Kernel methods are a class of non-parametric learning techniques relying on kernels. A kernel generalizes dot products to arbitrary domains and can thus be seen as a similarity measure between data points with complex structures. The use of kernels allows to decouple the representation of the data from the specific learning algorithm, provided it can be defined in terms of distance or similarity between instances. Under this unifying formalism a wide range of methods have been developed, dealing with binary and multiclass classification, regression, ranking, clustering and novelty detection to name a few. Recent developments include statistical tests of dependency and alignments between related domains, such as documents written in different languages. Key to the success of any kernel method is the definition of an appropriate kernel for the data at hand. A well-designed kernel should capture the aspects characterizing similar instances while being computationally efficient. Building on the seminal work by D. Haussler on convolution kernels, a vast literature on kernels for structured data has arisen. Kernels have been designed for sequences, trees and graphs, as well as arbitrary relational data represented in first or higher order logic. From the representational viewpoint, this allowed to address one of the main limitations of statistical learning approaches, namely the difficulty to deal with complex domain knowledge. Interesting connections between the complementary fields of statistical and symbolic learning have arisen as one of the consequences. Another interesting connection made possible by kernels is between generative and discriminative learning. Here data are represented with generative models and appropriate kernels are built on top of them to be used in a discriminative setting.

In this chapter we revise the basic principles underlying kernel machines and describe some of the most popular approaches which have been developed. We give an extensive treatment of the literature on kernels for structured data and suggest some basic principles for developing novel ones. We finally discuss kernel methods
for predicting structures. These algorithms deal with structured-output prediction, a learning setting in which the output is itself a structure which has to be predicted from the input one.

1 A Gentle Introduction to Kernel Methods

In the typical statistical learning framework a supervised learning algorithm is given a training set of input-output pairs \( \mathcal{D} = \{(x_1, y_1), \ldots, (x_m, y_m)\} \), with \( x_i \in \mathcal{X} \) and \( y_i \in \mathcal{Y} \), sampled identically and independently from a fixed but unknown probability distribution \( \rho \). The set \( \mathcal{X} \) is called the input (or instance) space and can be any set. The set \( \mathcal{Y} \) is called the output (or target) space. For instance, in the case of binary classification \( \mathcal{Y} = \{-1, 1\} \) while the case of regression \( \mathcal{Y} \) is the set of real numbers. The learning algorithm outputs a function \( f: \mathcal{X} \mapsto \mathcal{Y} \) that approximates the probabilistic relation \( \rho \) between inputs and outputs. The class of functions that is searched is called the hypothesis space.

Intuitively, \( f \) should assign to a novel input \( x \) the same (or a similar) \( y \) of similar inputs already observed in the training set. A kernel is a function \( : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R} \) measuring the similarity between pairs of inputs. For example, the similarity between a pair of sequences could be the number of common subsequences of length up to a certain \( m \). The kernel should satisfy the following equation:

\[
k(x, x') = \langle \Phi(x), \Phi(x') \rangle.
\]

It thus corresponds to mapping examples to a (typically high dimensional) feature space \( \mathcal{H} \) and computing the dot product in that space. In the sequence example, \( \mathcal{H} \) is made of vectors of booleans, with an entry for each possible sequence of length up to \( m \) given the alphabet. \( \Phi(x) \) maps \( x \) to a vector with one for entries occurring in \( x \) and zero otherwise. However, the kernel function does not need to explicitly do the mapping (which can be even infinite dimensional) in computing the similarity.

**Fig. 1** A simple binary classifier in feature space: \( \mu_+ \) and \( \mu_- \) are the mean vectors of positive (circles) and negative (squares) examples respectively. The algorithm assigns a novel example to the class with the nearer mean. The decision boundary is a hyperplane (solid line) half-way down between the line linking the means (the dotted line).