cdot\,), \kappa(\mathbf{x}, \cdot) \rangle = f_F(\mathbf{x}), \tag{16.12}$$

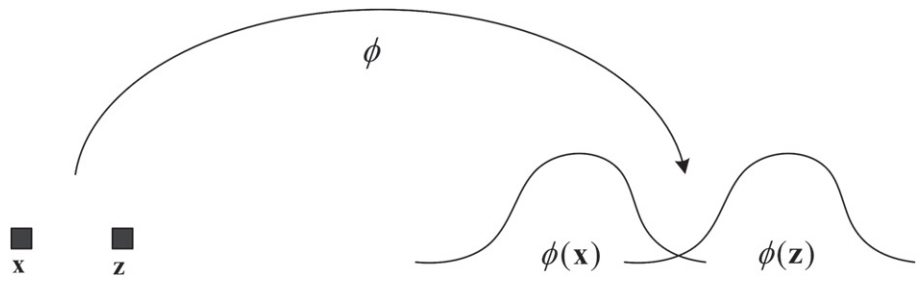

FIGURE 16.1

One instantiation of the feature mapping using a Gaussian kernel.

where f_F is a function of the function space F, and function $\kappa(\mathbf{x}, \cdot)$ is the mapping $\phi(\mathbf{x})$ which actually represents the similarity of $\mathbf{x}$ to all other points in $\mathcal{X}$, as shown in Figure 16.1 [6].

By construction, $f_F(\cdot)$ takes the form of an arbitrarily-weighted linear combination of countable images of the original inputs. For any two such functions

$$f_{F1}(\cdot) = \sum_{i=1}^{\ell_1} \alpha_i \kappa(\mathbf{x}_i, \cdot), \quad f_{F2}(\cdot) = \sum_{j=1}^{\ell_2} \beta_j \kappa(\mathbf{x}'_j, \cdot) \tag{16.13}$$

where $\ell_1, \ell_2 \in \mathbb{N}$, $\alpha_i, \beta_j \in \mathbb{R}$ and $\mathbf{x}_i, \mathbf{x}'_j \in \mathcal{X}$, the dot product is defined as

$$\langle f_{F1}(\cdot), f_{F2}(\cdot) \rangle \triangleq \sum_{i=1}^{\ell_1} \sum_{j=1}^{\ell_2} \alpha_i \beta_j \kappa(\mathbf{x}_i, \mathbf{x}'_j). \tag{16.14}$$

The inner product space or pre-Hilbert space formed by $f_F(\cdot)$ is then completed to form the Hilbert space F where the mathematical trick "completion" refers to adding all limit points of Cauchy sequences to the space [6].

It should be noted that there are different approaches to constructing feature spaces for any given kernel. Besides the above construction, the Mercer kernel map [22], though not mentioned much here, is also widely applicable, especially in the SVM literature. The feature spaces constructed in different ways can even have different dimensions. However, since we are only interested in dot products, these spaces can be regarded as identical.

For some kernels, the feature map and feature space can be explicitly built with a simple form. For instance, consider the homogeneous quadratic kernel

$$\kappa(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle^2, \tag{16.15}$$

which can be reformulated as

$$\kappa(\mathbf{x}, \mathbf{z}) = (\mathbf{x}'\mathbf{z})^2 = \mathbf{z}'(\mathbf{x}\mathbf{x}')\mathbf{z} = \langle \mathrm{vec}(\mathbf{z}\mathbf{z}'), \mathrm{vec}(\mathbf{x}\mathbf{x}') \rangle, \tag{16.16}$$

where $\mathrm{vec}(A)$ stacks the column of matrix A on top of each other in the manner that the first column situates at the top. The feature map corresponding to κ would be $\phi(\mathbf{x}) = \mathrm{vec}(\mathbf{x}\mathbf{x}')$. The feature space can be the Euclidean space with dimensionality being the total number of entries of $\mathrm{vec}(\mathbf{x}\mathbf{x}')$.

1.16.2.3 Types of kernels

The use of kernels provides a powerful and principled approach to modeling nonlinear patterns through linear patterns in a feature space. Another benefit is that the design of kernels and linear methods can be decoupled, which greatly facilitates the modularity of machine learning methods.

Representative kernels include the linear kernel $\kappa(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle$, inhomogeneous polynomial kernel

$$\kappa(\mathbf{x}, \mathbf{z}) = (\langle \mathbf{x}, \mathbf{z} \rangle + R)^d \tag{16.17}$$

where d is the degree of the polynomial and parameter $R \in \mathbb{R}$, and the Gaussian radial basis function (RBF) kernel (Gaussian kernel for short) with parameter $\sigma > 0$

$$\kappa(\mathbf{x}, \mathbf{z}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{z}\|^2}{2\sigma^2}\right). \tag{16.18}$$

The polynomial kernel (16.17) can be expanded by the binomial theorem as

$$(\langle \mathbf{x}, \mathbf{z} \rangle + R)^d = \sum_{s=0}^{d} \binom{d}{s} R^{d-s} \langle \mathbf{x}, \mathbf{z} \rangle^s. \tag{16.19}$$

Hence, the features for each component in the sum together form the features of the kernel. In other words, we have a reweighting of the features of the homogeneous polynomial kernel

$$\kappa(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle^s, \quad s = 0, \ldots, d, \tag{16.20}$$

where one construction of the feature map corresponding to kernel (16.20) is using a vector with entries being all ordered monomials (e.g., $x_1 x_2$ and $x_2 x_1$ are treated as separate features) of degree s, that is, each entry is an instantiation of product $x_{j_1} \ldots x_{j_s}$ with $j_1, \ldots j_s \in \{1, \ldots, n\}$ [6]. The parameter R allows the control of the relative weightings of the monomials with different degrees. The weight formulation $\binom{d}{s} R^{d-s}$ indicates that increasing R decreases the relative weighting of higher order monomials [1].

For the Gaussian kernel (16.18) the images of all points have norm 1 in the feature space as a result of $\kappa(\mathbf{x}, \mathbf{x}) = 1$. It can be obtained by normalizing $\exp(\langle \mathbf{x}, \mathbf{z} \rangle / \sigma^2)$

$$\begin{aligned} \exp\left(-\frac{\|\mathbf{x} - \mathbf{z}\|^2}{2\sigma^2}\right) &= \exp\left(\frac{\langle \mathbf{x}, \mathbf{z} \rangle}{\sigma^2} - \frac{\langle \mathbf{x}, \mathbf{x} \rangle}{2\sigma^2} - \frac{\langle \mathbf{z}, \mathbf{z} \rangle}{2\sigma^2}\right) \\ &= \frac{\exp(\langle \mathbf{x}, \mathbf{z} \rangle / \sigma^2)}{\sqrt{\exp(\|\mathbf{x}\|^2/\sigma^2) \exp(\|\mathbf{z}\|^2/\sigma^2)}}. \end{aligned} \tag{16.21}$$

Because an exponential function can be arbitrarily closely approximated by polynomials with positive coefficients

$$\exp(x) = \sum_{i=0}^{\infty} \frac{1}{i!} x^i, \tag{16.22}$$

the function $\exp(\langle \mathbf{x}, \mathbf{z} \rangle / \sigma^2)$ is arguably a kernel. Therefore, the Gaussian kernel (16.18) is a polynomial kernel of infinite degree, and its features can be all ordered monomials of input features with no restriction placed on the degrees. However, with increasing degree the weighting of individual monomials falls off as $i!$ [1].

One appeal of using kernel methods is that kernels are not restricted to vectorial data, making it possible to apply the techniques to diverse types of objects. Not surprisingly, kernels can be designed

for sets, strings, text documents, graphs and graph-nodes [1]. For these kernels, we would not elaborate here. However, an effective design of kernels has to be embedded with some prior knowledge on how to characterize similarity between data.

We now focus on two types of kernels induced by probabilistic models, marginalization kernels and Fisher kernels. These techniques are useful for combining generative and discriminative methods for machine learning. The marginalization kernels are defined as follows.

Definition 4 (Marginalization kernels). Given a set of data models M and a prior distribution P_M on M, the probability that an example pair $\mathbf{x}$ and $\mathbf{z}$ is generated together can be computed as

$$P_M(\mathbf{x}, \mathbf{z}) = \sum_{m \in M} P(\mathbf{x}|m) P(\mathbf{z}|m) P_M(m). \tag{16.23}$$

If we consider the mapping function

$$\phi : \mathbf{x} \mapsto (P(\mathbf{x}|m))_{m \in M} \in F \tag{16.24}$$

in a feature space F indexed by M, $P_M(\mathbf{x}, \mathbf{z})$ corresponds to the inner product

$$\langle f, g \rangle = \sum_{m \in M} f_m g_m P_M(m) \tag{16.25}$$

between $\phi(\mathbf{x})$ and $\phi(\mathbf{z})$. $P_M(\mathbf{x}, \mathbf{z})$ is referred to as the marginalization kernel for the model class M.

The above computation can be viewed as a marginalization operation for the probability distribution of triples $(\mathbf{x}, \mathbf{z}, m)$ over m (with conditional independence of $\mathbf{x}$ and $\mathbf{z}$ given a specific model m), and therefore comes the name marginalization kernels. The assumption of conditional independence is a sufficient condition for positive semi-definiteness. For an input, marginalization kernels treat the output probability given one model as a feature. Since the information from a single model is quite limited, they usually adopt multiple different models to reach a representation of the input.

Fisher kernels, defined by [23,24], are an alternative way of extracting information, usually from a single generative model, however. The single model is required to be smoothly parameterized so that derivatives of the model with respect to the parameters is computable. An intuitive interpretation of Fisher kernels is that it describes data points by the variation of some quantity (say the log of the likelihood function) caused by slight parameter perturbations.

Definition 5 (Fisher score and Fisher information matrix). For a given setting of the parameters $\boldsymbol{\theta}^0$ (e.g., obtained by the maximum likelihood rule) the log-likelihood of a data point $\mathbf{x}$ with respect to the model $m(\boldsymbol{\theta}^0)$ is defined to be $\log P(x|\boldsymbol{\theta}^0)$. Consider the gradient vector of the log-likelihood

$$\mathbf{g}(\boldsymbol{\theta}, \mathbf{x}) = \left(\frac{\partial \log P(x|\boldsymbol{\theta})}{\partial \theta_i} \right)_{i=1}^{N}, \tag{16.26}$$

where $\boldsymbol{\theta} \in \mathbb{R}^N$. The Fisher score of a data point $\mathbf{x}$ with respect to the model $m(\boldsymbol{\theta}^0)$ is $\mathbf{g}(\boldsymbol{\theta}^0, \mathbf{x})$. The Fisher information matrix with respect to the model $m(\boldsymbol{\theta}^0)$ is given by

$$\mathbf{I}_{\text{Fisher}} = \mathbb{E}\left[\mathbf{g}(\boldsymbol{\theta}^0, \mathbf{x}) \mathbf{g}(\boldsymbol{\theta}^0, \mathbf{x})^\top \right], \tag{16.27}$$

where the expectation is over the distribution of the data point $\mathbf{x}$.

The Fisher score embeds a data point into the feature space $\mathbb{R}^N$, and provides direct constructions of kernels.

Definition 6 (Fisher kernel). The invariant Fisher kernel with respect to the model $m(\boldsymbol{\theta}^0)$ for a given setting of the parameters $\boldsymbol{\theta}^0$ is defined as

$$\kappa(\mathbf{x}, \mathbf{z}) = \mathbf{g}(\boldsymbol{\theta}^0, \mathbf{x})^\top \mathbf{I}_{Fisher}^{-1} \mathbf{g}(\boldsymbol{\theta}^0, \mathbf{z}). \tag{16.28}$$

The practical Fisher kernel is defined as

$$\kappa(\mathbf{x}, \mathbf{z}) = \mathbf{g}(\boldsymbol{\theta}^0, \mathbf{x})^\top \mathbf{g}(\boldsymbol{\theta}^0, \mathbf{z}). \tag{16.29}$$

The invariant Fisher kernel is computationally more demanding as it requires the computation and inversion of the Fisher information matrix. It is named "invariant" because the resulting kernel would not change if we reparameterize the model with an invertible differentiable transformation $\boldsymbol{\psi} = \boldsymbol{\psi}(\boldsymbol{\theta})$. Suppose $\tilde{\kappa}$ is the transformed kernel. It follows that

$$\mathbf{g}(\boldsymbol{\theta}^0, \mathbf{x})^\top = \left(\left(\frac{\partial \log P(x|\boldsymbol{\psi})}{\partial \psi_i} \right)_{i=1}^N \right)^\top \mathbf{J}(\boldsymbol{\psi}) = \mathbf{g}(\boldsymbol{\psi}^0, \mathbf{x})^\top \mathbf{J}(\boldsymbol{\psi}^0), \tag{16.30}$$

where matrix $\mathbf{J}(\boldsymbol{\psi}^0)$ is the Jacobian of the transformation $\boldsymbol{\psi}$ evaluated at $\boldsymbol{\psi}^0$ [1]. Now we have

$$\begin{aligned} \tilde{\kappa}(\mathbf{z}_1, \mathbf{z}_2) &= \mathbf{g}(\boldsymbol{\psi}^0, \mathbf{z}_1)^\top \mathbb{E}\left[(\mathbf{J}(\boldsymbol{\psi}^0)^{-1})^\top \mathbf{g}(\boldsymbol{\theta}^0, \mathbf{x}) \mathbf{g}(\boldsymbol{\theta}^0, \mathbf{x})^\top \mathbf{J}(\boldsymbol{\psi}^0)^{-1} \right]^{-1} \mathbf{g}(\boldsymbol{\psi}^0, \mathbf{z}_2) \\ &= \mathbf{g}(\boldsymbol{\theta}^0, \mathbf{z}_1)^\top \mathbb{E}\left[\mathbf{g}(\boldsymbol{\theta}^0, \mathbf{x}) \mathbf{g}(\boldsymbol{\theta}^0, \mathbf{x})^\top \right]^{-1} \mathbf{g}(\boldsymbol{\theta}^0, \mathbf{z}_2) \\ &= \kappa(\mathbf{z}_1, \mathbf{z}_2). \end{aligned} \tag{16.31}$$

Hence, the invariant Fisher kernel is desirable if the choice of parameterizations is somewhat arbitrary. But for this kernel there is a caveat when the natural approximation of the Fisher information matrix by its empirical estimate is used

$$\hat{\mathbf{I}}_{\text{Fisher}} = \hat{\mathbb{E}}\left[\mathbf{g}(\boldsymbol{\theta}^0, \mathbf{x}) \mathbf{g}(\boldsymbol{\theta}^0, \mathbf{x})^\top \right] = \frac{1}{m} \sum_{i=1}^m \mathbf{g}(\boldsymbol{\theta}^0, \mathbf{x}_i) \mathbf{g}(\boldsymbol{\theta}^0, \mathbf{x}_i)^\top, \tag{16.32}$$

in which case $\hat{\mathbf{I}}_{\text{Fisher}}$ is the empirical covariance matrix of the Fisher scores. The invariant Fisher kernel is thus equivalent to whitening the scores. The negative effect is that we may amplify noise if some parameters are not relevant for the information, and therefore the signal to noise ratio is possibly reduced. This can be regarded as the cost of the invariance.

Apart from the kernels introduced so far, more complicated kernels can be constructed with them as building blocks. The following theorem [1] lists some strategies for kernel constructions.

Theorem 7 (Kernel constructions). *Let κ_1, κ_2, and κ_3 be valid kernels, ϕ any feature map to the domain of κ_3, $a \geq 0$, $f(\cdot)$ any real-valued function, and* $\mathbf{B}$ *a positive semi-definite matrix. Then the following functions are valid kernels:*

- $\kappa(\mathbf{x}, \mathbf{z}) = \kappa_1(\mathbf{x}, \mathbf{z}) + \kappa_2(\mathbf{x}, \mathbf{z})$,
- $\kappa(\mathbf{x}, \mathbf{z}) = a\kappa_1(\mathbf{x}, \mathbf{z})$,

- $\kappa(\mathbf{x}, \mathbf{z}) = \kappa_1(\mathbf{x}, \mathbf{z})\kappa_2(\mathbf{x}, \mathbf{z})$,
- $\kappa(\mathbf{x}, \mathbf{z}) = f(\mathbf{x})f(\mathbf{z})$,
- $\kappa(\mathbf{x}, \mathbf{z}) = \kappa_3(\phi(\mathbf{x}), \phi(\mathbf{z}))$,
- $\kappa(\mathbf{x}, \mathbf{z}) = \mathbf{x}^\top \mathbf{B}\mathbf{z}$ *(for now* $\mathbf{x}$ *and* $\mathbf{z}$ *are vectorial data).*

1.16.3 Fundamental kernel methods

In this section, we introduce some fundamental kernel methods ranging from unsupervised learning to supervised learning. These methods have a large popularity either because they are among the first uses of kernels or because they address very fundamental learning problems.

1.16.3.1 Kernel principal component analysis

Principal component analysis (PCA) finds a set of orthogonal directions which forms a subspace to maximize variances. In this way, data can be reconstructed with minimal quadratic error. Suppose the inputs of the data set S given in Section 1.16.2.1 is centered with mean $\mathbf{0}$. The direction that maximizes the variance can be found by solving the following problem

$$\begin{aligned} \max_{\mathbf{w}} \quad & \mathbf{w}^\top \mathbf{C}\mathbf{w} \\ \text{s.t.} \quad & \|\mathbf{w}\| = 1, \end{aligned} \tag{16.33}$$

where $\mathbf{C} = \frac{1}{m}\mathbf{X}^\top\mathbf{X}$ is the covariance matrix (strictly speaking, an empirical estimate of the covariance) of the input data. The solution is given by the eigenvector of $\mathbf{C}$ corresponding to the largest eigenvalue with the objective value being the eigenvalue. The direction of the second largest variance can be searched for in the subspace orthogonal to the direction already found. This results in the eigenvector corresponding to the second largest eigenvalue. It is readily provable that PCA projects data into the space spanned by the k largest eigenvectors of $\mathbf{C}$ if we would like to find a k-dimensional subspace. The new coordinates by which we represent the data are known as principal components. Although centering data before performing PCA is not a must, it has the advantage of reducing the overall sum of the eigenvalues and thus removing irrelevant variance arising from data shift [1].

The kernel PCA [25] extends the linear PCA algorithm to extracting nonlinear structures in terms of kernels. Now we provide a simple derivation of the kernel PCA by exploiting the relationship between $\mathbf{X}^\top\mathbf{X}$ and $\mathbf{X}\mathbf{X}^\top$ [1]. It is easy to show that these two matrices have the same rank. More interestingly, their eigen-decompositions correspond to each other. Suppose that $\mathbf{w}, \lambda$ is an eigenvector-eigenvalue pair for $\mathbf{X}^\top\mathbf{X}$, then $\mathbf{X}\mathbf{w}, \lambda$ is for $\mathbf{X}\mathbf{X}^\top$

$$(\mathbf{X}\mathbf{X}^\top)\mathbf{X}\mathbf{w} = \mathbf{X}(\mathbf{X}^\top\mathbf{X})\mathbf{w} = \lambda\mathbf{X}\mathbf{w}, \tag{16.34}$$

and conversely, if $\boldsymbol{\alpha}, \lambda$ is an eigenvector-eigenvalue pair for the matrix $\mathbf{X}\mathbf{X}^\top$, then $\mathbf{X}^\top\boldsymbol{\alpha}, \lambda$ is for $\mathbf{X}^\top\mathbf{X}$

$$(\mathbf{X}^\top\mathbf{X})\mathbf{X}^\top\boldsymbol{\alpha} = \mathbf{X}^\top(\mathbf{X}\mathbf{X}^\top)\boldsymbol{\alpha} = \lambda\mathbf{X}^\top\boldsymbol{\alpha}. \tag{16.35}$$

This gives a dual representation for the eigenvector of $\mathbf{X}^\top\mathbf{X}$ from the eigen-decomposition of $\mathbf{X}\mathbf{X}^\top$. $\mathbf{X}\mathbf{X}^\top$ is actually a kernel matrix if we replace each row $\mathbf{x}_i^\top$ of $\mathbf{X}$ by its image $\phi(\mathbf{x}_i)^\top$ in a feature space, and $\mathbf{X}^\top\mathbf{X}$ would be the scaled covariance matrix without centering.

Centering data in a feature space is not so simple as in the original space. Suppose that a kernel κ is adopted with the kernel matrix $\mathbf{K}$ computed from the original data. Centering data in the feature space corresponds to defining a new feature map $\hat{\phi}(\mathbf{x}) = \phi(\mathbf{x}) - \frac{1}{m}\sum_{i=1}^{m}\phi(\mathbf{x}_i)$. The new kernel matrix for the centered data would be

$$\hat{\mathbf{K}} = \mathbf{K} - \frac{1}{m}\mathbf{j}\mathbf{j}^\top\mathbf{K} - \frac{1}{m}\mathbf{K}\mathbf{j}\mathbf{j}^\top + \frac{1}{m^2}(\mathbf{j}^\top\mathbf{K}\mathbf{j})\mathbf{j}\mathbf{j}^\top, \tag{16.36}$$

where $\mathbf{j}$ is the all 1s vector [1]. Suppose that $\hat{\boldsymbol{\alpha}}$, $\hat{\lambda}$ is an eigenvector-eigenvalue pair for the kernel matrix $\hat{\mathbf{K}} = \hat{\mathbf{X}}\hat{\mathbf{X}}^\top$ where $\|\hat{\boldsymbol{\alpha}}\| = 1$ and the ith row of $\hat{\mathbf{X}}$ is $\hat{\phi}(\mathbf{x}_i)^\top$. Then $\hat{\mathbf{X}}^\top\hat{\boldsymbol{\alpha}}$ is the eigenvector of the covariance matrix $\frac{1}{m}\hat{\mathbf{X}}^\top\hat{\mathbf{X}}$ which has the same eigenvectors with $\hat{\mathbf{X}}^\top\hat{\mathbf{X}}$. Usually we require that the final projection vector is normalized, that is, $\|\hat{\mathbf{X}}^\top\hat{\boldsymbol{\alpha}}\| = 1$. Because for $\|\hat{\boldsymbol{\alpha}}\| = 1$ we have

$$\|\hat{\mathbf{X}}^\top\hat{\boldsymbol{\alpha}}\|^2 = \hat{\boldsymbol{\alpha}}^\top\hat{\mathbf{X}}\hat{\mathbf{X}}^\top\hat{\boldsymbol{\alpha}} = \hat{\boldsymbol{\alpha}}^\top\hat{\mathbf{K}}\hat{\boldsymbol{\alpha}} = \hat{\lambda}, \tag{16.37}$$

to meet $\|\hat{\mathbf{X}}^\top\hat{\boldsymbol{\alpha}}\| = 1$, $\hat{\boldsymbol{\alpha}}$ should be further divided by $\sqrt{\hat{\lambda}}$. Hence, the k projection directions derived from kernel PCA should be

$$\left\{\frac{1}{\sqrt{\hat{\lambda}_i}}\hat{\mathbf{X}}^\top\hat{\boldsymbol{\alpha}}_i\right\}_{i=1}^{k}, \tag{16.38}$$

where $\{\hat{\boldsymbol{\alpha}}_i, \hat{\lambda}_i\}_{i=1}^{k}$ are the k leading eigenvector-eigenvalue pairs for the kernel matrix $\hat{\mathbf{K}}$ and the norms of $\{\hat{\boldsymbol{\alpha}}_i\}_{i=1}^{k}$ are all 1. The projections of a new input $\mathbf{x}$ would be the inner products between the above directions and $\phi(\mathbf{x}) - \frac{1}{m}\sum_{i=1}^{m}\phi(\mathbf{x}_i)$.

1.16.3.2 Kernel canonical correlation analysis

Canonical correlation analysis (CCA), proposed by [26], works on a paired dataset (i.e., data with two representations) to find two linear transformations each for one of the two representations such that the correlations between the transformed variables are maximized. It was later generalized to more than two sets of variables in several ways [27,28]. Here we only focus on the situation of two sets of variables.

Suppose we have a paired dataset $S_{\mathbf{x},\mathbf{u}} = \{(\mathbf{x}_1, \mathbf{u}_1), \ldots, (\mathbf{x}_m, \mathbf{u}_m)\}$. For example, $\mathbf{x}_i$ and the corresponding $\mathbf{u}_i$ can be the representations of a same sematic content in two different languages. CCA attempts to seek the projection directions $\mathbf{w}_x$ and $\mathbf{w}_u$ to maximize the following empirical correlation

$$\frac{\mathrm{cov}(\mathbf{w}_x^\top\mathbf{x}, \mathbf{w}_u^\top\mathbf{u})}{\sqrt{\mathrm{var}(\mathbf{w}_x^\top\mathbf{x})\mathrm{var}(\mathbf{w}_u^\top\mathbf{u})}} = \frac{\mathbf{w}_x^\top\mathbf{C}_{xu}\mathbf{w}_u}{\sqrt{(\mathbf{w}_x^\top\mathbf{C}_{xx}\mathbf{w}_x)(\mathbf{w}_u^\top\mathbf{C}_{uu}\mathbf{w}_u)}}, \tag{16.39}$$

where covariance matrix $\mathbf{C}_{xu}$ is defined as (definitions for $\mathbf{C}_{xx}$ and $\mathbf{C}_{uu}$ can be obtained analogously)

$$\mathbf{C}_{xu} = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{x}_i - \mathbf{m}_x)(\mathbf{u}_i - \mathbf{m}_u)^\top \tag{16.40}$$

with $\mathbf{m}_x$ and $\mathbf{m}_u$ being the means of the two representations, respectively

$$\mathbf{m}_x = \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i, \quad \mathbf{m}_u = \frac{1}{m}\sum_{i=1}^{m}\mathbf{u}_i. \tag{16.41}$$

Because the scales of $\mathbf{w}_x$ and $\mathbf{w}_u$ have no effects on the value of (16.39), we can constrain each of the two terms in the denominator to take value 1. Thus we reach another widely used objective for CCA

$$\begin{aligned} \max_{\mathbf{w}_x, \mathbf{w}_u} \quad & \rho = \mathbf{w}_x^\top \mathbf{C}_{xu} \mathbf{w}_u \\ \text{s.t.} \quad & \mathbf{w}_x^\top \mathbf{C}_{xx} \mathbf{w}_x = 1, \ \mathbf{w}_u^\top \mathbf{C}_{uu} \mathbf{w}_u = 1. \end{aligned} \tag{16.42}$$

The solution is given by first solving the generalized eigenvalue problem [1]

$$\begin{pmatrix} \mathbf{0} & \mathbf{C}_{xu} \\ \mathbf{C}_{ux} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{w}_x \\ \mathbf{w}_u \end{pmatrix} = \lambda \begin{pmatrix} \mathbf{C}_{xx} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{uu} \end{pmatrix} \begin{pmatrix} \mathbf{w}_x \\ \mathbf{w}_u \end{pmatrix}, \tag{16.43}$$

and then normalizing the resulting directions to comply with the constraints of (16.42). Note that the eigenvalue λ for a particular eigenvector $\binom{\mathbf{w}_x}{\mathbf{w}_u}$ gives the corresponding correlation value

$$\rho = \mathbf{w}_x^\top \mathbf{C}_{xu} \mathbf{w}_u = \mathbf{w}_x^\top (\lambda \mathbf{C}_{xx} \mathbf{w}_x) = \lambda. \tag{16.44}$$

Consequently, all eigenvalues lie in the interval $[-1, +1]$. Interestingly, if $\binom{\mathbf{w}_x}{\mathbf{w}_u}$, λ is an eigenvector-eigenvalue pair, so is $\binom{\mathbf{w}_x}{-\mathbf{w}_u}$, $-\lambda$. Therefore, only half the spectrum, e.g., the positive eigenvalues, are necessary to be considered, and the corresponding eigenvectors constitute desirable projection directions (as with PCA, we often need more than one projection directions). The eigenvectors with largest eigenvalues identify the strongest correlations.

Now we give the dual form of CCA to facilitate the derivation of kernel CCA [29–31]. Assume that the dataset $S_{\mathbf{x},\mathbf{u}}$ is centered, that is, the mean value of each of the two representations is zero. We consider expressing $\mathbf{w}_x$ and $\mathbf{w}_u$ as linear combinations of training examples

$$\mathbf{w}_x = \mathbf{X}^\top \boldsymbol{\alpha}_x, \quad \mathbf{w}_u = \mathbf{U}^\top \boldsymbol{\alpha}_u, \tag{16.45}$$

where the rows of $\mathbf{X}$ and $\mathbf{U}$ are vectors $\mathbf{x}_i^\top$ and $\mathbf{u}_i^\top$ $(i = 1, \ldots, m)$, respectively. Substituting (16.45) into (16.42) results in

$$\begin{aligned} \max_{\boldsymbol{\alpha}_x, \boldsymbol{\alpha}_u} \quad & \boldsymbol{\alpha}_x^\top \mathbf{X}\mathbf{X}^\top \mathbf{U}\mathbf{U}^\top \boldsymbol{\alpha}_u \\ \text{s.t.} \quad & \boldsymbol{\alpha}_x^\top \mathbf{X}\mathbf{X}^\top \mathbf{X}\mathbf{X}^\top \boldsymbol{\alpha}_x = 1, \ \boldsymbol{\alpha}_u^\top \mathbf{U}\mathbf{U}^\top \mathbf{U}\mathbf{U}^\top \boldsymbol{\alpha}_u = 1. \end{aligned} \tag{16.46}$$

Since the above formulation only involves inner products among training examples, we can write down the objective for kernel CCA simply as

$$\begin{aligned} \max_{\boldsymbol{\alpha}_x, \boldsymbol{\alpha}_u} \quad & \boldsymbol{\alpha}_x^\top \mathbf{K}_x \mathbf{K}_u \boldsymbol{\alpha}_u \\ \text{s.t.} \quad & \boldsymbol{\alpha}_x^\top \mathbf{K}_x^2 \boldsymbol{\alpha}_x = 1, \ \boldsymbol{\alpha}_u{}^\top \mathbf{K}_u^2 \boldsymbol{\alpha}_u = 1, \end{aligned} \tag{16.47}$$

where $\mathbf{K}_x$ and $\mathbf{K}_u$ are the kernel matrices for the two representations, respectively (if data are not centered in feature spaces, techniques similar to centering for kernel PCA can be adopted).

It was shown that overfitting with perfect correlations which fail to distinguish spurious features from those revealing the underlying semantics can appear using the above versions of CCA and kernel CCA

[1,27]. In other words, some kind of regularization is needed to detect meaningful patterns. Statistical stability analysis shows that controlling the norms of the two projection directions is a good way for regularization [1]. Hence, we have the regularized CCA whose objective is to maximize

$$\frac{\mathbf{w}_x^\top \mathbf{C}_{xu}\mathbf{w}_u}{\sqrt{\left((1-\tau_x)\mathbf{w}_x^\top \mathbf{C}_{xx}\mathbf{w}_x + \tau_x\|\mathbf{w}_x\|^2\right)\left((1-\tau_u)\mathbf{w}_u^\top \mathbf{C}_{uu}\mathbf{w}_u + \tau_u\|\mathbf{w}_u\|^2\right)}}, \tag{16.48}$$

where regularization parameters τ_x and τ_u vary in the interval $[0, 1]$. The kernel regularized CCA corresponding to (16.47) is given by optimizing

$$\begin{aligned} &\max_{\boldsymbol{\alpha}_x, \boldsymbol{\alpha}_u} \quad \boldsymbol{\alpha}_x^\top \mathbf{K}_x \mathbf{K}_u \boldsymbol{\alpha}_u \\ &\text{s.t.} \quad (1-\tau_x)\boldsymbol{\alpha}_x^\top \mathbf{K}_x^2 \boldsymbol{\alpha}_x + \tau_x \boldsymbol{\alpha}_x^\top \mathbf{K}_x \boldsymbol{\alpha}_x = 1, \\ &(1-\tau_u)\boldsymbol{\alpha}_u^\top \mathbf{K}_u^2 \boldsymbol{\alpha}_u + \tau_u \boldsymbol{\alpha}_u^\top \mathbf{K}_u \boldsymbol{\alpha}_u = 1. \end{aligned} \tag{16.49}$$

1.16.3.3 Kernel Fisher discriminant analysis

The Fisher discriminant is a classification function

$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^\top \mathbf{x} + b), \tag{16.50}$$

where the weight vector $\mathbf{w}$ is found through a specific optimization to well separate different classes. In particular, a direction is found which maximizes the distance between projected class means and simultaneously minimizes the projected class variances. In this article, the binary case is considered. The parameter b in the Fisher discriminant is usually determined by projecting training data to $\mathbf{w}$ and then identifying the middle point of two class means.

Suppose examples from two different classes are given by $S_1 = \{\mathbf{x}_1^1, \ldots, \mathbf{x}_{m_1}^1\}$ and $S_2 = \{\mathbf{x}_1^2, \ldots, \mathbf{x}_{m_2}^2\}$. Fisher discriminant analysis [32,33] finds $\mathbf{w}$ which maximizes

$$J(\mathbf{w}) = \frac{\mathbf{w}^\top \mathbf{S}_B \mathbf{w}}{\mathbf{w}^\top \mathbf{S}_W \mathbf{w}}, \tag{16.51}$$

where

$$\begin{aligned} \mathbf{S}_B &= (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^\top, \\ \mathbf{S}_W &= \sum_{i=1,2} \sum_{\mathbf{x} \in S_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^\top \end{aligned} \tag{16.52}$$

are respectively the between and within class scatter matrices and $\mathbf{m}_i$ is defined by $\mathbf{m}_i = \frac{1}{m_i} \sum_{j=1}^{m_i} \mathbf{x}_j^i$. The solution is the eigenvector corresponding to the largest eigenvalue of the generalized eigendecomposition

$$\mathbf{S}_B \mathbf{w} = \lambda \mathbf{S}_W \mathbf{w}. \tag{16.53}$$

Since the matrix $\mathbf{S}_B$ has rank 1, only the leading eigenvector contains meaningful information.

Let ϕ be a nonlinear map to some feature space F. Kernel Fisher discriminant analysis attempts to find a direction $\mathbf{w} \in F$ to maximize

$$J(\mathbf{w}) = \frac{\mathbf{w}^\top \mathbf{S}_B^\phi \mathbf{w}}{\mathbf{w}^\top \mathbf{S}_W^\phi \mathbf{w}}, \tag{16.54}$$

where

$$\begin{aligned} \mathbf{S}_B^\phi &= (\mathbf{m}_1^\phi - \mathbf{m}_2^\phi)(\mathbf{m}_1^\phi - \mathbf{m}_2^\phi)^\top, \\ \mathbf{S}_W^\phi &= \sum_{i=1,2} \sum_{\mathbf{x} \in S_i} \left(\phi(\mathbf{x}) - \mathbf{m}_i^\phi\right) \left(\phi(\mathbf{x}) - \mathbf{m}_i^\phi\right)^\top \end{aligned} \tag{16.55}$$

with $\mathbf{m}_i^\phi = \frac{1}{m_i} \sum_{j=1}^{m_i} \phi(\mathbf{x}_j^i)$.

Define $S = S_1 \cup S_2$ and denote its elements by $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ with $m = m_1 + m_2$. We would like to find an expansion for $\mathbf{w}$ in the form $\mathbf{w} = \sum_{i=1}^m \alpha_i \phi(\mathbf{x}_i)$. It follows that

$$\mathbf{w}^\top \mathbf{m}_i^\phi = \frac{1}{m_i} \sum_{j=1}^{m} \sum_{k=1}^{m_i} \alpha_j \kappa(\mathbf{x}_j, \mathbf{x}_k^i) = \boldsymbol{\alpha}^\top \mathbf{M}_i, \tag{16.56}$$

where vector $\mathbf{M}_i$ is defined as $(\mathbf{M}_i)_j = \frac{1}{m_i} \sum_{k=1}^{m_i} \kappa(\mathbf{x}_j, \mathbf{x}_k^i)$ and the dot products are replaced with kernels [33]. Based on (16.56), the numerator of (16.54) can be rewritten as

$$\mathbf{w}^\top \mathbf{S}_B^\phi \mathbf{w} = \boldsymbol{\alpha}^\top \mathbf{M} \boldsymbol{\alpha}, \tag{16.57}$$

where $\mathbf{M} = (\mathbf{M}_1 - \mathbf{M}_2)(\mathbf{M}_1 - \mathbf{M}_2)^\top$. And the denominator is rewritten as

$$\mathbf{w}^\top \mathbf{S}_W^\phi \mathbf{w} = \boldsymbol{\alpha}^\top \mathbf{N} \boldsymbol{\alpha}, \tag{16.58}$$

where $\mathbf{N} = \sum_{j=1,2} \mathbf{K}_j (\mathbf{I} - \mathbf{1}_{m_j}) \mathbf{K}_j^\top$, $\mathbf{K}_j$ is an $m \times m_j$ matrix with $(\mathbf{K}_j)_{ik} = \kappa(\mathbf{x}_i, \mathbf{x}_k^j)$, $\mathbf{I}$ is the identity matrix and $\mathbf{1}_{m_j}$ is the matrix with all entries $\frac{1}{m_j}$ [33].

Hence, (16.54) is reformulated as

$$J(\boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha}^\top \mathbf{M} \boldsymbol{\alpha}}{\boldsymbol{\alpha}^\top \mathbf{N} \boldsymbol{\alpha}}. \tag{16.59}$$

The problem can be solved similarly to (16.51). To enhance numerical stability and perform capacity control in the feature space, $\mathbf{N}$ in the above formulation is usually replace by $\mathbf{N} + \mu \mathbf{I}$ with positive μ. An alternative regularization is penalizing $\|\mathbf{w}\|^2$ as in kernel CCA instead of the current $\|\boldsymbol{\alpha}\|^2$ which corresponds to the term $\mu \mathbf{I}$.

1.16.3.4 SVMs for classification and regression

Given the training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ of points $\mathbf{x}_i \in \mathbb{R}^n$ with corresponding labels $y_i \in \{1, -1\}$, SVM classifiers attempt to find a classification hyperplane induced from the maximum

margin principle [7,8]. In real applications data are usually not linearly separable. Thus a loss on the violation of the linearly separable constraints has to be introduced. A common choice is the hinge loss

$$\max\left(0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)\right), \tag{16.60}$$

which can be represented by a slack variable ξ_i.

The optimization problem for SVM classification is formulated as

$$\begin{aligned} &\min_{\mathbf{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\xi_i \\ &\text{s.t.} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad i = 1, \ldots, m, \\ &\xi_i \geq 0, \quad i = 1, \ldots, m, \end{aligned} \tag{16.61}$$

where the scalar C controls the balance between the margin and empirical loss, and $\boldsymbol{\xi} = [\xi_1, \ldots, \xi_m]^\top$. The large margin principle is reflected by minimizing $\frac{1}{2}\|\mathbf{w}\|^2$ with $2/\|\mathbf{w}\|$ being the margin between two hyperplanes $\mathbf{w}^\top \mathbf{x} + b = 1$ and $\mathbf{w}^\top \mathbf{x} + b = -1$ (For the linearly separable case, the concepts of the margin and classification hyperplane are illustrated in Figure 16.2). The SVM classifier would be

$$c_{\mathbf{w},b}(\mathbf{x}) = \text{sign}(\mathbf{w}^\top \mathbf{x} + b). \tag{16.62}$$

The Lagrangian of problem (16.61) is

$$\begin{aligned} L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\lambda}, \boldsymbol{\gamma}) = &\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\lambda_i\left[y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1 + \xi_i\right] \\ &- \sum_{i=1}^{m}\gamma_i\xi_i, \quad \lambda_i \geq 0, \ \gamma_i \geq 0, \end{aligned} \tag{16.63}$$

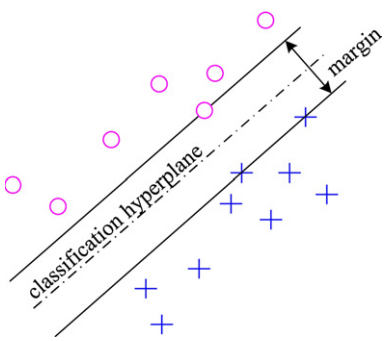

FIGURE 16.2

An illustration of the margin and classification hyperplane for the linearly separable binary case.

where $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_m]^\top$ and $\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_m]^\top$ are the associated Lagrange multipliers. Using the superscript star to denote the solutions of the optimization problem, according to the KKT (Karush-Kuhn-Tucker) conditions [34,35], we obtain

$$\partial_{\mathbf{w}} L(\mathbf{w}^*, b^*, \boldsymbol{\xi}^*, \boldsymbol{\lambda}^*, \boldsymbol{\gamma}^*) = \mathbf{w}^* - \sum_{i=1}^{m} \lambda_i^* y_i \mathbf{x}_i = 0, \tag{16.64}$$

$$\partial_b L(\mathbf{w}^*, b^*, \boldsymbol{\xi}^*, \boldsymbol{\lambda}^*, \boldsymbol{\gamma}^*) = -\sum_{i=1}^{m} \lambda_i^* y_i = 0, \tag{16.65}$$

$$\partial_{\xi_i} L(\mathbf{w}^*, b^*, \boldsymbol{\xi}^*, \boldsymbol{\lambda}^*, \boldsymbol{\gamma}^*) = C - \lambda_i^* - \gamma_i^* = 0, \quad i = 1, \ldots, m. \tag{16.66}$$

From (16.64), the solution $\mathbf{w}^*$ has the form

$$\mathbf{w}^* = \sum_{i=1}^{m} \lambda_i^* y_i \mathbf{x}_i. \tag{16.67}$$

Since examples with $\lambda_i^* = 0$ can be omitted from the expression, the training examples for which $\lambda_i^* > 0$ are called support vectors.

By substituting (16.64)–(16.66) into the Lagrangian, we can finally get the dual optimization problem [35]

$$\begin{aligned} \max_{\boldsymbol{\lambda}} \quad & \boldsymbol{\lambda}^\top \mathbf{j} - \frac{1}{2} \boldsymbol{\lambda}^\top D \boldsymbol{\lambda} \\ \text{s.t.} \quad & \boldsymbol{\lambda}^\top \mathbf{y} = 0, \\ & \boldsymbol{\lambda} \succeq \mathbf{0}, \\ & \boldsymbol{\lambda} \preceq C\mathbf{j}, \end{aligned} \tag{16.68}$$

where $\mathbf{j}$ is the vector with all entries being 1, $\mathbf{y} = [y_1, \ldots, y_m]^\top$ and D is a symmetric $m \times m$ matrix with entries $D_{ij} = y_i y_j \mathbf{x}_i^\top \mathbf{x}_j$ [1,36].

The complementary slackness condition (also called the zero KKT-gap requirement) [6] implies

$$\begin{aligned} \lambda_i^* \left[y_i (\mathbf{x}_i^\top \mathbf{w}^* + b^*) - 1 + \xi_i^* \right] = 0, \quad & i = 1, \ldots, m, \\ \gamma_i^* \xi_i^* = 0, \quad & i = 1, \ldots, m. \end{aligned} \tag{16.69}$$

Combining (16.66) and (16.69), we can solve $b^* = y_i - \mathbf{x}_i^\top \mathbf{w}^*$ for any support vector $\mathbf{x}_i$ with $0 < \lambda_i^* < C$. The existence of $0 < \lambda_i^* < C$ is a reasonable assumption, though there lacks a rigorous justification [36]. Once $\boldsymbol{\lambda}^*$ and b^* are solved, the SVM classifier is given by

$$c^*(\mathbf{x}) = \operatorname{sign}\left(\sum_{i=1}^{m} y_i \lambda_i^* \mathbf{x}^\top \mathbf{x}_i + b^* \right). \tag{16.70}$$

Using the kernel trick, the optimization problem (16.68) for SVMs becomes

$$\max_{\boldsymbol{\lambda}} \quad \boldsymbol{\lambda}^\top \mathbf{j} - \frac{1}{2} \boldsymbol{\lambda}^\top D \boldsymbol{\lambda}$$

$$\begin{aligned} &\text{s.t.} \quad \boldsymbol{\lambda}^\top \mathbf{y} = 0, \\ &\boldsymbol{\lambda} \succeq \mathbf{0}, \\ &\boldsymbol{\lambda} \preceq C\mathbf{j}, \end{aligned} \tag{16.71}$$

where the entries of D are $D_{ij} = y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j)$. The solution for the corresponding SVM classifier is formulated as

$$c^*(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{m} y_i \lambda_i^* \kappa(\mathbf{x}_i, \mathbf{x}) + b^*\right). \tag{16.72}$$

For regression problems, the labels in the training set S are real numbers, that is $y_i \in \mathbb{R}$ $(i = 1, \ldots, m)$. In order to induce a sparse representation for the decision function (i.e., some training examples can be ignored), [8] devised the following ϵ-insensitive function and applied it to support vector regression

$$|y - f(\mathbf{x})|_\epsilon = \max\{0, |y - f(\mathbf{x})| - \epsilon\}, \quad \epsilon \geq 0. \tag{16.73}$$

The standard form of support vector regression is to minimize

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} |y_i - f(\mathbf{x}_i)|_\epsilon, \tag{16.74}$$

where the positive scalar C reflects the trade-off between the margin and the empirical loss. An equivalent optimization that is commonly used is

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\xi}^*} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \xi_i + C\sum_{i=1}^{m} \xi_i^* \\ \text{s.t.} \quad & \langle \mathbf{w}, \phi(\mathbf{x}_i)\rangle + b - y_i \leq \epsilon + \xi_i, \\ & y_i - \langle \mathbf{w}, \phi(\mathbf{x}_i)\rangle - b \leq \epsilon + \xi_i^*, \\ & \xi_i, \xi_i^* \geq 0, \quad i = 1, \ldots, m, \end{aligned} \tag{16.75}$$

where $\phi(\mathbf{x}_i)$ is the image of $\mathbf{x}_i$ in the feature space, and $\boldsymbol{\xi}, \boldsymbol{\xi}^*$ are defined similarly as before. The prediction output of support vector regression is

$$c^*(\mathbf{x}) = \langle \mathbf{w}^*, \phi(\mathbf{x})\rangle + b^*, \tag{16.76}$$

where $\mathbf{w}^*$ and b^* are the solution of (16.75). For support vector regression, the derivation for the dual representation of solutions and the dual optimization problem can consult the counterpart for classification, and thus is omitted here.

1.16.3.5 Bayesian kernel methods: Gaussian processes

All the previous methods introduced in this section can be summarized into the framework of risk minimization. The Bayesian learning approach differs from them in several aspects. The key distinction is that the Bayesian approach intuitively incorporates prior knowledge into the process of estimation [6]. Another benefit of the Bayesian framework is the possibility of measuring the confidence of the estimation in a straightforward manner. However, algorithms designed by the Bayesian approach (e.g., with maximum a posterior estimation) can have similar counterparts originating from the risk minimization

framework. Below we focus on the Gaussian process approach for regression, which is a classical Bayesian kernel method.

The Gaussian process models have two kinds of equivalent representations, namely the function-space view and the weight-space view [37]. We will start with the weight-space view to illustrate the explicit roles of kernels using the Bayesian treatment of linear regression, followed by a very brief introduction of the function-space view.

Suppose the training set S is $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ as defined in Section 1.16.2.1. The standard linear regression model with Gaussian noise is

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}, \quad y = f(\mathbf{x}) + \varepsilon, \tag{16.77}$$

where f is the function value, y is the noisy observed value, and the noise obeys an independent, identically distributed Gaussian distribution with mean zero and variance σ_n^2

$$\varepsilon \sim \mathcal{N}(0, \sigma_n^2). \tag{16.78}$$

This gives rise to the likelihood of the independent observations in the training set

$$\begin{aligned} p(\mathbf{y}|\mathbf{X}, \mathbf{w}) &= \prod_{i=1}^{m} p(y_i|\mathbf{x}_i, \mathbf{w}) = \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma_n} \exp\left(-\frac{(y_i - \mathbf{w}^\top \mathbf{x}_i)^2}{2\sigma_n^2}\right) \\ &= \frac{1}{(2\pi\sigma_n^2)^{m/2}} \exp\left(-\frac{\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2}{2\sigma_n^2}\right) = \mathcal{N}(\mathbf{X}\mathbf{w}, \sigma_n^2 \mathbf{I}), \end{aligned} \tag{16.79}$$

where $\mathbf{y} = [y_1, \ldots, y_m]^\top$ and $\mathbf{X}^\top = [\mathbf{x}_1, \ldots, \mathbf{x}_m]$. Suppose we specify a Gaussian prior on the parameters with mean zero and covariance matrix Σ_p [37]

$$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma_p). \tag{16.80}$$

According to Bayes' rule, the posterior of the parameters is proportional to the product of the prior and likelihood

$$\begin{aligned} p(\mathbf{w}|\mathbf{X}, \mathbf{y}) &\propto \exp(-\frac{1}{2}\mathbf{w}^\top \Sigma_p^{-1}\mathbf{w}) \exp\left(-\frac{1}{2\sigma_n^2}(\mathbf{y} - \mathbf{X}\mathbf{w})^\top (\mathbf{y} - \mathbf{X}\mathbf{w})\right) \\ &\propto \exp\left(-\frac{1}{2}(\mathbf{w} - \bar{\mathbf{w}})^\top A (\mathbf{w} - \bar{\mathbf{w}})\right), \end{aligned} \tag{16.81}$$

where $A = \frac{1}{\sigma_n^2}\mathbf{X}^\top \mathbf{X} + \Sigma_p^{-1}$, and $\bar{\mathbf{w}} = \frac{1}{\sigma_n^2} A^{-1} \mathbf{X}^\top \mathbf{y}$. It tends out that the posterior is a Gaussian distribution with mean $\bar{\mathbf{w}}$ and covariance A^{-1}.

The predictive distribution for a test example $\mathbf{x}$ is given by averaging the outputs of all possible linear models from the above Gaussian posterior

$$\begin{aligned} p(f(\mathbf{x})|\mathbf{x}, \mathbf{X}, \mathbf{y}) &= \int p(f(\mathbf{x})|\mathbf{x}, \mathbf{w}) p(\mathbf{w}|\mathbf{X}, \mathbf{y}) d\mathbf{w} \\ &= \mathcal{N}\left(\frac{1}{\sigma_n^2}\mathbf{x}^\top A^{-1}\mathbf{X}^\top \mathbf{y}, \mathbf{x}^\top A^{-1}\mathbf{x}\right). \end{aligned} \tag{16.82}$$

Now suppose we use a function $\phi(\cdot)$ to map the inputs in the original space to a feature space, and perform linear regression there. The predictive distribution would be

$$p(f(\mathbf{x})|\mathbf{x}, \mathbf{X}, \mathbf{y}) = \mathcal{N}\left(\frac{1}{\sigma_n^2}\phi(\mathbf{x})^\top A^{-1}\Phi^\top \mathbf{y}, \phi(\mathbf{x})^\top A^{-1}\phi(\mathbf{x})\right), \tag{16.83}$$

where $\Phi^\top = [\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_m)]$, and $A = \frac{1}{\sigma_n^2}\Phi^\top\Phi + \Sigma_p^{-1}$. Using matrix transformations such as the matrix inversion lemma, we can rewrite (16.83) as

$$\begin{aligned} p(f(\mathbf{x})|\mathbf{x}, \mathbf{X}, \mathbf{y}) = \mathcal{N}\Big(&\phi(\mathbf{x})^\top\Sigma_p\Phi^\top(K + \sigma_n^2\mathbf{I})^{-1}\mathbf{y}\,, \\ &\phi\,(\mathbf{x})^\top\Sigma_p\phi(\mathbf{x}) - \phi(\mathbf{x})^\top\Sigma_p\Phi^\top(K + \sigma_n^2\mathbf{I})^{-1}\Phi\Sigma_p\phi(\mathbf{x})\Big), \end{aligned} \tag{16.84}$$

where $K = \Phi\Sigma_p\Phi^\top$ [37]. Notice that in the above formulation the terms related to the images in the feature space can be represented in the form of $\phi(\mathbf{x})^\top\Sigma_p\phi(\mathbf{x}')$ with $\mathbf{x}$ and $\mathbf{x}'$ in either the training or test sets [37]. Define $\kappa(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top\Sigma_p\phi(\mathbf{x}')$. By Theorem 7, we know that $\kappa(\mathbf{x}, \mathbf{x}')$ is a valid kernel function. In the Gaussian process literature, it is often called the covariance function.

The function-space view of the Gaussian processes is given by the following definition which describes a distribution over functions [37].

Definition 8 (Gaussian processes). A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.

A Gaussian process is specified by its mean function and covariance function. If we define the mean function $m(\mathbf{x})$ and the covariance function $k(\mathbf{x}, \mathbf{x}')$ of a real process $f(\mathbf{x})$ as

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})], \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}\left[\big(f(\mathbf{x}) - m(\mathbf{x})\big)\big(f(\mathbf{x}') - m(\mathbf{x}')\big)\right], \end{aligned} \tag{16.85}$$

the Gaussian process can be written as

$$f(\mathbf{x}) \sim \mathcal{GP}\left(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\right). \tag{16.86}$$

Figure 16.3 shows samples of functions drawn from a specific Gaussian process.

The Bayesian linear regression model $f(\mathbf{x}) = \mathbf{w}^\top\phi(\mathbf{x})$ with parameter prior $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma_p)$ can be cast into the above function-space view. It is simple to see that the function values $f(\mathbf{x}_1), \ldots, f(\mathbf{x}_q)$ corresponding to any number of inputs q are jointly Gaussian, and the mean and covariance are given by

$$\begin{aligned} \mathbb{E}[f(\mathbf{x})] &= \mathbb{E}[\mathbf{w}^\top]\phi(\mathbf{x}) = 0, \\ \mathbb{E}[f(\mathbf{x})f(\mathbf{x}')] &= \phi(\mathbf{x})^\top\mathbb{E}[\mathbf{w}\mathbf{w}^\top]\phi(\mathbf{x}') = \phi(\mathbf{x})^\top\Sigma_p\phi(\mathbf{x}'), \end{aligned} \tag{16.87}$$

where the second equation recovers our definition of the kernel function for the weight-space view. In other words, now $m(\mathbf{x}) = 0$ and $k(\mathbf{x}, \mathbf{x}') = \kappa(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top\Sigma_p\phi(\mathbf{x}')$.

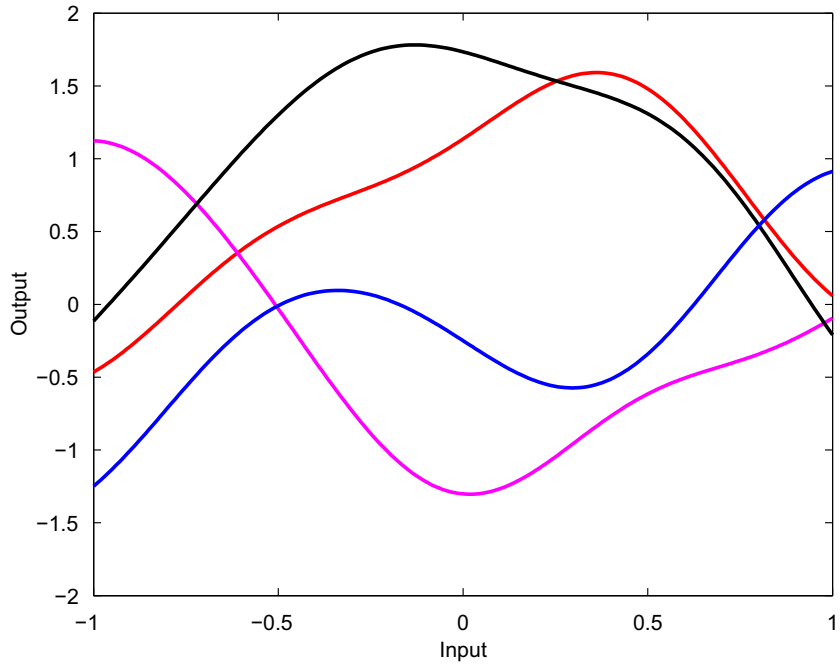

FIGURE 16.3

Samples from a Gaussian process with zero mean and a Gaussian kernel as the covariance function.

1.16.4 Computational issues of kernel methods

Implementation of kernel methods often involve eigen-decomposition of kernel matrices or inversion of the sum of a kernel matrix and a scaled identity matrix. The computational complexity of this operation is typically $O(m^3)$ with m being the number of training examples. This can be very demanding for large training sets, and therefore approximation algorithms are desirable.

Good low-rank approximations of kernel matrices are usually enough to deal with the above problems. For example, the matrix inversion lemma can use the low-rank decomposition to invert the sum of the kernel matrix and a scaled identity matrix efficiently. Eigen-decomposition of kernel matrices can also be converted to do the same operation on much smaller matrices. Here we just give a pointer to some of the approximation methods. Interested readers can refer to the corresponding literature for detailed implementation techniques.

Partial Gram-Schmidt orthogonalization [38] and incomplete Cholesky decomposition [27] are good approaches for finding low-rank approximations of kernel matrices. These two approaches are essentially equivalent, since performing a Cholesky decomposition of the kernel matrix is equivalent to performing

Gram-Schmidt orthogonalization in the feature space [1]. In other words, incomplete Cholesky decomposition can be viewed as the dual implementation of the partial Gram-Schmidt orthogonalization.

The sparse greedy matrix approximation [39] and the Nyström approximation [40] are two alternative approaches. The idea of the former is to select a collection of basis functions to obtain an approximate kernel matrix $\widetilde{K}$ whose distance to the original kernel matrix K is small. The Nyström approximation is much simpler, which randomly chooses r rows/columns of K without replacement, and then sets $\widetilde{K} = K_{m,r}K_{r,r}^{-1}K_{r,m}$, where $K_{m,r}$ is the $m \times r$ block of K and similar definitions apply to the other blocks. For a given r, the sparse greedy matrix approximation produces a better approximation to K, but computationally more demanding [40].

1.16.5 Multiple kernel learning

In practical problems, for a given decision-making task, there can be multiple different data sources. These data can be heterogeneous, which means that they represent different properties (e.g., visual features or lingual features) or have different forms (e.g., continuous value or discrete value). Consequently, using a different kernel to account for each of the data sources and then combining them is sensible. In other cases, even if the data are homogeneous, we may still want to adopt multiple kernels to capture more information. This problem of learning a combination of multiple kernels is termed multiple kernel learning [17] and now is an active research topic [41–47]. Here we review some multiple kernel learning methods, several of which have sparsity regularizations [48,49], to provide a brief outline of the research progress.

The kernel learning approach proposed by [17] is to add to the original optimization problems some extra constraints on the symmetric kernel matrix K, e.g., by

$$\begin{aligned} &K = \sum_{i=1}^{t} \mu_i K_i, \\ &\mu_i \in \mathbb{R}, \quad i = 1, \ldots, t, \\ &K \succeq 0, \\ &\text{trace}(K) \leq c, \end{aligned} \tag{16.88}$$

or

$$\begin{aligned} &K = \sum_{i=1}^{t} \mu_i K_i, \\ &\mu_i \geq 0, \quad i = 1, \ldots, t, \\ &K \succeq 0, \\ &\text{trace}(K) \leq c, \end{aligned} \tag{16.89}$$

where t is the number of individual kernels, and then formulate the problem in terms of semidefinite programming (SDP) [34,35]. The advantages of the second set of constraints over the first include reducing computational burden and facilitating the study of some statistical properties of kernel matrices [17].

However, the learning problem would become intractable when the number of training examples or kernels grow large.

[48] reformulated the problem and proposed a sequential minimal optimization (SMO) algorithm to improve the efficiency. They used second-order cone programming (SOCP) and Moreau-Yosida regularization to derive the SMO algorithm and made multiple kernel learning applicable for medium-scale problems. The corresponding KKT conditions not only lead to support vectors, but also to "support kernels" which means a sparse combination of candidate kernels can be expected. [50] adopted semi-infinite liner programming (SILP) to formulate the multiple kernel learning problem, based on which they iteratively solved a standard SVM problem with a single kernel and a linear program whose constraints increase with iterations. This approach makes multiple kernel learning applicable to large-scale problems. Later, [49] further improved the efficiency by using a formulation of a weighted 2-norm regularization with sparsity considerations imposed on the weights. Evidence show that it is globally faster than the mentioned SILP approach but with more kernels selected.

[41] considered multiple kernel learning with infinite number of basic kernels. In particular, kernels are selected from the convex hull of continuously parameterized kernels. Making use of the conjugate function and von Neumann minimax theorem [44], they adopted a greedy algorithm to solve the optimization problem, where the DC (difference of convex functions) programming techniques that attempt to optimize a non-convex function by the difference of two convex functions [51] were used to optimize a subroutine of the algorithm. Experimental results indicated the advantage of working with a continuous parameterization over a predesignated finite number of basic kernels.

For Bayesian multiple kernel learning, recently, [52] proposed a new variational approximation for infinite mixtures of Gaussian processes. The mixtures of Gaussian processes have the advantages of characterizing varying covariances or multimodal data and reducing the cubic computational complexity of the single Gaussian process model [53–55]. They used mean field variational inference and a truncated stick-breaking representation of the Dirichlet process to approximate the posterior of latent variables, and applied the approach to traffic prediction problems.

1.16.6 Applications

The applications of kernel methods and SVMs are rather broad. Here we just list some of its typical applications.

Biometrics refers to the identification of humans based on their physical or behavioral traits, which can be used for access control. Typical methods for biometrics include face recognition, signature recognition and EEG-based biometrics [56]. Over the past years, kernel methods and SVMs have been successfully applied to this field [57,58].

Intelligent transportation systems are an important application platform for machine learning techniques. Representative applications include pedestrian recognition, traffic flow forecasting, and traffic bottleneck identification. Kernel methods including Gaussian processes have achieved very good performance in this area [52,59].

Research on brain-computer interfaces which aim to enable severely disabled people to drive communication or control devices, arouses many interests recently. The discrimination of different brain

signals is essentially a pattern classification problem, where SVMs have been shown to be a very useful tool [60,61].

Natural language processing, e.g., text classification and retrieval, is an active research field which has used a lot of machine learning methods. Kernel techniques applied to this task include kernel design, supervised classification and semi-supervised classification [14,62,63].

1.16.7 Open issues and problems

In this article, we have presented some key techniques for using kernel methods, such as how to derive the dual formulation of an original method, what are essential conditions for valid kernels, typical kernel functions, and how to construct new kernels. This constitutes the foundations of kernel methods. Then, we introduced some fundamental kernel methods which are well-known and now used widely for unsupervised or supervised learning. In particular, as a representative of Bayesian kernel methods, Gaussian processes were introduced.

The computational complexity of kernel methods is usually cubic with respect to the number of training examples. Therefore, reducing the computational costs has been an important research topic. For this problem, we briefly pointed out four methods—partial Gram-Schmidt orthogonalization, incomplete Cholesky decomposition, sparse greedy matrix approximation and the Nyström approximation, and explained the idea on why they can be used to alleviate the computational burden.

In addition, we have introduced the recent developments on multiple kernel learning which has shown its merit over single kernel learning in the past few years. However, for multiple kernel learning, we have to learn both the combination coefficients for candidate kernels and other parameters inherited from traditional single kernel learning. Therefore, there are various efforts to reformulate the optimization problem to accelerate learning, and indeed people have achieved some encouraging results.

Studies on kernel methods can be further deepened in different aspects, e.g., the above mentioned multiple kernel learning, and combining kernel techniques with other machine learning mechanisms. Another line of important open problems would be performing theoretical analysis on the generalization errors of newly emerging kernel methods, such as the multitask SVMs and multitask multiclass SVMs. We hope this article is helpful to promote the applications and theoretical developments of kernel methods in the future.

Glossary

Canonical correlation analysis	A method to find two linear transformations respectively for two representations such that the correlations between the transformed variables are maximized
Fisher discriminant analysis	A method for classification which seeks a direction to maximize the distance between projected class means and simultaneously minimize the projected class variances
Gaussian process	A collection of random variables, any finite number of which have a joint Gaussian distribution

Kernel trick	A method to extend any algorithm only involving computations of inner products from the original space to a feature space with kernel functions
Multiple kernel learning	A learning mechanism which aims to learn a combination of multiple kernels to capture more information or reduce the computational complexity
Principal component analysis	A method to find a set of orthogonal directions which forms a subspace to maximize data variances along the directions in this subspace
Reproducing kernel Hilbert space	A function space which is a Hilbert space possessing a reproducing kernel
Support vector machine	A method to learn a hyperplane induced from the maximum margin principle, which has wide applications including classification and regression

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Project 61075005, the Fundamental Research Funds for the Central Universities, and the PASCAL2 Network of Excellence. This publication only reflects the authors' views.

References

[1] J. Shawe-Taylor, N. Cristianini, Kernel Methods for Pattern Analysis, Cambridge University Press, Cambridge, UK, 2004.
[2] F. Rosenblatt, The perceptron: A probabilistic model for information storage and organization in the brain, Psychol. Reviews 65 (1958) 386–408.
[3] J. Hertz, A. Krogh, R. Palmer, Introduction to the Theory of Neural Computation, Addison-Wesley, Reading, MA, 1991.
[4] L. Breiman, J. Friedman, R. Olshen, C. Stone, Classification and Regression Trees, Wadsworth International, Belmont, CA, 1984.
[5] J. Quinlan, C4.5: programs for Machine Learning, San Mateo, California: Morgan Kauffmann, 1993.
[6] B. Schölkopf, A. Smola, Learning with Kernels, MIT Press, Cambridge, MA, 2002.
[7] B. Boser, I. Guyon, V. Vapnik, A training algorithm for optimal margin classifier, in: Proceedings of the 5th ACM Worksop on Computational Learning Theory, 1992, pp. 144–152.
[8] V. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag, New York, 1995.
[9] M. Aizerman, E. Braverman, L. Rozonoer, Theoretical foundations of the potential function method in pattern recognition learning, Autom. Remote Control 25 (1964) 821–837.
[10] R. Duda, P. Hart, Pattern Classification and Scene Analysis, John Wiley & Sons, New York, 1973.
[11] T. Evgeniou, M. Pontil, Regularized multi-task learning, in: Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2004, pp. 109–117.
[12] J. Farquhar, D. Hardoon, H. Meng, J. Shawe-Taylor, S. Szedmak, Two view learning: SVM-2K, theory and practice, Adv. Neural Inform. Process. Syst. 18 (2006) 355–362.

[13] D. Rosenberg, V. Sindhwani, P. Bartlett, P. Niyogi, Multiview point cloud kernels for semisupervised learning, IEEE Signal Process. Mag. 26 (2009) 145–150.
[14] S. Sun, J. Shawe-Taylor, Sparse semi-supervised learning using conjugate functions, J. Mach. Learn. Res. 11 (2010) 2423–2455.
[15] Y. Ji, S. Sun, Multitask multiclass support vector machines, in: Proceedings of the IEEE International Conference on Data Mining Workshops, 2011, pp. 512–518.
[16] S. Sun, Multi-view Laplacian support vector machines, Lect. Notes Comput. Sci. 7121 (2011) 209–222.
[17] G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, M. Jordan, Learning the kernel matrix with semidefinite programming, J. Mach. Learn. Res. 5 (2004) 27–72.
[18] C. Bishop, Pattern Recognition and Machine Learning, Springer-Verlag, New York, 2006.
[19] R. Duda, P. Hart, D. Stork, Pattern Classification, John Wiley & Sons, New York, 2001.
[20] S. Theodoridis, K. Koutroumbas, Pattern Recognition, Academic Press 2008.
[21] N. Aronszajn, Theory of reproducing kernels, Trans. Am. Math. Soc. 68 (1950) 337–404.
[22] J. Mercer, Functions of positive and negative type and their connection with the theory of integral equations, Philos. Trans. Roy. Soc., London, A 209 (1909) 415–446.
[23] T. Jaakkola, D. Haussler, Exploiting generative models in discriminative classifiers, Adv. Neural Inform. Process. Syst. 11 (1999a) 487–493.
[24] T. Jaakkola, D. Haussler, Probabilistic kernel regression models, in: Proceedings of the International Conference on AI and, Statistics, 1999b, pp. 1–9.
[25] B. Schölkopf, A. Smola, K. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Comput. 10 (1998) 1299–1319.
[26] H. Hotelling, Relations between two sets of variates, Biometrika 28 (1936) 321–377.
[27] F. Bach, M. Jordan, Kernel independent component analysis, J. Mach. Learn. Res. 3 (2002) 1–48.
[28] J. Kettenring, Canonical analysis of several sets of variables, Biometrika 58 (1971) 433–451.
[29] S. Akaho, A kernel method for canonical correlation analysis, in: Proceedings of the International Meeting of the Psychometric Society 2001.
[30] C. Fyfe, P. Lai, ICA using kernel canonical correlation analysis, in: Proceedings of the International Workshop on Independent Component Analysis and Blind Singal Separation, 2000, pp. 279–284.
[31] T. Melzer, M. Reiter, H. Bischof, Nonlinear feature extraction using generalized canonical correlation analysis, in: Proceedings of the International Conference on Artificial, Neural Networks, 2001, pp. 353–360.
[32] K. Fukunaga, Introduction to Statistical Pattern Recognition, Academic Press, San Diego, CA, 1990.
[33] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, K. Müller, Fisher discriminant analysis with kernels, Neural Netw. Signal Process. IX (1999) 41–48.
[34] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, England, 2004.
[35] J. Shawe-Taylor, S. Sun, A review of optimization methodologies in support vector machines, Neurocomputing 74 (2011) 3609–3618.
[36] E. Osuna, R. Freund, F. Girosi, Support vector machines: training and applications, Technical Report AIM-1602, Massachusetts Institute of Technology, MA, 1997.
[37] C. Rasmussen, C. Williams, Gaussian Processes for Machine Learning, MIT Press, Cambridge, MA, 2006.
[38] D. Hardoon, S. Szedmak, J. Shawe-Taylor, Canonical correlation analysis: an overview with application to learning methods, Neural Comput. 16 (2004) 2639–2664.
[39] A. Smola, B. Schölkopf, Sparse greedy matrix approximation for machine learning, in: Proceedings of the 17th International Conference on, Machine learning, 2000, pp. 911–918.
[40] C. Williams, M. Seeger, Using the Nyström method to speed up kernel machines, Adv. Neural Inform. Process. Syst. 13 (2001) 682–688.
[41] A. Argyriou, R. Hauser, C. Micchelli, M. Pontil, A DC-programming algorithm for kernel selection, in: Proceedings of the 23rd International Conference on, Machine Learning, 2006, pp. 41–48.

[42] M. Girolami, S. Rogers, Hierarchic Bayesian models for kernel learning, in: Proceedings of the 22nd International Conference on, Machine Learning, 2005, pp. 241–248.
[43] J. Li, S. Sun, Nonlinear combination of multiple kernels for support vector machines, in: Proceedings of the 20th International Conference on, Pattern Recognition, 2010, pp. 1–4.
[44] A. Micchelli, M. Pontil, Learning the kernel function via regularization, J. Mach. Learn. Res. 6 (2005) 1099–1125.
[45] C. Ong, A. Smola, R. Williamson, Learning the kernel with hyperkernels, J. Mach. Learn. Res. 6 (2005) 1043–1071.
[46] Y. Ying, D. Zhou, Learnability of Gaussians with flexible variances, J. Mach. Learn. Res. 8 (2007) 249–276.
[47] A. Zien, C. Ong, Multiclass multiple kernel learning, in: Proceedings of the 24th International Conference on, Machine Learning, 2007, pp. 1191–1198.
[48] F. Bach, G. Lanckriet, M. Jordan, Multiple kernel learning, conic duality and the SMO algorithm, in: Proceedings of the 21st International Conference on, Machine Learning, 2004, pp. 6–13.
[49] A. Rakotomamonjy, F. Bach, S. Canu, Y. Grandvalet, More efficiency in multiple kernel learning, in: Proceedings of the 24th International Conference on, Machine Learning, 2007, pp. 775–782.
[50] S. Sonnenburg, G. Raetsch, C. Schaefer, B. Schölkopf, Large scale multiple kernel learning, J. Mach. Learn. Res. 7 (2006) 1531–1565.
[51] R. Horst, V. Thoai, DC programming: Overview, J. Optimiz. Theory App. 103 (1999) 1–41.
[52] S. Sun, X. Xu, Variational inference for infinite mixtures of Gaussian processes with applications to traffic flow prediction, IEEE Trans. Intel. Transp. Syst. 12 (2011) 466–475.
[53] E. Meeds, S. Osindero, An alternative infinite mixture of Gaussian process experts, Adv. Neural Inform. Process. Syst. 18 (2006) 883–890.
[54] C. Rasmussen, Z. Ghahramani, Infinite mixtures of Gaussian process experts, Adv. Neural Inform. Process. Sys. 14 (2002) 881–888.
[55] V. Tresp, Mixtures of Gaussian processes, Adv. Neural Inform. Process. Syst. 13 (2001) 654–660.
[56] S. Sun, Multitask learning for EEG-based biometrics. in: Proceedings of the 19th International Conference on, Pattern Recognition, 2008, pp. 1–4.
[57] A. Tefas, C. Kotropoulos, I. Pitas, Using support vector machines to enhance the performance of elastic graph matching for frontal face authentication, IEEE Trans. Pattern Anal. Mach. Intel. 23 (2001) 735–746.
[58] E. Justino, F. Bortolozzi, R. Sabourin, A comparison of SVM and HMM classifiers in the off-line signature verification. Pattern Recogn. Lett. 26, 1377–1385.
[59] S. Munder, D. Gavrila, An experimental study on pedestrian classification, IEEE Trans. Pattern Anal. Mach. Intell. 28 (2006) 1863–1868.
[60] D. Garrett, D. Peterson, C. Anderson, M. Thaut, Comparison of linear, nonlinear, and feature selection methods for EEG signal classification, IEEE Trans. Neural Syst. Rehabil. Eng. 11 (2003) 141–144.
[61] S. Sun, C. Zhang, D. Zhang, An experimental evaluation of ensemble methods for EEG signal classification, Pattern Recogn. Lett. 28 (2007) 2157–2163.
[62] M. Collins, N. Duffy, Convolution kernels for natural language, Advances in Neural Information Processing Systems 14 (2001) 625–632.
[63] T. Joachims, Text categorization with support vector machines: Learning with many relevant features, Lect. Notes Comput. Sci. 1398 (1998) 137–142.