1

Introduction

Overview. *The goal of this introduction is to give a gentle and informal overview of what this book is about. In particular, we will discuss key concepts and questions on statistical learning. Furthermore, the underlying ideas of support vector machines are presented, and important questions for understanding their learning mechanisms will be raised.*

Usage. *This introduction serves as a tutorial that presents key concepts and questions of this book. By connecting these to corresponding parts of the book, it is furthermore a guidepost for reading this book.*

It is by no means a simple task to give a precise definition of learning and, furthermore, the notion of learning is used in many different topics. Rather than attempting to give a definition of learning on our own, we thus only mention two possible versions. Following the Encyclopædia Britannica (online version), *learning is the*

"process of acquiring modifications in existing knowledge, skills, habits, or tendencies through experience, practice, or exercise."

Simon (1983, p. 28), who was awarded the Nobel Prize in Economic Sciences in 1978, defined learning in the following way:

"Learning denotes changes in the system that are adaptive in the sense that they enable the system to do the same task or tasks drawn from the same population more efficiently and more effectively the next time."

1.1 Statistical Learning

Statistical learning is a particular mathematical formulation of the general concept of learning. Before we present this formulation, let us first describe and motivate it in a rather informal way. To this end, let us assume that we want to relate a certain type of *input* value or measurement(s) to an *output* or response. Knowing whether such a dependence structure between input and output values exists, and if so which functional relationship describes it, might be of interest in real-life applications such as the following:

(a) make a diagnosis based on some clinical measurements;

(b) assign the ASCII code to digitalized images of handwritten characters;
(c) predict whether a client will pay back a loan to a bank;
(d) assess the price of a house based on certain characteristics;
(e) estimate the costs of claims of insurees based on insurance data.

Another characteristic of the applications we have in mind is that there is a need to *automatically* assign the response. For example, this may be the case because the structure of the measurements is too complex to be reasonably understood by human experts, or the amount of measurements is too vast to be manually processed in a timely fashion. The examples above illustrate, however, that we often do not have a reasonable description of a functional relationship between the measurements and the desired response that can easily be formalized and implemented on a computer. One way to resolve this problem is the one taken by statistical learning theory or machine learning. In this approach, it is assumed that we have already gathered a finite set of input values together with corresponding output values. For example, in the scenarios above, we could have collected: *(a)* data on some patients who had already developed the disease, *(b)* handwritten characters with manually assigned ASCII code, *(c)* data on clients who got a loan from the bank within the last ten years, *(d)* the prices of recently sold houses, or *(e)* insurance data of previous years. In the machine learning approach, the limited number of input values with known output values are then used to "learn" the assumed but unknown functional relationship between the input values and the output values by an algorithm, which in turn makes it possible to predict the output value for *future* input values. This is a crucial point. In many applications, the collected data set consisting of input values and output values can be thought of as a finite sample taken from all possible input and output values. However, the goal is not to find a suitable description of the dependency between input and output values of the collected data set (because we already know the output values) but to find a prediction rule for output values that works well for new, so far unseen input values.

In order to formalize this approach, we first assume that all input values x are contained in a known set X that describes their format and their range. In the examples above, this could be *(a)* the set of possible values of clinically measured parameters, *(b)* the set of all possible sixteen by sixteen digitalized black and white images, *(c)* the set of all possible information on the clients, *(d)* the set of possible configurations and locations of houses in a town, or *(e)* the set of all possible collected personal information of insurees. In addition, we assume that we have a known set Y that describes the format and the range of possible responses. For example, this could simply be the set $\{$"negative","positive"$\}$ or $\{-1, +1\}$ when we want to diagnose a disease, while in the other examples Y could be the set of ASCII codes related to numerals or letters, a price range, or a range that contains all possible claim amounts.

As already mentioned informally, we assume in the machine learning approach that we have collected a sequence $D := ((x_1, y_1), \ldots, (x_n, y_n))$ of input/output pairs that are used to "learn" a function $f : X \to Y$ such that $f(x)$ is a good approximation of the possible response y to an arbitrary x. Obviously, in order to find such a function, it is necessary that the already collected data D have something in common with the new and unseen data. In the framework of statistical learning theory, this is guaranteed by assuming that both past and future pairs (x, y) are *independently* generated by the same, but of course *unknown*, probability distribution P on $X \times Y$. In other words, a pair (x, y) is generated in two steps. First, the input value x is generated according to the marginal distribution P_X. Second, the output value y is generated according to the conditional probability $\mathrm{P}(\,\cdot\,|x)$ on Y given the value of x. Note that by letting x be generated by an *unknown* distribution P_X, we basically assume that we have no control over how the input values have been and will be observed. Furthermore, assuming that the output value y to a given x is stochastically generated by $\mathrm{P}(\,\cdot\,|x)$ accommodates the fact that in general the information contained in x may not be sufficient to determine a single response in a deterministic manner. In particular, this assumption includes the two extreme cases where either all input values determine an (almost surely) unique output value or the input values are completely irrelevant for the output value. Finally, assuming that the conditional probability $\mathrm{P}(\,\cdot\,|x)$ is *unknown* contributes to the fact that we assume that we do not have a reasonable description of the relationship between the input and output values. Note that this is a fundamental difference from *parametric models*, in which the relationship between the inputs x and the outputs y is assumed to follow some unknown function $f \in \mathcal{F}$ from a *known, finite-dimensional* set of functions $\mathcal{F}$.

So far, we have only described the nature of the data with which we are dealing. Our next goal is to describe what we actually mean by "learning." To this end, we assume that we have means to assess the quality of an estimated response $f(x)$ when the true input and output pair is (x, y). To simplify things, we assume throughout this book that the set of possible responses Y is a subset of $\mathbb{R}$ and that all estimated responses are real-valued. Moreover, we assume that our quality measure is a non-negative real number $L(x, y, f(x))$ that is smaller when the estimated response $f(x)$ is better. In other words, we have a function $L : X \times Y \times \mathbb{R} \to [0, \infty)$, which in the following is called a *loss function*, where the term "loss" indicates that we are interested in small values of L. Of course, in order to assess the quality of a learned function f, it does not suffice to know the value $L(x, y, f(x))$ for a particular choice of (x, y), but in fact we need to quantify how small the *function* $(x, y) \mapsto L(x, y, f(x))$ is. Clearly, there are many different ways to do this, but in statistical learning theory one usually considers the *expected* loss of f, that is, the quantity

$$\mathcal{R}_{L,\mathrm{P}}(f) := \int_{X \times Y} L\big(x, y, f(x)\big)\, d\mathrm{P}(x, y) = \int_X \int_Y L\big(x, y, f(x)\big)\, d\mathrm{P}(y|x) d\mathrm{P}_X\,,$$

which in the following is called the *risk* of f.[1] Let us consider a few examples to motivate this choice.

For the first example, we recall the scenario *(b)*, where the goal was to assign ASCII codes to digitalized images of handwritten characters. A straightforward loss function for this problem is the function $L(x, y, f(x))$, which equals one whenever the prediction $f(x)$ does not equal the true ASCII code y and is otherwise equal to zero. Now assume that we have already learned a function f and our goal is now to apply f to new bitmaps $x_{n+1}, \dots, x_m$ that correspond to the ASCII codes $y_{n+1}, \dots, y_m$. Then intuitively a function is better the fewer errors it makes on $x_{n+1}, \dots, x_m$, and obviously this is equivalent to saying that the average future empirical loss

$$\frac{1}{m-n} \sum_{i=n+1}^{m} L\bigl(x_i, y_i, f(x_i)\bigr) \tag{1.1}$$

should be as small as possible. Now recall that we always assume that $(x_{n+1}, y_{n+1}), \dots, (x_m, y_m)$ are independently generated by the same distribution P, and consequently the law of large numbers (see Theorem A.4.8) shows that, for $m \to \infty$, the average empirical loss in (1.1) converges in probability to $\mathcal{R}_{L,\mathrm{P}}(f)$. In other words, the risk is indeed a very reasonable quality measure for the function f.

In the example above the risk directly assesses how well a function f performs a certain task, namely classification. In contrast to this, our second example considers a case where the main learning objective is to estimate a function f^* and the risk used is only a relatively arbitrary tool to describe this objective. To be more concrete, let us recall the scenario *(d)*, where we wished to assess the prices of houses. According to our general assumption, these prices have the unknown distributions $\mathrm{P}(\,\cdot\,|x)$, where x may describe the configuration and location of a certain house. Now, depending on the particular application, it could be reasonable to estimate the center of the conditional probability distribution such as the conditional mean or the conditional median of Y given x. Let us write $f^*(x)$ for either of them. Intuitively, a good estimator $f(x)$ of the quantity $f^*(x)$ should be close to it, and hence we could consider loss functions of the form

$$L_p(x, y, f(x)) := |f^*(x) - f(x)|^p,$$

where $p > 0$ is some fixed number.[2] The average loss (i.e., the risk) of an estimator f then becomes

[1] Throughout the introduction, we ignore technicalities such as measurability, integrability, etc., to simplify the presentation. In the subsequent chapters, however, we will seriously address these issues.

[2] The experienced reader probably noticed that for these learning goals one often uses the least squares loss or the absolute distance loss, respectively. However, these loss functions are strictly speaking only *tools* to determine f^* and do not define the goal itself. While later in the book we will also use these loss functions

$$\mathcal{R}_{L_p,\mathrm{P}}(f) = \int_X \bigl|f^*(x) - f(x)\bigr|^p \, d\mathrm{P}_X(x) \,,$$

which obviously is a reasonable quality measure with a clear intuitive meaning. Note, however, that, unlike in the first example, we are not able to actually compute the loss functions L_p since we do not know f^*. Moreover, there seems to be no natural choice for p either, though at least for the problem of estimating the conditional mean we will see in Example 2.6 that $p = 2$ is in some sense a canonical choice. Similarly, we will see in Example 3.67 that, under some relatively mild assumptions on the conditional distributions $\mathrm{P}(\,\cdot\,|x)$, the choice $p = 1$ is suitable for the problem of estimating the conditional median.

Let us now return to the general description of the learning problem. To this end, recall that a function is considered to be better the smaller its risk $\mathcal{R}_{L,\mathrm{P}}(f)$ is. Hence it is natural to consider the smallest possible risk,

$$\mathcal{R}^*_{L,\mathrm{P}} := \inf_{f:X\to\mathbb{R}} \mathcal{R}_{L,\mathrm{P}}(f) \,,$$

where the infimum is taken over the set of *all* possible functions. Note that considering all possible functions is in general necessary since we do not make assumptions on the distribution P. Nevertheless, for particular loss functions, we can actually consider smaller sets of functions without changing the quantity $\mathcal{R}^*_{L,\mathrm{P}}$. For example, in the scenario where we wish to assign ASCII codes to digitalized images, it clearly makes no difference whether we consider all functions or just all functions that take values in the ASCII codes of interest. Moreover, we will see in Section 5.5 that in many cases it suffices to consider sets of functions that are in some sense dense.

So far, we have described that we wish to find a function $f : X \to \mathbb{R}$ that (approximately) minimizes the risk $\mathcal{R}_{L,\mathrm{P}}$. If the distribution P is known, this is often a relatively easy task, as we will see in Section 3.1.[3] In our setup, however, the distribution P is unknown, and hence it is in general impossible to find such an (approximate) minimizer without additional information. In the framework of statistical learning theory, this information comes in the form of the already collected finite data set $D := ((x_1, y_1), \ldots, (x_n, y_n))$, where all n data points (x_i, y_i) are assumed to be generated independently from the same distribution P. Based on this data set, we then want to build a function $f_D : X \to \mathbb{R}$ whose risk $\mathcal{R}_{L,\mathrm{P}}(f_D)$ is close to the minimal risk $\mathcal{R}^*_{L,\mathrm{P}}$. Since the process of building such a function should be done in a systematic manner, we restrict our considerations throughout this book to *learning methods*, that

as tools, it is conceptionally important to distinguish between the *learning goal* and the *tools to achieve this goal*. A systematic treatment of this difference will be given in Chapter 3.

[3] In this case there is, of course, no need to learn since we already know the distribution P that describes the desired functional relationship between input and output values. Nonetheless, we will see that we gain substantial insight into the learning process by considering the case of known P.

is, to deterministic methods that assign to *every* finite sequence D a *unique* function f_D. Now one way to formalize what is meant by saying that a learning method is able to learn is the notion of *universal consistency.* This notion which we will discuss in detail in Section 6.1, requires that, for *all* distributions P on $X \times Y$, the functions f_D produced by the learning method satisfy

$$\mathcal{R}_{L,\mathrm{P}}(f_D) \to \mathcal{R}^*_{L,\mathrm{P}}\,, \qquad n \to \infty, \tag{1.2}$$

in probability. In other words, we wish to have a learning method that in the long run finds functions with near optimal response performance *without* knowing any specifics of P. Although this generality with respect to P may seem to be a very strong goal, it has been known since a seminal paper by Stone (1977) that for certain loss functions there do exist such learning methods. Moreover, in Section 6.4 and Chapter 9, we will see that the learning methods we will deal with, namely *support vector machines* (SVMs), are often universally consistent.

One clear disadvantage of the notion of universal consistency is that no speed of convergence is quantified in (1.2). In particular, we cannot *a priori* exclude the possibility that universally consistent methods are only able to find functions f_D with near optimal response performances for extremely large values of n. Unfortunately, it turns out by the so-called *no-free-lunch theorem* shown by Devroye (1982) that in general this issue cannot be resolved. To be more precise, we will see in Section 6.1 that for every learning method and every *a priori* fixed speed of convergence there exists a distribution P for which the learning method cannot achieve (1.2) with the prescribed speed. Having said that, it is, however, well-known that for many learning methods it is possible to derive uniform convergence rates, and sometimes also asymptotic distributions, under certain *additional* assumptions on P. For the earlier mentioned and more restrictive case of parametric models and for local asymptotic optimality results, we refer to Lehmann and Casella (1998) and LeCam (1986), respectively. Moreover, if P is an element of a neighborhood of a parametric model and a robust statistical method is used, we refer to Huber (1964, 1967, 1981), Hampel *et al.* (1986), and Rieder (1994). On the other hand, convergence rates for regression with smooth but otherwise unspecified target functions are discussed in great detail by Györfi *et al.* (2002). While one can show that convergence rates for regression can also be obtained by certain SVMs, we will mainly focus on convergence rates for classification. In particular, we will present a class of mild assumptions on P in Chapter 8 that, while realistic in many cases, still allow us to derive reasonable learning rates. Finally, one should always keep in mind that the existence of convergence rates only provides theoretical assurance up to a certain degree since in practice we can almost never rigorously prove that the required assumptions are met.

1.2 Support Vector Machines: An Overview

Let us now describe the basic ideas of support vector machines. In order to fix ideas, we focus here in the introduction on only one particular type of learning problem, namely binary classification. For convenience, let us in this and the following section assume that the loss function depends on x only via $f(x)$ such that we can simply write $L(y, f(x))$ instead of $L(x, y, f(x))$. As in example *(a)* mentioned earlier, where the goal was to make a diagnosis, the goal in binary classification is to estimate a response that only has two states. Consequently, we define the set of possible response values by $Y := \{-1, +1\}$. Moreover, the *classification loss function* L_{class} commonly used in binary classification only penalizes misclassifications (i.e., $L_{\mathrm{class}}(y, f(x))$ equals 1 if $\operatorname{sign} f(x) \neq y$ and equals 0 otherwise). Finally, we assume that all possible input values are contained in some set, say $X \subset \mathbb{R}^d$.

Let us now recall that the *learning goal* was to find a function f^* that (approximately) achieves the smallest possible risk,

$$\mathcal{R}^*_{L,\mathrm{P}} = \inf_{f:X\to\mathbb{R}} \mathcal{R}_{L,\mathrm{P}}(f)\,, \tag{1.3}$$

where $L := L_{\mathrm{class}}$. Since the distribution P generating the input/output pairs is unknown, the risk $\mathcal{R}_{L,\mathrm{P}}(f)$ is unknown and consequently we cannot directly find f^*. To resolve this problem, it is tempting to replace the risk $\mathcal{R}_{L,\mathrm{P}}(f)$ in (1.3) by its empirical counterpart

$$\mathcal{R}_{L,\mathrm{D}}(f) := \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i))\,,$$

where $D := ((x_1, y_1), \dots, (x_n, y_n))$ is the finite sequence of already gathered samples.[4] Unfortunately, however, even though the law of large numbers shows that $\mathcal{R}_{L,\mathrm{D}}(f)$ is an approximation of $\mathcal{R}_{L,\mathrm{P}}(f)$ for each *single* f, solving

$$\inf_{f:X\to\mathbb{R}} \mathcal{R}_{L,\mathrm{D}}(f) \tag{1.4}$$

does not in general lead to an approximate minimizer of $\mathcal{R}_{L,\mathrm{P}}(\,\cdot\,)$. To see this, consider the function that classifies all x_i in D correctly but equals 0 everywhere else. Then this function is clearly a solution of (1.4), but since this function only memorizes D, it is in general a very poor approximation of (1.3). This example is an extreme form of a phenomenon called *overfitting*, in which the learning method produces a function that models too closely the output values in D and, as a result, has a poor performance on future data.

One common way to avoid overfitting is to choose a small set $\mathcal{F}$ of functions $f : X \to \mathbb{R}$ that is assumed to contain a reasonably good approximation of the solution of (1.3). Then, instead of minimizing $\mathcal{R}_{L,\mathrm{D}}(\,\cdot\,)$ over all functions, one minimizes only over $\mathcal{F}$; i.e., one solves

[4] The corresponding empirical distribution is denoted by $\mathrm{D} := \frac{1}{n}\sum_{i=1}^{n} \delta_{(x_i,y_i)}$.

$$\inf_{f\in\mathcal{F}} \mathcal{R}_{L,\mathrm{D}}(f)\,. \tag{1.5}$$

This approach, which is called *empirical risk minimization* (ERM), often tends to produce approximate solutions of the infinite-sample counterpart of (1.5),

$$\mathcal{R}^*_{L,\mathrm{P},\mathcal{F}} := \inf_{f\in\mathcal{F}} \mathcal{R}_{L,\mathrm{P}}(f)\,. \tag{1.6}$$

In particular, we will see in Section 6.3 that this is true if $\mathcal{F}$ is finite or if $\mathcal{F}$ can at least be approximated by a finite set of functions. Unfortunately, however, this approach has two serious issues. The first one is that in the problems we are interested in our knowledge of P is in general not rich enough to identify a set $\mathcal{F}$ such that a solution of (1.6) is a reasonably good approximation of the solution of (1.3). In other words, we usually cannot guarantee that the *model error* or *approximation error*

$$\mathcal{R}^*_{L,\mathrm{P},\mathcal{F}} - \mathcal{R}^*_{L,\mathrm{P}} \tag{1.7}$$

is sufficiently small. The second issue is that solving (1.5) may be computationally infeasible. For example, the 0/1 loss function used in binary classification is non-convex, and as a consequence solving (1.5) is often NP-hard, as Höffgen *et al.* (1995) showed.

To resolve the first issue, one usually increases the size of the set $\mathcal{F}$ with the sample size n so that the approximation error (1.7) decreases with the sample size. In this approach, it is crucial to ensure that solving (1.5) for larger sets $\mathcal{F}$ still leads to approximate solutions of (1.6), which is usually achieved by controlling the growth of $\mathcal{F}$ with the sample size. The second issue is often resolved by replacing the risk $\mathcal{R}_{L,\mathrm{D}}(\,\cdot\,)$ in (1.5) by a suitable surrogate that is computationally more attractive. Various proposed learning methods follow these two basic ideas in one form or another, and a complete account of these methods would fill another textbook. Consequently, we will focus in the following on how support vector machines implement these two basic strategies.

Let us first explain the idea of how SVMs make the optimization problem computationally feasible. Here the first step is to replace the 0/1 classification loss by a *convex* surrogate. The most common choice in this regard is the hinge loss, which is defined by

$$L_{\mathrm{hinge}}(y,t) := \max\{0, 1-yt\}, \qquad y\in\{-1,+1\},\ t\in\mathbb{R};$$

see also Figure 3.1. It is easy to see that the corresponding empirical risk $\mathcal{R}_{L_{\mathrm{hinge}},\mathrm{D}}(f)$ is convex in f, and consequently, if we further assume that $\mathcal{F}$ is a convex set, we end up with the convex optimization problem

$$\inf_{f\in\mathcal{F}} \mathcal{R}_{L_{\mathrm{hinge}},\mathrm{D}}(f)\,, \tag{1.8}$$

which defines our *learning method*. Before we present another step SVMs take to make the optimization problem more attractive, let us briefly note that using the surrogate loss function L_{hinge} instead of the non-convex classification loss raises a new issue. To explain this, let us assume that (1.8) is well-behaved in the sense that solving (1.8) leads to a function f_D whose risk $\mathcal{R}_{L_{\text{hinge}},\mathrm{P}}(f_D)$ is close to

$$\mathcal{R}^*_{L_{\text{hinge}},\mathrm{P},\mathcal{F}} := \inf_{f\in\mathcal{F}} \mathcal{R}_{L_{\text{hinge}},\mathrm{P}}(f)\,.$$

Moreover, we assume that the *approximation error* $\mathcal{R}^*_{L_{\text{hinge}},\mathrm{P},\mathcal{F}} - \mathcal{R}^*_{L_{\text{hinge}},\mathrm{P}}$ with respect to the hinge loss is small. In other words, we assume that we are in a situation where we can hope to find a function $f_D \in \mathcal{F}$ such that

$$\mathcal{R}_{L_{\text{hinge}},\mathrm{P}}(f_D) - \mathcal{R}^*_{L_{\text{hinge}},\mathrm{P}}$$

is small. However, we are actually interested in learning with respect to the classification risk; i.e., we want $\mathcal{R}_{L_{\text{class}},\mathrm{P}}(f_D) - \mathcal{R}^*_{L_{\text{class}},\mathrm{P}}$ to be small. Obviously, one way to ensure this is to establish inequalities between the two differences. Fortunately, for the hinge and classification losses, it will turn out in Section 2.3 that relatively elementary considerations yield

$$\mathcal{R}_{L_{\text{class}},\mathrm{P}}(f) - \mathcal{R}^*_{L_{\text{class}},\mathrm{P}} \;\le\; \mathcal{R}_{L_{\text{hinge}},\mathrm{P}}(f) - \mathcal{R}^*_{L_{\text{hinge}},\mathrm{P}}$$

for all functions $f : X \to \mathbb{R}$. Therefore, the idea of using the hinge loss is justified as long as we can guarantee that our learning method learns well in terms of the hinge loss. Unfortunately, however, for several other learning problems, such as estimating the conditional median, it turns out to be substantially more difficult to establish inequalities that relate the risk used in the *learning method* to the risk defining the *learning goal*. Since this is a rather general issue, we will develop in Chapter 3 a set of tools that make it possible to derive such inequalities in a systematic way.

Let us now return to support vector machines. So far, we have seen that for binary classification they replace the non-convex 0/1-classification loss by a convex surrogate loss such as the hinge loss. Now, the second step of SVMs toward computational feasibility is to consider very specific sets of functions, namely *reproducing kernel Hilbert spaces*[5] H. These spaces will be introduced and investigated in detail in Chapter 4, and hence we skip a formal definition here in the introduction and only mention that for now we may simply think of them as Hilbert spaces that consist of functions $f : X \to \mathbb{R}$. We will see in Chapter 4 that every RKHS possesses a unique function $k : X \times X \to \mathbb{R}$, called its *kernel*, that can be used to describe all functions contained in H. Moreover, the value $k(x, x')$ can often be interpreted as a measure of dissimilarity between the input values x and x'. Let us fix such an RKHS H and denote its norm by $\|\cdot\|_H$. For a fixed real number $\lambda > 0$, support vector machines then find a minimizer of

$$\inf_{f\in H} \lambda\|f\|_H^2 + \mathcal{R}_{L,\mathrm{D}}(f)\,, \tag{1.9}$$

[5] We will often use the abbreviation RKHS for such Hilbert spaces.

where the *regularization term* $\lambda\|f\|_H^2$ is used to penalize functions f with a large RKHS norm. One motivation of this regularization term is to reduce the danger of overfitting; rather complex functions $f \in H$, which model too closely the output values in the training data set D, tend to have large H-norms. The regularization term penalizes such functions more than "simple" functions.

If L is a convex loss function such as the hinge loss, the objective function in (1.9) becomes convex in f. Using this, we will see in Section 5.1 and Chapter 11 that (1.9) has in basically all situations of interest a *unique* and *exact* minimizer, which in the following we denote by $f_{\mathrm{D},\lambda}$. Moreover, we will see in Section 5.1 that this minimizer is of the form

$$f_{\mathrm{D},\lambda} = \sum_{i=1}^{n} \alpha_i k(x_i, \cdot\,)\,, \tag{1.10}$$

where $k : X \times X \to \mathbb{R}$ is the kernel that belongs to H and $\alpha_1, \dots, \alpha_n \in \mathbb{R}$ are suitable coefficients. In other words, the minimizer $f_{\mathrm{D},\lambda}$ is a weighted average of (at most) n functions $k(x_i, \cdot\,)$, where the weights α_i are data-dependent. A remarkable consequence of the representation given in (1.10) is the fact that $f_{\mathrm{D},\lambda}$ is contained in a known *finite* dimensional space, namely the linear span of $k(x_i, \cdot\,)$, $1 \le i \le n$, even if the space H itself is substantially larger. This observation makes it possible to consider even *infinite* dimensional spaces H such as the one belonging to the popular *Gaussian radial basis function* (RBF) *kernel* (see Section 4.4 for a detailed account) defined by

$$k_\gamma(x, x') := \exp\left(-\gamma^{-2}\|x - x'\|_2^2\right)\,, \quad x, x' \in \mathbb{R}^d,$$

where $\gamma > 0$ is a fixed parameter called the *width*. Moreover, for particular loss functions such as the hinge loss, we will see below, and in much more detail in Chapter 11, that the solution $f_{\mathrm{D},\lambda}$ of (1.9) can be found by solving a *convex quadratic* optimization problem with linear constraints. For the hinge loss, we will further develop and analyze efficient algorithms to compute $f_{\mathrm{D},\lambda}$ in Section 11.2.

Although computational feasibility is an important feature of every learning algorithm, one of the most important features is definitely its ability to find decision functions having near optimal risks. Let us now motivate why SVMs can find such functions in many situations. To this end, we again restrict our considerations here in the introduction to SVMs using the hinge loss L. For this loss function, we easily check that $\mathcal{R}_{L,\mathrm{D}}(0) = 1$ for every sample set D and hence obtain

$$\lambda\|f_{\mathrm{D},\lambda}\|_H^2 \;\le\; \lambda\|f_{\mathrm{D},\lambda}\|_H^2 + \mathcal{R}_{L,\mathrm{D}}(f_{\mathrm{D},\lambda}) \;\le\; \mathcal{R}_{L,\mathrm{D}}(0) = 1\,,$$

where in the second estimate we used that, by definition, $f_{\mathrm{D},\lambda}$ is a solution of (1.9). Consequently, $f_{\mathrm{D},\lambda}$ is also a solution of the optimization problem

$$\min_{\|f\|_H \le \lambda^{-1/2}} \lambda\|f\|_H^2 + \mathcal{R}_{L,\mathrm{D}}(f)\,.$$

Now smaller values of λ lead to larger sets $\{f \in H : \|f\|_H \leq \lambda^{-1/2}\}$, the minimum is computed over, and in addition smaller values of λ also reduce the influence of the regularization term $\lambda\|\cdot\|_H^2$. Consequently, the SVM optimization problem can be interpreted as an approximation of the ERM approach with increasing sets of functions. So far this interpretation is, however, little more than a vague analogy, but we will see in Section 6.4 that the techniques used to analyze ERM can actually be extended to a basic statistical analysis of SVMs. In particular, we will see there that with probability not smaller than $1 - e^{-\tau}$ we have

$$\lambda\|f_{\mathrm{D},\lambda}\|_H^2 + \mathcal{R}_{L,\mathrm{P}}(f_{\mathrm{D},\lambda}) \;\leq\; \inf_{f\in H} \lambda\|f\|_H^2 + \mathcal{R}_{L,\mathrm{P}}(f) + \varepsilon(n,\lambda,\tau)\,, \tag{1.11}$$

where $\tau > 0$ is arbitrary and the value $\varepsilon(n,\lambda,\tau)$, which we will derive explicitly, converges to 0 for $n \to \infty$. Consequently, we see that besides this statistical analysis we also need to understand how the right-hand side of (1.11) behaves as a function of λ. This will be one of the major topics of Chapter 5, where we will in particular show that

$$\lim_{\lambda\to 0}\bigl(\inf_{f\in H} \lambda\|f\|_H^2 + \mathcal{R}_{L,\mathrm{P}}(f)\bigr) = \inf_{f\in H} \mathcal{R}_{L,\mathrm{P}}(f)\,. \tag{1.12}$$

Starting from this observation, we will further investigate in Section 5.5 under which conditions H can be used to approximate the minimal risk in the sense of $\inf_{f\in H} \mathcal{R}_{L,\mathrm{P}}(f) = \mathcal{R}_{L,\mathrm{P}}^*$. In particular, we will see with some results from Section 4.4 that in almost all situations the RKHSs of the Gaussian RBF kernels satisfy this equality. Combining this with (1.11), we will then show in Section 6.4 that SVMs using such a kernel can be made universally consistent by using suitable null sequences (λ_n) of regularization parameters that depend only on the sample size n but *not* on the distribution P. In addition, similar consistency results for more general cases with *unbounded* Y are given in Section 9.2 for regression and Section 9.3 for quantile regression. Interestingly, the analysis of Section 6.4 further shows that we obtain learning rates for such sequences (λ_n) whenever we know the speed of convergence in (1.12). Unfortunately, however, the best possible rates this analysis yields can only be achieved by sequences (λ_n) that use knowledge about the speed of convergence in (1.12).[6] Since the latter is unknown in almost all applications, we thus need strategies that *adaptively* (i.e., without knowledge of P) find nearly optimal values for λ. In Section 6.5, we will analyze a very simple version of such a strategy that despite its simplicity resembles many ideas of commonly used, more complex strategies. In practice, not only the quantity λ but also some other so-called *hyperparameters* have an impact on the quality of $f_{\mathrm{D},\lambda}$. For example, if we use a Gaussian RBF kernel, we have to specify the value of the width γ. We will deal with this issue in Section 8.2 from a theoretical point of view and in Section 11.3 from a practical point of view.

[6] This phenomenon remains true for the more advanced analysis in Chapter 7.

The discussion above showed that, in order to understand when SVMs learn with a favorable learning rate, one needs to know when the RKHS H approximates the risk easily in the sense of the convergence given in (1.12). By the no-free-lunch theorem mentioned earlier and (1.11), the latter requires assumptions on the distribution P and the space H. In Section 8.2, we will present such an assumption for binary classification that *(a)* is weak enough to be likely met in practice and *(b)* still allows a reasonable mathematical treatment. Roughly speaking (see Figures 8.1 and 8.2 for some illustrations), this type of assumption describes how the data are typically concentrated in the vicinity of the decision boundary.

The ability to learn from a finite number of data points with a reasonable algorithmic complexity is in general not enough for a learning method to be successful from both a theoretical and an applied point of view. Indeed, already Hadamard (1902) thought that a well-posed mathematical problem should have the property that there *exists a unique solution that additionally depends continuously on the data*. In our case, this means that a few outlying data points that are far away from the pattern set by the majority of the data should influence $f_{\mathrm{D},\lambda}$ and its associated risk $\mathcal{R}_{L,\mathrm{D}}(f_{\mathrm{D},\lambda})$ only in a continuous and bounded manner. More generally, a good learning method should give stable results for (almost all) distributions Q lying in a small neighborhood of the unknown distribution P. Therefore, we will describe in Chapter 10 some modern concepts of robust statistics such as the influence function, sensitivity curve, and maxbias. By applying these general concepts to SVMs, it will be shown that SVMs have—besides other good properties—the advantage of being *robust* if the loss function L and the kernel k are suitably chosen. In particular, weak conditions on L and k are derived that guarantee good robustness of SVM methods for large classes of probability distributions. These results will be derived not only for the classification case but also for quantile regression and regression for the mean, where for the latter two cases we assume an unbounded range Y of possible outputs. For the latter, we will need an explicit control over the growth of the loss function, and in Section 2.2 we therefore introduce a general notion, namely so-called *Nemitski loss functions*, that describes this growth. Although this notion was originally tailored to regression with unbounded Y, it will turn out that it is also a very fruitful concept for many other learning problems. Finally, we would like to point out that combining the robustness results from Chapter 10 with those from Chapter 9 on SVMs for regression shows that learning properties and robustness properties of SVMs are connected to each other. Roughly speaking, it will turn out that the SVMs with better robustness properties are able to learn over larger classes of distributions than those SVMs with worse robustness properties. Chapter 10 therefore complements recent *stability* results obtained by Poggio *et al.* (2004) and Mukherjee *et al.* (2006), who study the impact of one data point on SVMs under the boundedness assumption of the space of input and output values.

Besides robustness, another important property of learning methods is their ability to find decision functions that can be evaluated in a computationally efficient manner. For support vector machines, the time needed to evaluate $f_{D,\lambda}(x)$ is obviously proportional to the number of nonzero coefficients α_i in the representation (1.10). For SVMs used for binary classification, we will thus investigate the typical number of such coefficients in Sections 8.4 and 8.5. Here and also in Chapter 11, on computational aspects of SVMs, it will turn out that yet another time the choice of the loss function in (1.9) plays a crucial role.

Finally, it is important to mention that support vector machines are often used as *one* tool in data mining projects. Therefore, we will briefly describe in Chapter 12 a general and typical data mining strategy. In particular, we will show how SVMs can be a successful part of a data mining project and mention a few alternative statistical modeling tools often used for data mining purposes, such as generalized linear models and trees. Last but not least, a brief comparison of the advantages and disadvantages of such tools with respect to SVMs is given.

1.3 History of SVMs and Geometrical Interpretation

Considering regularized empirical (least squares) risks over reproducing kernel Hilbert spaces is a relatively old idea (see, e.g., Poggio and Girosi, 1990; Wahba, 1990; and the references therein). Although this view on support vector machines will be adopted throughout the rest of this book, it is nonetheless interesting to take a sidestep and have a look of how a geometric idea led to the first algorithms named "support vector machines." To this end, we again consider a binary classification problem with $Y = \{-1, +1\}$. For this learning problem, the original SVM approach by Boser *et al.* (1992) was derived from the *generalized portrait algorithm* invented earlier by Vapnik and Lerner (1963). Therefore, we begin by describing the latter algorithm. To this end, let us assume that our input space X is a subset of the Euclidean space $\mathbb{R}^d$. Moreover, we assume that we have a training set $D = ((x_1, y_1), \ldots, (x_n, y_n))$ for which there exists an element $w \in \mathbb{R}^d$ with $\|w\|_2 = 1$ and a real number $b \in \mathbb{R}$ such that

$$\begin{aligned} \langle w, x_i \rangle + b &> 0, \qquad && \text{for all } i \text{ with } y_i = +1, \\ \langle w, x_i \rangle + b &< 0, \qquad && \text{for all } i \text{ with } y_i = -1. \end{aligned}$$

In other words, the affine linear hyperplane described by (w, b) perfectly separates the training set D into the two groups $\{(x_i, y_i) \in D : y_i = +1\}$ and $\{(x_i, y_i) \in D : y_i = -1\}$. Now, the generalized portrait algorithm constructs a perfectly separating hyperplane, described by (w_D, b_D) with $\|w_D\|_2 = 1$, that has maximal *margin* (i.e., maximal distance to the points in D). Its resulting decision function is then defined by

$$f_D(x) := \operatorname{sign}\bigl(\langle w_D, x\rangle + b_D\bigr) \tag{1.13}$$

for all $x \in \mathbb{R}^d$. In other words, the decision function f_D assigns negative labels to one affine half-space defined by the hyperplane (w_D, b_D) and positive labels to the other affine half-space. Now note that these half-spaces do not change if we consider $(\kappa w_D, \kappa b_D)$ for some $\kappa > 0$. Instead of looking for an element $w_D \in \mathbb{R}^d$ with $\|w_D\|_2 = 1$ that maximizes the margin, we can thus also fix a lower bound on the margin and look for a vector $w^* \in \mathbb{R}^d$ that respects this lower bound and has minimal norm. In other words, we can seek a solution $(w_D^*, b_D^*) \in \mathbb{R}^d \times \mathbb{R}$ of the optimization problem

$$\begin{array}{lll} \text{minimize} & \langle w, w\rangle & \text{over } w \in \mathbb{R}^d,\ b \in \mathbb{R} \\ \text{subject to} & y_i(\langle w, x_i\rangle + b) \geq 1 & i = 1, \dots, n\,. \end{array} \tag{1.14}$$

Simple linear algebra shows that $w_D = w_D^*/\|w_D^*\|_2$ and $b_D = b_D^*/\|w_D^*\|_2$, and hence solving the optimization problem (1.14) indeed yields the affine hyperplane constructed by the generalized portrait algorithm.

Although geometrically compelling, the ansatz of the generalized portrait algorithm obviously has two shortcomings:

i) A *linear* form of the decision function may not be suitable for the classification task at hand. In particular, we may be confronted with situations in which the training set D cannot be linearly separated at all and hence (w_D, b_D) does not exist.

ii) In the presence of noise, it can happen that we need to misclassify some training points in order to avoid overfitting. In particular, if the dimension d is greater than or equal to the sample size n, overfitting can be a serious issue.

To resolve the first issue, the SVM initially proposed by Boser *et al.* (1992) maps the input data $(x_1, \dots, x_n)$ into a (possibly infinite-dimensional) Hilbert space H_0, the so-called *feature space*, by a typically non-linear map $\Phi : X \to H_0$ called the *feature map*. Then the generalized portrait algorithm is applied to the mapped data set $((\Phi(x_1), y_1), \dots, (\Phi(x_n), y_n))$; i.e., it is applied in H_0 instead of in X. In other words, we replace x and x_i in (1.13) and (1.14) by $\Phi(x)$ and $\Phi(x_i)$, respectively, and the vector w in (1.14) is chosen from the Hilbert space H_0. The corresponding learning method was initially called *maximal margin classifier*, and later also *hard margin SVM*.

We will see in Section 4.6 that for certain feature maps Φ the first issue of the generalized portrait algorithm is successfully addressed. In particular, we will show that there exist feature maps for which *every* training set without contradicting examples (i.e., without samples (x_i, y_i) and (x_j, y_j) satisfying $x_i = x_j$ and $y_i \neq y_j$) can be perfectly separated by a hyperplane in the feature space. The price for this high flexibility however is, that the separating hyperplane now lies in a high or even infinite-dimensional space, and hence the second issue of generating overfitted decision functions becomes even more serious.

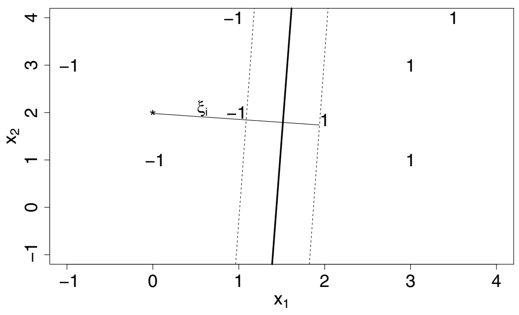

Fig. 1.1. Geometric interpretation of the soft margin SVM in a two-dimensional feature space.

This second issue was first addressed by the *soft margin support vector machine* of Cortes and Vapnik (1995). To explain their approach, recall that in the optimization problem (1.14) the constraints $y_i(\langle w, x_i\rangle + b) \geq 1$ forced the hyperplanes to make no errors on the training data set D. The approach of the soft margin SVM is thus to relax these constraints by requiring only that (w, b) satisfy $y_i\bigl(\langle w, x_i\rangle + b\bigr) \geq 1 - \xi_i$ for some so-called *slack variables* $\xi_i \geq 0$. However, if these slack variables are too large, the relaxed constraints would be trivially satisfied, and hence one has to add safeguards against such behavior. One way to do so is to add the slack variables to the objective function in (1.14).[7] Combining these modifications with the feature map idea leads to the quadratic optimization problem

$$
\begin{array}{lll}
\text{minimize} & \frac{1}{2}\langle w, w\rangle + C\sum\limits_{i=1}^{n}\xi_i & \text{for } w \in H_0,\ b \in \mathbb{R},\ \xi \in \mathbb{R}^n \\
\text{subject to} & y_i(\langle w, \Phi(x_i)\rangle + b) \geq 1 - \xi_i, & i = 1, \dots, n \\
 & \xi_i \geq 0, & i = 1, \dots, n,
\end{array}
\tag{1.15}
$$

where $C > 0$ is a free (but fixed) parameter that is used to balance the first term of the objective function with the second. Note that, due to the special form of the supplemented term $C\sum_{i=1}^n \xi_i$, the objective function is still convex, or to be more precise, quadratic, while the constraints are all linear.

Although this optimization problem looks at first glance more complicated than that of the generalized portrait algorithm, it still enjoys a nice geometrical interpretation for *linear kernels* $k(x, x') := \langle x, x'\rangle$ where $x, x' \in \mathbb{R}^d$ (see Lemma 4.7). Let us illustrate this in the case where $X = H_0 = \mathbb{R}^2$ and

[7] Their original motivation for this step was a little more involved, but at this point we decided to slightly sacrifice historical accuracy for the sake of clarity.

$\Phi : X \to H_0$ is the identity; see Figure 1.1. To this end, we fix a vector $w = (w_1, w_2) \in \mathbb{R}^2$, where without loss of generality we assume $w_1 < 0$ and $w_2 > 0$. Moreover, we fix a sample (x, y) from D whose index denoting the sample number we omit here for notational reasons. Instead we denote the coordinates of x by indexes; i.e., we write $x = (x_1, x_2)$. Let us first consider the case $y = 1$. Since $\langle w, \Phi(x)\rangle = w_1x_1 + w_2x_2$ it is then easy to see that the linear constraint

$$y(\langle w, \Phi(x)\rangle + b) \geq 1 - \xi \tag{1.16}$$

requires a strictly positive slack variable (i.e., $\xi > 0$) if and only if

$$x_2 < \frac{1-b}{w_2} - \frac{w_1}{w_2} x_1 . \tag{1.17}$$

In an analogous manner, we obtain in the case $y = -1$ that (1.16) requires a strictly positive slack variable if and only if

$$x_2 > \frac{-1-b}{w_2} - \frac{w_1}{w_2} x_1 .$$

A comparison of these inequalities shows that both lines have equal slopes but different intercept terms. The latter terms define a tube of width $2/w_2$ in the x_2-direction around the affine hyperplane given by (w, b), which in our simple case is described by

$$x_2 = -\frac{b}{w_2} - \frac{w_1}{w_2} x_1 .$$

Let us compare the decisions made by this hyperplane with the behavior of the slack variables. To this end, we again restrict our considerations to the case $y = 1$. Moreover, we assume that $x = (x_1, x_2)$ is correctly classified, i.e.,

$$x_2 > -\frac{b}{w_2} - \frac{w_1}{w_2} x_1 ,$$

and that x is contained inside the tube around the separating hyperplane. Then (1.17) is satisfied and hence we have a strictly positive slack variable that is penalized in the objective function of (1.15). In other words, the objective function in (1.15) penalizes margin errors (i.e., points inside the tube or lying in the wrong affine hyperplane) and not only classification errors (i.e., points lying in the wrong affine half-space).

Let us now relate the optimization problem (1.15) to the previous SVM formulation given by the optimization problem (1.9). To this end, we observe that the first set of linear constraints can be rewritten as $\xi_i \geq 1 - y_i(\langle w, \Phi(x_i)\rangle + b)$. Combining this constraint with the second set of constraints, namely $\xi_i \geq 0$, we see that the slack variables must satisfy

$$\xi_i \geq \max\bigl\{0, 1 - y_i(\langle w, \Phi(x_i)\rangle + b)\bigr\} = L\bigl(y_i, \langle w, \Phi(x_i)\rangle + b\bigr),$$

where L is the hinge loss introduced earlier. Obviously, the objective function in (1.15) becomes minimal in ξ_i if this inequality is actually an equality. For a given $(w, b) \in H_0 \times \mathbb{R}$, let us now consider the function $f_{(w,b)} : X \to \mathbb{R}$ defined by $f_{(w,b)}(x) := \langle w, \Phi(x_i)\rangle + b$. Multiplying the objective function in (1.15) by $2\lambda := \frac{1}{nC}$ we can thus rewrite (1.15) in the form

$$\min_{(w,b)\in H_0\times\mathbb{R}} \lambda\langle w, w\rangle + \frac{1}{n}\sum_{i=1}^{n} L\big(y_i, f_{(w,b)}(x_i)\big). \tag{1.18}$$

Compared with the optimization problem (1.9), that is,

$$\inf_{f\in H} \lambda\|f\|_H^2 + \frac{1}{n}\sum_{i=1}^{n} L\big(y_i, f(x_i)\big),$$

there are obviously two major differences. The first one is that in the geometrical approach we consider a *general* Hilbert space H_0 and define a function $f_{(w,b)}$ in terms of an affine hyperplane specified by (w, b) in this space, while in (1.9) we start with an RKHS H and directly consider the functions contained in H. Remarkably, however, both approaches are equivalent if we fix b. More precisely, we will see in Section 4.2 that the functions $\langle w, \Phi(\,\cdot\,)\rangle$, $w \in H_0$, form an RKHS H whose norm can be computed by

$$\|f\|_H = \inf\big\{\|w\|_{H_0} : w \in H_0 \text{ with } f = \langle w, \Phi(\,\cdot\,)\rangle\big\}.$$

Consequently, (1.18) is equivalent to the optimization problem

$$\inf_{(f,b)\in H\times\mathbb{R}} \lambda\|f\|_H^2 + \frac{1}{n}\sum_{i=1}^{n} L\big(y_i, f(x_i) + b\big);$$

i.e., modulo the so-called *offset term* or *intercept term* b, the geometric approach is indeed equivalent to the RKHS approach (1.9). The offset term, however, makes a real difference and, in general, the decision functions produced by both approaches are different. This, of course, raises the question which optimization problem one should prefer. For very simple feature maps such as the identity map $\mathrm{id} : \mathbb{R}^d \to \mathbb{R}^d$, the offset term has obviously a clear advantage since it addresses translated data. Moreover, the version with the offset term is implemented in many standard software packages for SVMs. On the other hand, for more flexible feature maps such as those of Gaussian RBF kernels, which belong to the most important kernels in practice, the offset term has neither a known theoretical nor an empirical advantage. In addition, the theoretical analysis is often substantially complicated by the offset term. For the theoretical chapters of this book, we thus decided to exclusively consider the approach without an offset, while Chapter 11, which deals with computational aspects of SVMs, considers the approaches with and without an offset term. However, we sometimes mention theoretical results covering the offset term in the sections “Further Reading and Advanced Topics”, such as in Section 10.7 for robustness properties of SVMs.

The optimization problem (1.15) has the drawback that it has to be solved in an often high- or even infinite-dimensional Hilbert space H_0. We will see in Chapter 11 that in practice one thus uses the Lagrange approach to compute the corresponding dual program. For the hinge loss function, for example, this dual program is given by

$$\begin{aligned} \text{maximize } & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle \Phi(x_i), \Phi(x_j) \rangle \quad \text{over } \alpha \in [0, C]^n \\ \text{subject to } & \sum_{i=1}^{n} y_i \alpha_i = 0; \end{aligned}$$

see Example 11.3. Moreover, we will see that if $(\alpha_1^*, \ldots, \alpha_n^*)$ denotes a solution of this optimization problem, the solution (w_D^*, b_D^*) of (1.15) can be computed by

$$w_D^* = \sum_{i=1}^{n} y_i \alpha_i^* \Phi(x_i)$$

and

$$b_D^* = y_j - \sum_{i=1}^{n} y_i \alpha_i^* \langle \Phi(x_i), \Phi(x_j) \rangle,$$

where j is an index with $0 < \alpha_j^* < c$. Note that w_D^* only depends on the samples x_i whose weights satisfy $\alpha_i^* \neq 0$. Geometrically, this means that the affine hyperplane described by (w_D^*, b_D^*) is only "supported" by these $\Phi(x_i)$, and hence the corresponding data points (x_i, y_i) are called *support vectors*. As mentioned above, the decision function of the soft margin SVM is given by the constructed affine hyperplane,

$$f_{w_D^*, b_D^*}(x) = \langle w_D^*, \Phi(x) \rangle + b_D^* = \sum_{i=1}^{n} y_i \alpha_i^* \langle \Phi(x_i), \Phi(x) \rangle + b_D^* , \qquad x \in X.$$

Now note that in both the dual optimization problem and the evaluation of the resulting decision function only inner products of Φ with itself occur. Thus, instead of computing the feature map directly, it suffices to know the function $\langle \Phi(\cdot), \Phi(\cdot) \rangle : X \times X \to \mathbb{R}$. Interestingly, there do exist cases in which this function can be computed *without* knowing the feature map Φ itself. The Gaussian RBF kernels are examples of such a case, but there are many more. In Chapter 4, we will thus systematically investigate kernels; i.e., functions $k : X \times X \to \mathbb{R}$ for which there exists a feature map satisfying

$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle , \qquad x, x' \in X.$$

Obviously, using kernels directly instead of first computing feature maps works for all statistical methods and algorithms in which inner products of the feature map but not the feature map itself are needed. By using kernels, we can thus build a non-linear algorithm from a linear one without changing the core design of the algorithm. This observation, known as the *"kernel-trick,"* was to the best of our knowledge first explicitly stated by Schölkopf *et al.* (1998); however, it had already been used earlier by Aizerman *et al.* (1964) and Boser *et al.* (1992). Since then, various algorithms have been "kernelized," such as principal component analysis or Fisher's discriminant analysis. We refer to Schölkopf and Smola (2002) and Shawe-Taylor and Cristianini (2004) for detailed accounts. A second advantage of the kernel trick is that the input space X is no longer required to be a subset of $\mathbb{R}^d$ since all computations are done in the feature space. Interestingly, there exist various kernels that are defined on non-vectorial data such as text or DNA sequences. Therefore, the kernel trick indeed extends the applicability of methods that can be kernelized. We refer to Schölkopf and Smola (2002), Joachims (2002), Schölkopf *et al.* (2004), and Shawe-Taylor and Cristianini (2004) for various examples of kernel approaches for non-vectorial data.

1.4 Alternatives to SVMs

There exists a vast body of literature on both classification and regression procedures, and hence describing all these methods even briefly would fill yet another book. Nonetheless, it is always good to have alternatives, and hence we would like to briefly mention at least some of the most classical approaches. However, a comparison of a few of these methods is postponed to Section 12.2 on data mining, when our knowledge of SVMs will be more complete.

Besides the *least squares method*, which goes back to Gauss, Legendre, and Adrain, *linear discriminant analysis* is probably one of the oldest methods for pattern recognition. This procedure, which was developed by Sir R. A. Fisher in 1936, is strongly linked to multivariate normal distributions and uses affine hyperplanes as decision functions. A generalization of this procedure is *quadratic discriminant analysis*, which allows quadratic decision functions. Both methods are still used by many practitioners often with good success.

In 1956, one of the first iterative algorithms for learning a linear classification rule was proposed by Rosenblatt (1956, 1962) with the *perceptron*.

Another classical method is the *k-nearest-neighbor rule* which was introduced in 1951; see Fix and Hodges (1951, 1952). It has attracted many followers and is still used by many researchers. In addition, it was the first method for which universal consistency was established; see Stone (1977). The idea of k-nearest-neighbor methods for classificaton is to construct a decision function pointwise for each x by first determining the k points of D that are closest to x and then making the prediction for $y = 1$ if and only if the average of the k corresponding y-values is positive.

The goal in *cluster analysis* is to recognize clusters in unlabeled data. We refer to the books by Hartigan (1975) and Kaufman and Rousseeuw (2005) for an introduction to cluster analysis techniques.

Parametric *logistic regression* was proposed by Sir D. R. Cox to model binomial distributed outputs; see Cox and Snell (1989). This method is based on linear decision functions but does not make specific assumptions on the distribution of the inputs. Parametric logistic regression is a special case of *generalized linear models* in which the outputs are assumed to have a distribution from an exponential family, see McCullagh and Nelder (1989). Hastie and Tibshirani (1990) proposed a semi-parametric generalization called *generalized additive models* where the inputs may influence the outputs in an additive but not necessarily linear manner. The *lasso* (Tibshirani, 1996) is a method for regularizing a least squares regression. It minimizes the usual sum of squared errors, with a bound on the sum of the absolute values of the coefficients.

Other classical methods for classification and regression are *trees*, which were proposed by Breiman *et al.* (1984). The idea of trees is to partition the input space recursively into disjoint subsets such that points belonging to the same subset behave more homogeneously than points from different subsets. Trees often produce not only accurate results but are also able to uncover the predictive structure of the problem.

Neural networks in the context of machine learning are non-linear statistical data modeling tools that can be used to model complex relationships between inputs and outputs or to find patterns in data sets. In a neural network model, simple nodes (or "neurons") are connected together to form a network of nodes. The strength of the connections in the network depends on the data and may be time-dependent to allow for adaptivity. The motivation for neural networks, which were very popular in the 1990s, goes back to McCullogh and Pitts (1943) and Rosenblatt (1962). We refer also to Bishop (1996), Anthony and Bartlett (1999), and Vidyasagar (2002).

There also exist various other *kernel-based methods*. For *wavelets*, we refer to Daubechies (1991), and for *splines* to Wahba (1990). Recent developments for kernel-based methods in the context of SVMs are descibed by Cristianini and Shawe-Taylor (2000), Schölkopf and Smola (2002), and Shawe-Taylor and Cristianini (2004).

Boosting algorithms are based on an adaptive aggregation to construct from a set of weak learners a strong learner; see Schapire (1990), Freund (1995), and Freund and Schapire (1997).

Finally, the books by Hastie *et al.* (2001), Duda *et al.* (2001), and Bishop (2006) give a broad overview of various techniques used in statistical machine learning, whereas both Devroye *et al.* (1996) and Györfi *et al.* (2002) treat several classification and regression methods in a mathematically more rigorous way.