# Chapter 5
# Support Vector and Kernel Methods

Nello Cristianini* and John Shawe-Taylor**
*University of California at Davis, USA
**Royal Holloway, University of London, UK

Kernel Methods (KM) are a relatively new family of algorithms that presents a series of useful features for pattern analysis in datasets. In recent years, their simplicity, versatility and efficiency have made them a standard tool for practitioners, and a fundamental topic in many data analysis courses. We will outline some of their important features in this Chapter, referring the interested reader to more detailed articles and books for a deeper discussion (see for example [135] and references therein).

KMs combine the simplicity and computational efficiency of linear algorithms, such as the perceptron algorithm or ridge regression, with the flexibility of non-linear systems, such as for example neural networks, and the rigour of statistical approaches such as regularization methods in multivariate statistics. As a result of the special way they represent functions, these algorithms typically reduce the learning step to a convex optimization problem, that can always be solved in polynomial time, avoiding the problem of local minima typical of neural networks, decision trees and other non-linear approaches.

Their foundation in the principles of Statistical Learning Theory make them remarkably resistant to overfitting especially in regimes where other methods are affected by the 'curse of dimensionality'. It is for this reason that they have become popular in bioinformatics and text analysis. Another important feature for applications is that they can naturally accept input data that are not in the form of vectors, such as for example strings, trees and images.

Their characteristically modular design makes them amenable to theoretical analysis but also well suited to a software engineering approach: a general purpose learning module is combined with a data specific 'kernel function' that provides the interface with the data and incorporates domain knowledge. Many learning modules can be used depending on whether the task is one of classification, regression, clustering, novelty detection, ranking, etc. At the same time many kernel functions have been designed: for protein sequences, for text and hypertext documents, for images, time series, etc. The result is that this method can be used for dealing with rather exotic tasks, such as ranking strings, or clustering graphs, in addition to such classical tasks as classifying vectors. We will delay the definition of a kernel till the next section even though kernels form the core of this contribution.

In this Chapter, we will introduce the main concepts behind this approach to data analysis by discussing some simple examples. We will start with the simplest algorithm and the simplest kernel function, so as to illustrate the basic concepts. Then we will discuss the issue of overfitting, the role of generalization bounds and how they suggest more effective strategies, leading to the Support Vector Machine (SVM) algorithm. In the conclusion, we will briefly discuss other pattern recognition algorithms that exploit the same ideas, for example Principal Components Analysis (PCA), Canonical Correlation Analysis (CCA), and extensions of the SVM algorithm to regression and novelty detection. In this short chapter we err in favour of giving a detailed description of a few standard methods, rather than a superficial survey of the majority.

The problem we will use as an example throughout the chapter is the one of learning a binary classification function using a real-valued function $f : X \subseteq \mathbb{R}^n \to \mathbb{R}$ in the following way: the input $\mathbf{x} = (x_1, \ldots, x_n)'$ is assigned to the positive class, if $f(\mathbf{x}) \geq 0$, and otherwise to the negative class. We are interested in the case where $f(\mathbf{x})$ is a non-linear function of $\mathbf{x} \in X$, though we will solve the non-linear problem by using linear $f(\mathbf{x})$ in a space that is the image of a non-linear mapping.

We will use $X$ to denote the input space and $Y$ to denote the output domain. Usually we will have $X \subseteq \mathbb{R}^n$, while for binary classification $Y = \{-1, 1\}$ and for regression $Y \subseteq \mathbb{R}$. The *training set* is a collection of *training examples*, which are also called *training data*. It is denoted by

$$S = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)) \subseteq (X \times Y)^m ,$$

where $m$ is the number of examples. We refer to the $\mathbf{x}_i$ as *examples* or *instances* and the $y_i$ as their *labels*. Note that if $X$ is a vector space, the input vectors are column vectors as are the weight vectors. If we wish to form a row vector from $\mathbf{x}_i$ we can take the transpose $\mathbf{x}_i'$. We denote by $\langle \mathbf{x}, \mathbf{w} \rangle = \mathbf{x}'\mathbf{w} = \sum_i \mathbf{x}_i \mathbf{w}_i$ the inner product between the vectors $\mathbf{x}$ and $\mathbf{w}$.

## 5.1. Example: Kernel Perceptron

The main idea of Kernel Methods is to first embed the data into a suitable vector space, and then use simple linear methods to detect relevant patterns in the resulting set of points. If the embedding map is non-linear, this enables us to discover non-linear relations using linear algorithms. Hence, we consider a map