1 Introduction to Large Margin Classifiers

Alexander J. Smola
GMD FIRST
Rudower Chaussee 5
12489 Berlin, Germany
smola@first.gmd.de
http://www.first.gmd.de/∼smola

Peter Bartlett
Australian National University, RSISE
Canberra, ACT 0200, Australia
Peter.Bartlett@anu.edu.au
http://keating.anu.edu.au/people/∼bartlett

Bernhard Schölkopf
GMD FIRST
Rudower Chaussee 5
12489 Berlin, Germany
bs@first.gmd.de
http://www.first.gmd.de/∼bs

Dale Schuurmans
Department of Computer Science
Unversity of Waterloo
Waterloo, Ontario N2L 3G1, Canada
dale@cs.uwaterloo.ca
http://www.cs.uwaterloo.ca/∼dale

The aim of this chapter is to provide a brief introduction to the basic concepts of large margin classifiers for readers unfamiliar with the topic. Moreover it is aimed at establishing a common basis in terms of notation and equations, upon which the subsequent chapters will build (and refer to) when dealing with more advanced issues.

1.1 A Simple Classification Problem

training data

Assume that we are given a set of training data

$$X := \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subseteq \mathbb{R}^N \text{ where } m \in \mathbb{N} \tag{1.1}$$

labels

together with corresponding labels

$$Y := \{y_1, \ldots, y_m\} \subseteq \{-1, 1\}. \tag{1.2}$$

The goal is to find some decision function $g : \mathbb{R}^N \to \{-1, 1\}$ that accurately predicts the labels of unseen data points $(\mathbf{x}, y)$. That is, we seek a function g that minimizes the classification error, which is given by the probability that $g(\mathbf{x}) \neq y$. A common approach to representing decision functions is to use a real valued prediction function $f : \mathbb{R}^N \to \mathbb{R}$ whose output is passed through a sign threshold to yield the final classification $g(\mathbf{x}) = \operatorname{sgn}(f(\mathbf{x}))$. Let us start with a simple example: linear decision functions. In this case the unthresholded prediction is given by a simple linear function of the input vector $\mathbf{x}$

linear decision function

$$g(\mathbf{x}) := \operatorname{sgn}(f(\mathbf{x})) \text{ where } f(\mathbf{x}) = (\mathbf{x} \cdot \mathbf{w}) + b \text{ for } \mathbf{w} \in \mathbb{R}^N \text{ and } b \in \mathbb{R}. \tag{1.3}$$

This gives a classification rule whose decision boundary $\{\mathbf{x} | f(\mathbf{x}) = 0\}$ is an $N - 1$ dimensional hyperplane separating the classes "+1" and "−1" from each other. Figure 1.1 depicts the situation. The problem of learning from data can be

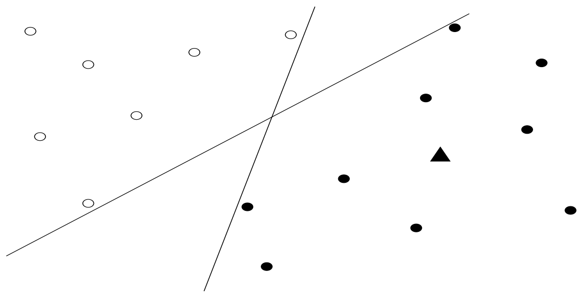

Figure 1.1 A linearly separable classification problem. Note that there may be several possible solutions as depicted by the two lines. The problem becomes non-separable if we replace the triangle by an open circle; in which case no solution $(\mathbf{w}, b)$ exists.

formulated as finding a set of parameters $(\mathbf{w}, b)$ such that $\operatorname{sgn}((\mathbf{w} \cdot \mathbf{x}_i) + b) = y_i$ for all $1 \leq i \leq m$. However, such a solution may not always exist, in particular if we are dealing with noisy data. For instance, consider Figure 1.1 with the triangle replaced by an open circle. This raises the question what to do in such a situation.

1.1.1 Bayes Optimal Solution

Under the assumption that the data X, Y was generated from a probability distribution $p(\mathbf{x}, y)$ on $\mathbb{R}^N \times \{-1, 1\}$ and that p is known, it is straightforward to find a function that minimizes the probability of misclassification

$$R(g) := \int_{\mathbb{R}^N \times \{-1,1\}} 1_{\{g(\mathbf{x}) \neq y\}} \; p(\mathbf{x}, y) d\mathbf{x} dy. \tag{1.4}$$

Bayes optimal decision function

This function satisfies

$$g(\mathbf{x}) = \operatorname{sgn}\left(p(\mathbf{x}, 1) - p(\mathbf{x}, -1)\right). \tag{1.5}$$

Consider a practical example.

Example 1.1 Two Gaussian Clusters
Assume that the two classes "+1" and "−1" are generated by two Gaussian clusters with the same covariance matrix Σ centered at $\boldsymbol{\mu}_+$ and $\boldsymbol{\mu}_-$ respectively

$$p(\mathbf{x}, y) = \frac{1}{2(2\sigma)^{N/2}|\Sigma|^{1/2}} \begin{cases} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_+)^\top \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_+)} & \text{if } y = +1 \\ e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_-)^\top \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_-)} & \text{if } y = -1. \end{cases} \tag{1.6}$$

Since the boundaries completely determine the decision function, we seek the set of points where $p(\mathbf{x}, +1) = p(\mathbf{x}, -1)$. In the case of (1.6) this is equivalent to seeking $\mathbf{x}$ such that

$$(\mathbf{x} - \boldsymbol{\mu}_+)^\top \Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu}_+) \;\; = \;\; (\mathbf{x} - \boldsymbol{\mu}_-)^\top \Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu}_-). \tag{1.7}$$

By rearranging we find that this condition is equivalent to

$$\begin{aligned} \mathbf{x}^\top \Sigma^{-1}\mathbf{x} - 2\boldsymbol{\mu}_+^\top \Sigma^{-1}\mathbf{x} + \boldsymbol{\mu}_+^\top \Sigma^{-1}\boldsymbol{\mu}_+ - \mathbf{x}^\top \Sigma^{-1}\mathbf{x} + 2\boldsymbol{\mu}_-^\top \Sigma^{-1}\mathbf{x} - \boldsymbol{\mu}_-^\top \Sigma^{-1}\boldsymbol{\mu}_- &= 0 \\ 2(\boldsymbol{\mu}_+^\top \Sigma^{-1} - \boldsymbol{\mu}_-^\top \Sigma^{-1})\mathbf{x} - (\boldsymbol{\mu}_+^\top \Sigma^{-1}\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-^\top \Sigma^{-1}\boldsymbol{\mu}_-) &= 0 \end{aligned} \tag{1.8}$$

The latter form is equivalent to having a linear decision function determined by

$$f(\mathbf{x}) = \left((\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-)^\top \Sigma^{-1}\right) \mathbf{x} - \tfrac{1}{2}(\boldsymbol{\mu}_+^\top \Sigma^{-1}\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-^\top \Sigma^{-1}\boldsymbol{\mu}_-). \tag{1.9}$$

linear discriminant

Hence in this simple example the Bayes optimal classification rule is linear.

Problems arise, however, if $p(\mathbf{x}, y)$ is not known (as generally happens in practice). In this case one has to obtain a good *estimate* of $g(\mathbf{x}) = \operatorname{sgn}(f(\mathbf{x}))$ from the training data X, Y. A famous example of an algorithm for linear separation is the perceptron algorithm.

1.1.2 The Perceptron Algorithm

The *perceptron algorithm* is "incremental," in the sense that small changes are made to the weight vector in response to each labelled example in turn. For any *learning rate* $\eta > 0$, the algorithm acts sequentially as shown in Table 1.1. Notice that $(\mathbf{w}, b)$ is only updated on a labelled example if the perceptron in state $(\mathbf{w}, b)$ *misclassifies* the example. It is convenient to think of the algorithm as maintaining

```
argument:  Training sample, X = {x_1,...,x_m} ⊂ 𝒳, Y = {y_1,...,y_m} ⊂ {±1}
           Learning rate, η
returns:   Weight vector w and threshold b.
function Perceptron(X, Y, η)
   initialize w, b = 0
   repeat
       for all i from i = 1,...,m
           Compute g(x_i) = sgn((w · x_i) + b)
           Update w, b according to
                  w' = w + (η/2)(y_i − g(x_i)) x_i
                  b' = b + (η/2)(y_i − g(x_i)).
       endfor
   until for all 1 ≤ i ≤ m we have g(x_i) = y_i
   return f : x ↦ (w · x) + b
end
```

Table 1.1 Basic Perceptron Algorithm.

perceptron algorithm

the hypothesis $g : \mathbf{x} \mapsto \operatorname{sgn}((\mathbf{w} \cdot \mathbf{x}) + b)$, which is updated each time it misclassifies an example. The algorithm operates on a training sample by repeatedly cycling through the m examples, and when it has completed a cycle through the training data without updating its hypothesis, it returns that hypothesis.

The following result shows that if the training sample is consistent with some simple perceptron, then this algorithm converges after a finite number of iterations. In this theorem, $\mathbf{w}^*$ and b^* define a decision boundary that correctly classifies all training points, and every training point is at least distance ρ from the decision boundary.

Theorem 1.1 Convergence of the Perceptron Algorithm
Suppose that there exists a $\rho > 0$, a weight vector $\mathbf{w}^*$ satisfying $\|\mathbf{w}^*\| = 1$, and a threshold b^* such that

$$y_i\left((\mathbf{w}^* \cdot \mathbf{x}_i) + b^*\right) \geq \rho \text{ for all } 1 \leq i \leq m. \tag{1.10}$$

Then for all $\eta > 0$, the hypothesis maintained by the perceptron algorithm converges after no more than $(b^{*2} + 1)(R^2 + 1)/\rho^2$ updates, where $R = \max_i \|x_i\|^2$. Clearly, the limiting hypothesis is consistent with the training data (X, Y).

Proof Let $(\mathbf{w}_j, b_j)$ be the state maintained immediately before the jth update occuring at, say, example $(\mathbf{x}_i, y_i)$. To measure the progress of the algorithm, we consider the evolution of the *angle* between $(\mathbf{w}_j, b_j)$ and $(\mathbf{w}^*, b^*)$ and note that the inner product $((\mathbf{w}_j, b_j) \cdot (\mathbf{w}^*, b^*))$ grows steadily with each update. To see this, note that $(\mathbf{w}_j, b_j)$ is only updated when the corresponding hypothesis g_j misclassifies y_i, which implies that $y_i - g_j(\mathbf{x}_i) = 2y_i$. Therefore,

$$\begin{aligned}((\mathbf{w}_{j+1}, b_{j+1}) \cdot (\mathbf{w}^*, b^*)) &= ([(\mathbf{w}_j, b_j) + (\eta/2)(y_i - g_j(\mathbf{x}_i))(\mathbf{x}_i, 1)] \cdot (\mathbf{w}^*, b^*)) \\ &= ((\mathbf{w}_j, b_j) \cdot (\mathbf{w}^*, b^*)) + \eta y_i((\mathbf{x}_i, 1) \cdot (\mathbf{w}^*, b^*))\end{aligned}$$

$$\geq ((\mathbf{w}_j, b_j) \cdot (\mathbf{w}^*, b^*)) + \eta\rho$$
$$\geq j\eta\rho.$$

On the other hand, the norm of $(\mathbf{w}_j, b_j)$ cannot grow too fast, because on an update we have $y_i((\mathbf{w}_j \cdot \mathbf{x}_i) + b_j) < 0$, and therefore

$$\begin{aligned}
\|(\mathbf{w}_{j+1}, b_{j+1})\|^2 &= \|(\mathbf{w}_j, b_j) + \eta y_i(\mathbf{x}_i, 1)\|^2 \\
&= \|(\mathbf{w}_j, b_j)\|^2 + 2\eta y_i((\mathbf{x}_i, 1) \cdot (\mathbf{w}_j, b_j)) + \eta^2\|(\mathbf{x}_i, 1)\|^2 \\
&\leq \|(\mathbf{w}_j, b_j)\|^2 + \eta^2\|(\mathbf{x}_i, 1)\|^2 \\
&\leq j\eta^2(R^2 + 1).
\end{aligned}$$

Combining these two observations with the Cauchy-Schwarz inequality shows that

$$\begin{aligned}
\sqrt{j\eta^2(R^2 + 1)} &\geq \|(\mathbf{w}_{j+1}, b_{j+1})\| \\
&\geq \frac{((\mathbf{w}_{j+1}, b_{j+1}) \cdot (\mathbf{w}^*, b^*))}{\sqrt{1 + b^{*2}}} \\
&\geq j\eta\rho,
\end{aligned}$$

and thus $j \leq (1 + b^{*2})(R^2 + 1)/\rho^2$ as desired. ■

Since the perceptron algorithm makes an update at least once in every cycle through the training data, and each iteration involves $O(N)$ computation steps, this theorem implies that the perceptron algorithm has time complexity $O((R^2 + 1)mN/\rho^2)$.

1.1.3 Margins

The quantity ρ plays a crucial role in the previous theorem, since it determines how well the two classes can be separated and consequently how fast the perceptron learning algorithm converges. This quantity ρ is what we shall henceforth call a *margin*.

Definition 1.1 Margin and Margin Errors
Denote by $f : \mathbb{R}^N \to \mathbb{R}$ a real valued hypothesis used for classification. Then

margin

$$\rho_f(\mathbf{x}, y) := yf(\mathbf{x}), \tag{1.11}$$

i.e. it is the margin by which the pattern $\mathbf{x}$ is classified correctly (so that a negative value of $\rho_f(\mathbf{x}, y)$ corresponds to an incorrect classification). Moreover denote by

minimum margin

$$\rho_f := \min_{1 \leq i \leq m} \rho_f(\mathbf{x}_i, y_i) \tag{1.12}$$

the minimum margin over the whole sample. It is determined by the "worst" classification on the whole training set X, Y.

It appears to be desirable to have classifiers that achieve a large margin ρ_f since one might expect that an estimate that is "reliable" on the training set will also perform well on unseen examples. Moreover such an algorithm is more robust with respect to both patterns and parameters:

- Intuitively, for a pattern $\mathbf{x}$ that is far from the decision boundary $\{\mathbf{x}|f(\mathbf{x})=0\}$ slight perturbations to $\mathbf{x}$ will not change its classification $\operatorname{sgn}(f(\mathbf{x}))$. To see this, note that if $f(\mathbf{x})$ is a continuous function in $\mathbf{x}$ then small variations in $\mathbf{x}$ will translate into small variations in $f(\mathbf{x})$. Therefore, if $y_i f(\mathbf{x}_i)$ is much larger than zero, $y_i f(\mathbf{x}_i \pm \varepsilon)$ will also be positive for small ε. (See, for example, Duda and Hart (1973).)

robustness in patterns

- Similarly, a slight perturbation to the function f will not affect any of the resulting classifications on the training data $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_m, y_m)$. Assume that $f_{\mathbf{w}}(\mathbf{x})$ is continuous in its parameters $\mathbf{w}$. Then, again, if $y_i f_{\mathbf{w}}(\mathbf{x}_i)$ is much larger than zero, $y_i f_{\mathbf{w}\pm\varepsilon}(\mathbf{x}_i)$ will also be positive for small ε.

robustness in parameters

1.1.4 Maximum Margin Hyperplanes

As pointed out in the previous section, it is desirable to have an estimator with a large margin. This raises the question whether there exists an estimator with *maximum* margin, i.e. whether there exists some f^* with

$$f^* := \operatorname*{argmax}_f \rho_f = \operatorname*{argmax}_f \min_i y_i f(\mathbf{x}_i). \tag{1.13}$$

Without some constraint on the size of $\mathbf{w}$, this maximum does not exist. In Theorem 1.1, we constrained $\mathbf{w}^*$ to have unit length. If we define $f : \mathbb{R}^N \to \mathbb{R}$ by

$$f(\mathbf{x}) = \frac{(\mathbf{w}\cdot\mathbf{x}) + b}{\|\mathbf{w}\|}, \tag{1.14}$$

then the maximum margin f is defined by the weight vector and threshold that satisfy

optimal hyperplane

$$\mathbf{w}^*, b^* = \operatorname*{argmax}_{\mathbf{w},b} \min_{i=1}^{m} \frac{y_i((\mathbf{w}\cdot\mathbf{x}_i)+b)}{\|\mathbf{w}\|} \tag{1.15}$$

$$= \operatorname*{argmax}_{\mathbf{w},b} \min_{i=1}^{m} \; y_i \; \operatorname{sgn}((\mathbf{w}\cdot\mathbf{x}_i)+b) \; \left\| \frac{(\mathbf{w}\cdot\mathbf{x}_i)}{\|\mathbf{w}\|^2}\mathbf{w} + \frac{b}{\|\mathbf{w}\|^2}\mathbf{w} \right\| \tag{1.16}$$

Euclidean Margin

The formulation (1.16) has a simple geometric interpretation: $-b\mathbf{w}/\|\mathbf{w}\|^2$ is the vector in direction $\mathbf{w}$ that ends right on the decision hyperplane (since $\left(\mathbf{w}\cdot(-b\mathbf{w}/\|\mathbf{w}\|^2)\right) = -b$), and for a vector $\mathbf{x}_i$, $(\mathbf{w}\cdot\mathbf{x}_i)\mathbf{w}/\|\mathbf{w}\|^2$ is the projection of $\mathbf{x}_i$ onto $\mathbf{w}$. Therefore, we are interested in maximizing the length of the vector differences $(\mathbf{w}\cdot\mathbf{x}_i)\mathbf{w}/\|\mathbf{w}\|^2 - (-b\mathbf{w}/\|\mathbf{w}\|^2)$ appropriatedly signed by $y_i g(\mathbf{x}_i)$.

The maxi-min problem (1.15) can be easily transformed into an equivalent constrained optimization task by conjecturing a lower bound on the margin, ρ, and maximizing ρ subject to the constraint that it really is a lower bound:

optimization problems

$$\mathbf{w}^*, b^*, \rho^*$$

$$= \operatorname*{argmax}_{\mathbf{w},b,\rho} \; \rho \quad \text{subject to } \frac{y_i((\mathbf{w}\cdot\mathbf{x}_i)+b)}{\|\mathbf{w}\|} \geq \rho \text{ for } 1 \leq i \leq m \tag{1.17}$$

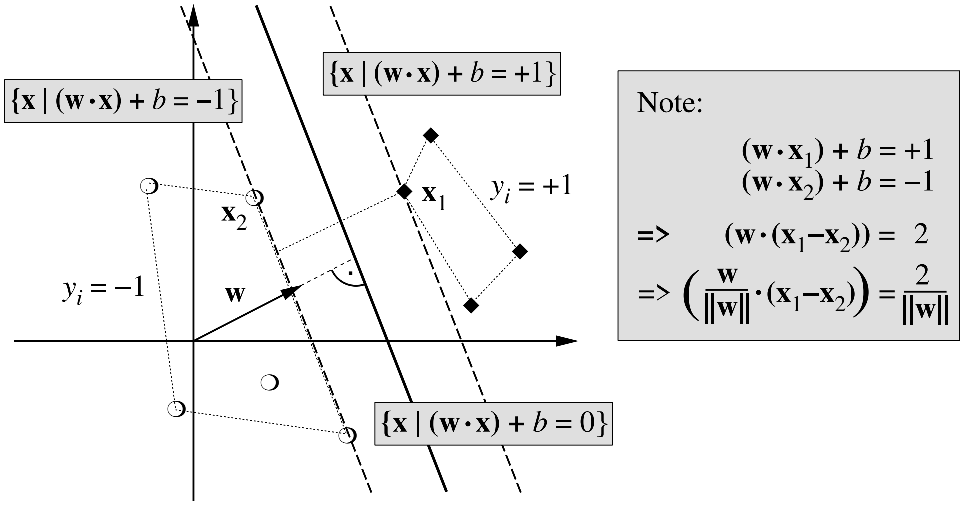

Figure 1.2 A binary classification toy problem: separate balls from diamonds. The *optimal hyperplane* is orthogonal to the shortest line connecting the convex hulls of the two classes (dotted), and intersects it half-way between the two classes. The problem being separable, there exists a weight vector $\mathbf{w}$ and a threshold b such that $y_i((\mathbf{w} \cdot \mathbf{x}_i) + b) > 0$ $(i = 1, \ldots, m)$. Rescaling $\mathbf{w}$ and b such that the point(s) closest to the hyperplane satisfy $|(\mathbf{w} \cdot \mathbf{x}_i) + b| = 1$, we obtain a *canonical* form $(\mathbf{w}, b)$ of the hyperplane, satisfying $y_i((\mathbf{w} \cdot \mathbf{x}_i) + b) \geq 1$. Note that in this case, the minimum Euclidean distance between the two classes (i.e. twice the margin), measured perpendicularly to the hyperplane, equals $2/\|\mathbf{w}\|$. This can be seen by considering two points $\mathbf{x}_1, \mathbf{x}_2$ on opposite sides of the margin, i.e. $(\mathbf{w} \cdot \mathbf{x}_1) + b = 1$, $(\mathbf{w} \cdot \mathbf{x}_2) + b = -1$, and projecting them onto the hyperplane normal vector $\mathbf{w}/\|\mathbf{w}\|$.

$$= \operatorname*{argmax}_{\mathbf{w},b,\rho} \quad \rho \quad \text{subject to } \|\mathbf{w}\| = 1 \text{ and } y_i((\mathbf{w} \cdot \mathbf{x}_i) + b) \geq \rho \text{ for } 1 \leq i \leq m \tag{1.18}$$

$$= \operatorname*{argmin}_{\mathbf{w},b} \quad \|\mathbf{w}\|^2 \quad \text{subject to } y_i((\mathbf{w} \cdot \mathbf{x}_i) + b) \geq 1 \text{ for } 1 \leq i \leq m \tag{1.19}$$

quadratic program

This last formulation is in the form of a quadratic programming problem, which can be easily handled using standard numerical routines (Luenberger, 1973; Bertsekas, 1995).

Notice that (1.18) is in a particularly intuitive form. This formulation states that we are seeking a weight vector $\mathbf{w}$ that obtains large dot products $y_i(\mathbf{w} \cdot \mathbf{x}_i)$, but constrain the weight vector to lie on the unit sphere to prevent obtaining such large dot products "for free" by scaling up $\mathbf{w}$. Interesting variants of problem (1.18) are obtained by choosing different norms to constrain the length of the weight vector. For example, constraining $\mathbf{w}$ to lie on the unit ℓ_1 sphere instead of the unit ℓ_2 sphere gives the problem of determining

$\mathbf{w}^*, b^*, \rho^*$

$$= \operatorname*{argmax}_{\mathbf{w},b,\rho} \rho \quad \text{subject to } \|\mathbf{w}\|_1 = 1 \text{ and } y_i((\mathbf{w}\cdot\mathbf{x}_i)+b) \geq \rho \text{ for } 1 \leq i \leq m \qquad (1.20)$$

ℓ_∞ margin

which can easily be shown to be in the form of a linear programming problem. Mangasarian (1997) shows that this is equivalent to finding the weight vector and threshold that maximize the minimum ℓ_∞ distance between the training patterns and the decision hyperplane, in a direct analogue to the original Euclidean formulation (1.15).

Similarly, the constraint that $\mathbf{w}$ lie on the unit ℓ_∞ sphere yields the problem

$$\mathbf{w}^*, b^*, \rho^*$$
$$= \operatorname*{argmax}_{\mathbf{w},b,\rho} \rho \quad \text{subject to } \|\mathbf{w}\|_\infty = 1 \text{ and } y_i((\mathbf{w}\cdot\mathbf{x}_i)+b) \geq \rho \text{ for } 1 \leq i \leq m \qquad (1.21)$$

ℓ_1 margin

which is also a linear programming problem, but now equivalent to finding the weight vector and threshold that maximize the minimum ℓ_1 distance between the training patterns and the decision hyperplane. In general, constraining $\mathbf{w}$ to lie on the unit ℓ_p sphere yields a convex programming problem

$$\mathbf{w}^*, b^*, \rho^*$$
$$= \operatorname*{argmax}_{\mathbf{w},b,\rho} \rho \quad \text{subject to } \|\mathbf{w}\|_p = 1 \text{ and } y_i((\mathbf{w}\cdot\mathbf{x}_i)+b) \geq \rho \text{ for } 1 \leq i \leq m \qquad (1.22)$$

ℓ_q margin

which is equivalent to finding the weight vector and threshold that maximize the minimum ℓ_q distance between the training patterns and the decision hyperplane, where ℓ_p and ℓ_q are conjugate norms, i.e. such that $\frac{1}{p}+\frac{1}{q}=1$ (Mangasarian, 1997).

In solving any of these constrained optimization problems, there is a notion of *critical constraints*; i.e. those inequality constraints that are satisfied as equalities by the optimal solution. In our setting, constraints correspond to training examples $(\mathbf{x}_i, y_i)$, $1 \leq i \leq m$, and the *critical* constraints are given by those training examples that lie right on the margin a distance ρ from the optimal hyperplane (cf. Figure 1.2). These critical training patterns are called *Support Vectors*.

Support Vectors

Notice that all the remaining examples of the training set are irrelevant: for non-critical examples the corresponding constraint $y_i((\mathbf{w}\cdot\mathbf{x}_i)+b) \geq 1$ in (1.19) does not play a role in the optimization, and therefore these points could be removed from the training set without affecting the results. This nicely captures our intuition of the problem: the hyperplane (cf. Figure 1.2) is completely determined by the patterns closest to it, the solution should not depend on the other examples.

soft margin hyperplane

In practice, a separating hyperplane may not exist, e.g. if a high noise level causes a large overlap of the classes. The previous maximum margin algorithms perform poorly in this case because the maximum achievable minimum margin is negative, and this means the critical constraints are the mislabelled patterns that are furthest from the decision hyperplane. That is, the solution hyperplane is determined entirely by misclassified examples! To overcome the sensitivity to noisy training patterns, a standard approach is to allow for the possibility of examples violating the constraint in (1.19) by introducing *slack variables* (Cortes and Vapnik, 1995; Vapnik, 1995b)

slack variables

$$\xi_i \geq 0, \text{ for all } i = 1, \ldots, m, \tag{1.23}$$

along with relaxed constraints

$$y_i((\mathbf{w} \cdot \mathbf{x}_i) + b) \geq 1 - \xi_i, \text{ for all } i = 1, \ldots, m. \tag{1.24}$$

A classifier which generalizes well is then found by controlling both the size of $\mathbf{w}$ and the number of training errors, minimizing the objective function

$$\tau(\mathbf{w}, \boldsymbol{\xi}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \xi_i \tag{1.25}$$

subject to the constraints (1.23) and (1.24), for some value of the constant $C > 0$.

In the following section, we shall see why the size of $\mathbf{w}$ is a good measure of the complexity of the classifier.

1.2 Theory

In order to provide a theoretical analysis of the learning problem we have to introduce a few definitions and assumptions about the process generating the data.

1.2.1 Basic Assumptions

independently identically distributed

We assume that the training data X, Y is drawn independently and identically distributed (iid) according to some probability measure $p(\mathbf{x}, y)$. This means that all examples $(\mathbf{x}_i, y_i)$ are drawn from $p(\mathbf{x}, y)$ regardless of the other examples or the index i.

This assumption is stronger than it may appear at first glance. For instance, time series data fails to satisfy the condition, since the observations are typically dependent, and their statistics might depend on the index i.

In (1.4), we defined the functional $R(g)$ of a decision function g as the probability of misclassification. We can generalize this definition to apply to prediction functions f as well as thresholded decision functions g. This yields what we call the risk functional.

Definition 1.2 Risk Functional
Denote by $c(\mathbf{x}, y, f(\mathbf{x})) : \mathbb{R}^N \times \mathbb{R} \times \mathbb{R} \to [0, \infty)$ a cost function and by $p(\mathbf{x}, y)$ a probability measure as described above. Then the risk functional for a function $f : \mathbb{R}^N \to \mathbb{R}$ is defined as

Expected Risk

$$R(f) := \int_{\mathbb{R}^N \times \mathbb{R}} c(\mathbf{x}, y, f(\mathbf{x}))\, p(\mathbf{x}, y)\, d\mathbf{x}dy. \tag{1.26}$$

Moreover the *empirical* risk functional for an m–sample X, Y is given by

$$R_{\mathrm{emp}}(f) := \frac{1}{m}\sum_{i=1}^{m} c(\mathbf{x}_i, y_i, f(\mathbf{x}_i)). \tag{1.27}$$

Empirical Risk

For thresholded decision functions $g : \mathbb{R}^N \to \{-1, 1\}$ we often use 0–1 classification error as the cost function $c(\mathbf{x}, y, g(\mathbf{x})) = 1_{\{g(\mathbf{x}) \neq y\}}$. In this case we obtain the risk functional defined in (1.4) (the probability of misclassification),

$$R(g) := \Pr\{g(\mathbf{x}) \neq y\}. \tag{1.28}$$

In this case, the empirical risk functional is

$$R_{\mathrm{emp}}(g) := \frac{1}{m}\sum_{i=1}^{m} 1_{\{g(\mathbf{x}_i) \neq y_i\}}, \tag{1.29}$$

which is just the training error.

margin error

Finally we need a quantity called the *margin error* which is given by the proportion of training points that have margin less than ρ, i.e.

$$R_\rho(f) := \frac{1}{m}\sum_{i=1}^{m} 1_{\{y_i f(\mathbf{x}_i) < \rho\}}. \tag{1.30}$$

This empirical estimate of risk counts a point as an error if it is either incorrectly classified or correctly classified by with margin less than ρ.

While one wants to minimize the risk $R(g)$ this is hardly ever possible since $p(\mathbf{x}, y)$ is unknown. Hence one may only resort to minimizing $R_{\mathrm{emp}}(g)$ which is based on the training data. This, however, is not an effective method by itself—just consider an estimator that memorizes all the training data X, Y and generates random outputs for any other data. This clearly would have an empirical risk $R_{\mathrm{emp}}(g) = 0$ but would obtain a true risk $R(g) = 0.5$ (assuming the finite training sample has measure 0). The solution is to take the complexity of the estimate g into account as well, which will be discussed in the following sections.

1.2.2 Error Bounds for Thresholded Decision Functions

The central result of this analysis is to relate the number of training examples, the training set error, and the complexity of the hypothesis space to the generalization error. For thresholded decision functions, an appropriate measure for the complexity of the hypothesis space is the Vapnik-Chervonenkis (VC) dimension.

VC dimension

Definition 1.3 VC dimension
The VC dimension h of a space of $\{-1, 1\}$-valued functions, G, is the size of the largest subset of domain points that can be labelled arbitrarily by choosing functions only from G (Vapnik and Chervonenkis, 1971).

The VC dimension can be used to prove high probability bounds on the error of a hypothesis chosen from a class of decision functions G—this is the famous result of Vapnik and Chervonenkis (1971). The bounds have since been improved slightly by Talagrand (1994)—see also (Alexander, 1984).

Theorem 1.2 VC Upper Bound
Let G be a class of decision functions mapping $\mathbb{R}^N$ to $\{-1, 1\}$ that has VC dimension h. For any probability distribution $p(\mathbf{x}, y)$ on $\mathbb{R}^N \times \{-1, 1\}$, with probability at least $1 - \delta$ over m random examples $\mathbf{x}$, for any hypothesis g in G the risk functional with 0–1 loss is bounded by

$$R(g) \leq R_{\text{emp}}(g) + \sqrt{\frac{c}{m}\left(h + \ln\left(\frac{1}{\delta}\right)\right)} \tag{1.31}$$

where c is a universal constant. Furthermore, if $g^* \in G$ minimizes $R_{\text{emp}}(\cdot)$, then with probability $1 - \delta$

$$R(g^*) \leq \inf_{g \in G} R(g) + \sqrt{\frac{c}{m}\left(h + \ln\left(\frac{1}{\delta}\right)\right)} \tag{1.32}$$

(A short proof of this result is given by Long (1998), but with worse constants than Talagrand's.)

These upper bounds are asymptotically close to the best possible, since there is also a lower bound with the same form:

Theorem 1.3 VC Lower Bound
Let G be a hypothesis space with finite VC dimension $h \geq 1$. Then for any learning algorithm there exist distributions such that with probability at least δ over m random examples, the error of its hypothesis g satisfies

$$R(g) \geq \inf_{g' \in G} R(g') + \sqrt{\frac{c}{m}\left(h + \ln\left(\frac{1}{\delta}\right)\right)} \tag{1.33}$$

where c is a universal constant.

(Results of this form have been given by Devroye and Lugosi (1995); Simon (1996); Anthony and Bartlett (1999), using ideas from Ehrenfeucht et al. (1989).)

Theorems 1.2 and 1.3 give a fairly complete characterization of the generalization error that can be achieved by choosing decision functions from a class G. However, this characterization suffers from two drawbacks.

- The first drawback is that the VC dimension must actually be determined (or at least bounded) for the class of interest—and this is often not easy to do. (However, bounds on the VC dimension h have been computed for many natural decision function classes, including parametric classes involving standard arithmetic and boolean operations. See Anthony and Bartlett (1999) for a review of these results.)
- The second (more serious) drawback is that the analysis ignores the *structure* of the mapping from training samples to hypotheses, and concentrates solely on the *range* of the learner's possible outputs. Ignoring the details of the learning map can omit many of the factors that are *crucial* for determining the success of the

learning algorithm in real situations.

For example, consider learning algorithms that operate by first computing a real valued prediction function f from some class F and then thresholding this hypothesis to obtain the final decision function $g(\mathbf{x}) = \operatorname{sgn}(f(\mathbf{x}))$. Here, the VC dimension is a particularly weak method for measuring the representational capacity of the resulting function class $G = \operatorname{sgn}(F)$.

One reason is that the VC dimension of G is not sensitive to the *scale* of F at the accuracy level of interest. That is, it does not pay attention to whether the complexity of the hypothesis class is at a scale that is relevant for the outcome of the predictions.

The first step towards a more refined analysis that takes scale into account is given by Vapnik (1979). Consider a set $X_0 \subset \mathbb{R}^N$ of input points with norm bounded by $R > 0$ (that is, $\|\mathbf{x}_i\| \leq R$ for $\mathbf{x} \in X_0$), and the set F of bounded linear functions defined on X_0,

$$F = \{\mathbf{x} \mapsto (\mathbf{w} \cdot \mathbf{x}) \ \mid \ \|\mathbf{w}\| \leq 1, \mathbf{x} \in X_0\} \tag{1.34}$$

satisfying $|f(\mathbf{x})| \geq \rho$ for all patterns $\mathbf{x}$ in X_0. Then if we consider the set G of linear decision functions obtained by thresholding functions in F, Vapnik (1979) shows

$$\mathrm{VCdim}(G) \leq \min\{R^2/\rho^2, N\} + 1. \tag{1.35}$$

Note that this can be much smaller than the VC dimension of $\operatorname{sgn}(F)$ obtained without taking ρ into account, which is $N + 1$ in this case. Therefore, one could hope to obtain significant benefits by using scale sensitive bounds which give much tighter results for large margin classifiers. Unfortunately, the bound (1.35) does not yet suffice for our purposes, because note that it requires that *all* points (including the test points) satisfy the margin condition, and therefore theorem 1.2 does not apply in this case. Rigorously obtaining these scale sensitive improvements is the topic we now address. In the following section, we consider scale-sensitive versions of the VC dimension, and obtain upper and lower bounds on risk in terms of these dimensions.

1.2.3 Margin Dependent Error Bounds for Real Valued Predictors

Definition 1.4 Fat Shattering Dimension

Let F be a set of real valued functions. We say that a set of points $S \subset \mathcal{X}$, which we will index as a vector $\mathbf{x} \in \mathcal{X}^{|S|}$, is *$\rho$-shattered by F* if there is a vector of real numbers $\mathbf{b} \in \mathbb{R}^{|S|}$ such that for any choice of signs $\mathbf{y} \in \{-1, 1\}^{|S|}$ there is a function f in F that satisfies

$$y_i(f(x_i) - b_i) \geq \rho \text{ for } 1 \leq i \leq |S|. \tag{1.36}$$

(That is, $f(x_i) \geq b_i + \rho$ if $y_i = 1$, and $f(x_i) \leq b_i - \rho$ if $y_i = -1$, for all x_i in S. Notice how similar this is to the notion of a minimum margin defined by (1.12).) The *fat shattering dimension* $\mathrm{fat}_F(\rho)$ of the set F is a function from the positive real fat shattering

numbers to the integers which maps a value ρ to the size of the largest ρ-shattered set, if this is finite, or infinity otherwise.

We may think of the fat-shattering dimension of a set of real-valued functions as the VC dimension obtained by thresholding but requiring that outputs are ρ above the threshold for positive classification and ρ below for negative.

The fat-shattering dimension is closely related to a more basic quantity, the covering number of a class of functions.

Definition 1.5 Covering Numbers of a Set
Denote by (S, d) a pseudometric space, $B_r(\mathbf{x})$ the closed ball in S centred at $\mathbf{x}$ with radius r, T a subset of S, and ε some positive constant. Then the covering number $\mathcal{N}(\varepsilon, T)$ is defined as the minimum cardinality (that is, number of elements) of a set of points $T' \subset S$ such that

covering number

$$T \subseteq \bigcup_{\mathbf{x}_i \in T'} B_\varepsilon(\mathbf{x}_i), \tag{1.37}$$

i.e. such that the maximum difference of any element in T and the closest element in T' is less than or equal to ε.

Covering a class of functions F with an ε-cover means that one is able to approximately represent F (which may be of infinite cardinality) by a finite set. For learning, it turns out that it suffices to approximate the restrictions of functions in a class F to finite samples. For a subset X of some domain $\mathcal{X}$, define the pseudometric $\ell_{\infty,X}$ by

$$\ell_{\infty,X}(f, f') = \max_{\mathbf{x} \in X} |f(\mathbf{x}) - f'(\mathbf{x})| \tag{1.38}$$

where f and f' are real-valued functions defined on $\mathcal{X}$. Let $\mathcal{N}(\varepsilon, F, m)$ denote the maximum, over all $X \subset \mathcal{X}$ of size $|X| = m$, of the covering number $\mathcal{N}(\varepsilon, F)$ with respect to $\ell_{\infty,X}$. The following theorem shows that the fat-shattering dimension is intimately related to these covering numbers. (The upper bound is due to Alon et al. (1997), and the lower bound to Bartlett et al. (1997).)

Theorem 1.4 Bounds on $\mathcal{N}$ in terms of fat_F
Let F be a set of real functions from a domain $\mathcal{X}$ to the bounded interval $[0, B]$. Let $\varepsilon > 0$ and let $m \geq \mathrm{fat}_F(\varepsilon/4)$. Then

$$\frac{\log_2 e}{8} \mathrm{fat}_F(16\varepsilon) \;\leq\; \log_2 \mathcal{N}(\varepsilon, F, m) \;\leq\; 3\,\mathrm{fat}_F(\tfrac{\varepsilon}{4}) \log_2^2 \left(\frac{4eBm}{\varepsilon}\right). \tag{1.39}$$

Unfortunately, directly bounding $\mathcal{N}$ can be quite difficult in general. Useful tools from functional analysis (which deal with the functional inverse of $\mathcal{N}$ wrt. ϵ, the so called entropy number) for obtaining these bounds have been developed for classes of functions F defined by linear mappings from Hilbert spaces (Carl and Stephani, 1990), and linear functions over kernel expansions (Williamson et al., 1998b).

The following result shows that we can use covering numbers to obtain upper bounds on risk in terms of margin error (Shawe-Taylor et al., 1998; Bartlett, 1998).

Theorem 1.5 Bounds on $R(f)$ in terms of $\mathcal{N}$ and ρ
Suppose that F is a set of real-valued functions defined on $\mathcal{X}$, $\varepsilon \in (0,1)$ and $\rho > 0$. Fix a probability distribution on $\mathcal{X} \times \{-1, 1\}$ and a sample size m. Then the probability that some f in F has $R_\rho(f) = 0$ but $R(f) \geq \varepsilon$ is no more than

$$2\,\mathcal{N}\left(\tfrac{\rho}{2}, F, 2m\right)\, 2^{-\varepsilon m/2}. \tag{1.40}$$

Furthermore,

$$\Pr\left(\text{“some } f \text{ in } F \text{ has } R(f) \geq R_\rho(f) + \varepsilon\text{”}\right) \;\leq\; 2\,\mathcal{N}\left(\tfrac{\rho}{2}, F, 2m\right)\, e^{-\varepsilon^2 m/8}. \tag{1.41}$$

In fact, it is possible to obtain a similar result that depends only on the behaviour of functions in F near the threshold (see (Anthony and Bartlett, 1999) for details).

anatomy of a uniform convergence bound

Let us have a close look at the bound (1.41) on the probability of excessive error. The factor $e^{-\varepsilon^2 m/8}$ in (1.41) stems from a bound of Hoeffding (1963) on the probability of a large deviation of a sum of random variables from its mean. The factor $\mathcal{N}\left(\frac{\rho}{2}, F, 2m\right)$ stems from the fact that the continuous class of functions F was approximated (to accuracy $\rho/2$) by a finite number of functions. The $2m$ is due to the use of a symmetrization argument which is needed to make the overall argument work. Theorem 1.4 shows that this term is bounded by an exponential function of the fat-shattering dimension at scale $\rho/8$.

Interestingly, a similar result holds in regression. (For a review of these uniform convergence results, see (Anthony and Bartlett, 1999).

Theorem 1.6 Bounds on $R(f)$ for Regression
Suppose that F is a set of functions defined on a domain $\mathcal{X}$ and mapping into the real interval $[0,1]$. Let p be any probability distribution on $\mathcal{X} \times [0,1]$, ε any real number between 0 and 1, and $m \in \mathbb{N}$. Then for the quadratic cost function $c(\mathbf{x}, y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2$ we have

$$\Pr\left(\sup_{f\in F} |R(f) - R_{\text{emp}}(f)| \geq \varepsilon\right) \;\leq\; 4\,\mathcal{N}\left(\tfrac{\varepsilon}{16}, F, 2m\right)\, e^{-\varepsilon^2 m/32}. \tag{1.42}$$

Comparing with (1.41), notice that the scale of the covering number depends on the desired accuracy ε, whereas in (1.41) it depends on the scale ρ at which the margins are examined.

1.2.4 Error Bounds for Linear Decision Functions

The following result, due to Bartlett and Shawe-Taylor (1999), gives a bound on the fat-shattering dimension of large margin linear classifiers. It has a similar form to the bound (1.35) on the VC dimension of linear functions restricted to certain sets. It improves on a straightforward corollary of that result, and on a result of Gurvits (1997).

Theorem 1.7 Fat Shattering Dimension for Linear Classifiers
Suppose that B_R is the ℓ_2 ball of radius R in $\mathbb{R}^n$, centered at the origin, and consider the set

$$F := \{f_{\mathbf{w}} \mid f_{\mathbf{w}}(\mathbf{x}) = (\mathbf{w} \cdot \mathbf{x}) \text{ with } \|\mathbf{w}\| \leq 1, \mathbf{x} \in B_R\}. \tag{1.43}$$

Then

$$\text{fat}_F(\rho) \leq \left(\tfrac{R}{\rho}\right)^2. \tag{1.44}$$

Using this result together with Theorems 1.4 and 1.5 gives the following theorem.

Theorem 1.8 Error Bounds for Linear Classifiers
Define the class F of real-valued functions on the ball of radius R as in (1.43). There is a constant c such that, for all probability distributions, with probability at least $1-\delta$ over m independently generated training examples, every $\rho > 0$ and every function $f \in F$ with margin at least ρ on all training examples (i.e. $R_\rho(f) = 0$) satisfies

$$R(f) \leq \frac{c}{m}\left(\frac{R^2}{\rho^2}\log^2\left(\frac{m}{\rho}\right) + \log\left(\frac{1}{\delta}\right)\right). \tag{1.45}$$

Furthermore, with probability at least $1-\delta$, for all $\rho > 0$, every function f in F has error

$$R(f) \leq R_\rho(f) + \sqrt{\frac{c}{m}\left(\frac{R^2}{\rho^2}\log^2\left(\frac{m}{\rho}\right) + \log\left(\frac{1}{\delta}\right)\right)}. \tag{1.46}$$

For estimators using a linear programming approach as in (Mangasarian, 1968) one may state the following result, which then, via Theorem 1.4 can be transformed into a generalization bound as well.

Theorem 1.9 Capacity Bounds for Linear Classifiers
There is a constant c such that for the class

$$F_R = \left\{\mathbf{x} \mapsto \mathbf{w}^T\mathbf{x} \mid \|\mathbf{x}\|_\infty \leq 1, \|\mathbf{w}\|_1 \leq R\right\} \tag{1.47}$$

we have

$$\text{fat}_{F_R}(\varepsilon) \leq c\left(\frac{R}{\varepsilon}\right)^2 \ln(2N+2). \tag{1.48}$$

Finally, we can obtain bounds for convex combinations of arbitrary hypotheses from a class G of $\{-1, 1\}$-valued functions,

$$\text{co}(G) = \left\{\sum_i \alpha_i g_i \;\middle|\; \alpha_i > 0, \sum_i \alpha_i = 1, g_i \in G\right\}. \tag{1.49}$$

See (Schapire et al., 1998). These bounds are useful in analysing boosting algorithms; see Section 1.4.

Theorem 1.10 Bounds for Convex Combinations of Hypotheses
Let $p(\mathbf{x}, y)$ be a distribution over $\mathcal{X} \times \{-1, 1\}$, and let X be a sample of m examples chosen iid according to p. Suppose the base-hypothesis space G has VC dimension h, and let $\delta > 0$. Then with probability at least $1 - \delta$ over the random choice of the training set X, Y, every convex combination of functions $f \in \operatorname{co}(G)$ satisfies the following bound for all $\rho > 0$.

$$R(f) \leq R_\rho(f) + \sqrt{\frac{c}{m}\left(\frac{h \log^2(m/h)}{\rho^2} + \log\left(\frac{1}{\delta}\right)\right)} \tag{1.50}$$

1.3 Support Vector Machines

1.3.1 Optimization Problem

To construct the *Optimal Hyperplane* (cf. Figure 1.2), one solves the following optimization problem:

$$\text{minimize} \quad \tau(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 \tag{1.51}$$

$$\text{subject to} \quad y_i((\mathbf{w} \cdot \mathbf{x}_i) + b) \geq 1, \text{ for all } i = 1, \ldots, m. \tag{1.52}$$

Lagrangian

This constrained optimization problem is dealt with by introducing Lagrange multipliers $\alpha_i \geq 0$ and a Lagrangian

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i \left(y_i((\mathbf{x}_i \cdot \mathbf{w}) + b) - 1\right). \tag{1.53}$$

The Lagrangian L has to be minimized with respect to the *primal variables* $\mathbf{w}$ and b and maximized with respect to the *dual variables* α_i (i.e. a saddle point has to be found). Let us try to get some intuition for this. If a constraint (1.52) is violated, then $y_i((\mathbf{w} \cdot \mathbf{x}_i) + b) - 1 < 0$, in which case L can be increased by increasing the corresponding α_i. At the same time, $\mathbf{w}$ and b will have to change such that L decreases. To prevent $-\alpha_i\left(y_i((\mathbf{w} \cdot \mathbf{x}_i) + b) - 1\right)$ from becoming arbitrarily large, the change in $\mathbf{w}$ and b will ensure that, provided the problem is separable, the constraint will eventually be satisfied.

KKT conditions

Similarly, one can understand that for all constraints which are not precisely met as equalities, i.e. for which $y_i((\mathbf{w} \cdot \mathbf{x}_i) + b) - 1 > 0$, the corresponding α_i must be 0: this is the value of α_i that maximizes L. The latter is the statement of the Karush-Kuhn-Tucker complementarity conditions of optimization theory (Karush, 1939; Kuhn and Tucker, 1951; Bertsekas, 1995).

The condition that at the saddle point, the derivatives of L with respect to the primal variables must vanish,

$$\frac{\partial}{\partial b} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0 \text{ and } \frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0, \tag{1.54}$$

leads to

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \tag{1.55}$$

and

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i. \tag{1.56}$$

support vector expansion

The solution vector thus has an expansion in terms of a subset of the training patterns, namely those patterns whose Lagrange multiplier α_i is non-zero. By the Karush-Kuhn-Tucker complementarity conditions these training patterns are the ones for which

$$\alpha_i(y_i((\mathbf{x}_i \cdot \mathbf{w}) + b) - 1) = 0, \quad i = 1, \ldots, m, \tag{1.57}$$

and therefore they correspond precisely to the *Support Vectors* (i.e. critical constraints) discussed in Section 1.1.4. Thus we have the satisfying result that the Support Vectors are the only training patterns that determine the optimal decision hyperplane; all other training patterns are irrelevant and do not appear in the expansion (1.56).

By substituting (1.55) and (1.56) into L, one eliminates the primal variables and arrives at the Wolfe dual of the optimization problem (e.g. Bertsekas, 1995): find multipliers α_i which

dual optimization problem

$$\text{maximize} \qquad W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \left(\mathbf{x}_i \cdot \mathbf{x}_j\right) \tag{1.58}$$

$$\text{subject to} \qquad \alpha_i \geq 0 \text{ for all } i = 1, \ldots, m, \text{ and } \sum_{i=1}^{m} \alpha_i y_i = 0. \tag{1.59}$$

The hyperplane decision function can thus be written as

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{m} y_i \alpha_i \left(\mathbf{x} \cdot \mathbf{x}_i\right) + b\right) \tag{1.60}$$

where b is computed using (1.57).

The structure of the optimization problem closely resembles those that typically arise in Lagrange's formulation of mechanics (e.g. Goldstein, 1986). In that case also, it is often only a subset of the constraints that are active. For instance, if we keep a ball in a box, then it will typically roll into one of the corners. The constraints corresponding to the walls which are not touched by the ball are irrelevant, the walls could just as well be removed.

Seen in this light, it is not too surprising that it is possible to give a mechanical interpretation of optimal margin hyperplanes (Burges and Schölkopf, 1997): If we assume that each support vector $\mathbf{x}_i$ exerts a perpendicular force of size α_i and sign y_i on a solid plane sheet lying along the hyperplane, then the solution satisfies the requirements of mechanical stability. The constraint (1.55) states that the forces on the sheet sum to zero; and (1.56) implies that the torques also sum to zero, via $\sum_i \mathbf{x}_i \times y_i\alpha_i\mathbf{w}/\|\mathbf{w}\| = \mathbf{w} \times \mathbf{w}/\|\mathbf{w}\| = 0$.

1.3.2 Feature Spaces and Kernels

To construct *Support Vector Machines*, the optimal hyperplane algorithm is augmented by a method for computing dot products in feature spaces that are *nonlinearly* related to input space (Aizerman et al., 1964; Boser et al., 1992). The basic idea is to map the data into some other dot product space (called the *feature space*) $\mathcal{F}$ via a nonlinear map

feature space

$$\Phi : \mathbb{R}^N \to \mathcal{F}, \tag{1.61}$$

and then in the space $\mathcal{F}$ perform the linear algorithm described above.

For instance, suppose we are given patterns $\mathbf{x} \in \mathbb{R}^N$ where most information is contained in the d-th order products (monomials) of entries x_j of $\mathbf{x}$, i.e. $x_{j_1}x_{j_2}\cdots x_{j_d}$, where $j_1,\ldots,j_d \in \{1,\ldots,N\}$. In that case, we might prefer to extract these monomial features first, and work in the feature space $\mathcal{F}$ of all products of d entries.

This approach, however, fails for realistically sized problems: for N-dimensional input patterns, there exist $(N+d-1)!/(d!(N-1)!)$ different monomials. Already 16×16 pixel input images (e.g. in character recognition) and a monomial degree $d = 5$ yield a dimensionality of 10^{10}.

This problem can be overcome by noticing that both the construction of the optimal hyperplane in $\mathcal{F}$ (cf. (1.58)) and the evaluation of the corresponding decision function (1.60) only require the evaluation of dot products $(\Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}'))$, and never require the mapped patterns $\Phi(\mathbf{x})$ in explicit form. This is crucial, since in some cases, the dot products can be evaluated by a simple kernel (Aizerman et al., 1964; Boser et al., 1992).

Mercer kernel

$$k(\mathbf{x},\mathbf{x}') = (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}')). \tag{1.62}$$

polynomial kernel

For instance, the polynomial kernel

$$k(\mathbf{x},\mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}')^d \tag{1.63}$$

can be shown to correspond to a map Φ into the space spanned by all products of exactly d dimensions of $\mathbb{R}^N$ (Poggio (1975); Boser et al. (1992)). For a proof, see Schölkopf (1997). For $d = 2$ and $\mathbf{x},\mathbf{x}' \in \mathbb{R}^2$, for example, we have (Vapnik, 1995b)

$$(\mathbf{x}\cdot\mathbf{x}')^2 = (x_1^2, x_2^2, \sqrt{2}\,x_1x_2)(y_1^2, y_2^2, \sqrt{2}\,y_1y_2)^\top = (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}')), \tag{1.64}$$

defining $\Phi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}\,x_1x_2)$.

By using $k(\mathbf{x},\mathbf{x}') = ((\mathbf{x}\cdot\mathbf{x}')+c)^d$ with $c > 0$, we can take into account all product of order up to d (i.e. including those of order smaller than d).

More generally, the following theorem of functional analysis shows that kernels k of positive integral operators give rise to maps Φ such that (1.62) holds (Mercer, 1909; Aizerman et al., 1964; Boser et al., 1992; Dunford and Schwartz, 1963):

Theorem 1.11 Mercer

positive integral operator

If k is a continuous symmetric kernel of a positive integral operator T, i.e.

$$(Tf)(\mathbf{x}') = \int_{\mathcal{X}} k(\mathbf{x}, \mathbf{x}') f(\mathbf{x})\, d\mathbf{x} \tag{1.65}$$

with

$$\int_{\mathcal{X}\times\mathcal{X}} k(\mathbf{x}, \mathbf{x}') f(\mathbf{x}) f(\mathbf{x}')\, d\mathbf{x}\, d\mathbf{x}' \geq 0 \tag{1.66}$$

for all $f \in L_2(\mathcal{X})$ ($\mathcal{X}$ being a compact subset of $\mathbb{R}^N$), it can be expanded in a uniformly convergent series (on $\mathcal{X} \times \mathcal{X}$) in terms of T's eigenfunctions ψ_j and positive eigenvalues λ_j,

$$k(\mathbf{x}, \mathbf{x}') = \sum_{j=1}^{N_{\mathcal{F}}} \lambda_j \psi_j(\mathbf{x}) \psi_j(\mathbf{x}'), \tag{1.67}$$

where $N_{\mathcal{F}} \leq \infty$ is the number of positive eigenvalues.

An equivalent way to characterize Mercer kernels is that they give rise to positive matrices $K_{ij} := k(\mathbf{x}_i, \mathbf{x}_j)$ for all $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ (Saitoh, 1988).

From (1.67), it is straightforward to construct a map Φ into a potentially infinite-dimensional l_2 space which satisfies (1.62). For instance, we may use

$$\Phi(\mathbf{x}) = (\sqrt{\lambda_1}\psi_1(\mathbf{x}), \sqrt{\lambda_2}\psi_2(\mathbf{x}), \ldots). \tag{1.68}$$

Rather than thinking of the feature space as an l_2 space, we can alternatively represent it as the Hilbert space $\mathcal{H}_k$ containing all linear combinations of the functions $f(.) = k(\mathbf{x}_i, .)$ ($\mathbf{x}_i \in \mathcal{X}$). To ensure that the map $\Phi : \mathcal{X} \to \mathcal{H}_k$, which in this case is defined as

$$\Phi(\mathbf{x}) = k(\mathbf{x}, .), \tag{1.69}$$

satisfies (1.62), we need to endow $\mathcal{H}_k$ with a suitable dot product $\langle ., . \rangle$. In view of the definition of Φ, this dot product needs to satisfy

$$\langle k(\mathbf{x}, .), k(\mathbf{x}', .) \rangle = k(\mathbf{x}, \mathbf{x}'), \tag{1.70}$$

reproducing kernel

which amounts to saying that k is a *reproducing kernel* for $\mathcal{H}_k$. For a Mercer kernel (1.67), such a dot product does exist. Since k is symmetric, the ψ_i ($i = 1, \ldots, N_{\mathcal{F}}$) can be chosen to be orthogonal with respect to the dot product in $L_2(C)$, i.e. $(\psi_j, \psi_n)_{L_2(C)} = \delta_{jn}$, using the Kronecker δ_{jn}. From this, we can construct $\langle ., . \rangle$ such that

$$\langle \sqrt{\lambda_j}\psi_j, \sqrt{\lambda_n}\psi_n \rangle = \delta_{jn}. \tag{1.71}$$

Substituting (1.67) into (1.70) then proves the desired equality (for further details, see Aronszajn (1950); Wahba (1973); Girosi (1998); Schölkopf (1997)).

sigmoid kernel

Besides (1.63), SV practictioners use sigmoid kernels

$$k(\mathbf{x}, \mathbf{x}') = \tanh(\kappa(\mathbf{x} \cdot \mathbf{x}') + \Theta) \tag{1.72}$$

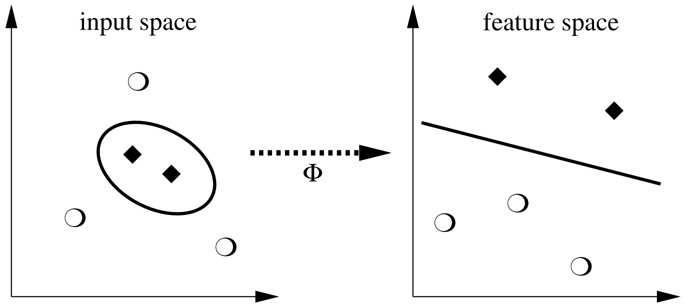

Figure 1.3 The idea of SV machines: map the training data nonlinearly into a higher-dimensional feature space via Φ, and construct a separating hyperplane with maximum margin there. This yields a nonlinear decision boundary in input space. By the use of a kernel function (1.62), it is possible to compute the separating hyperplane without explicitly carrying out the map into the feature space.

Gaussian RBF kernel

for suitable values of gain κ and threshold Θ, and radial basis function kernels, as for instance (Aizerman et al., 1964; Boser et al., 1992; Schölkopf et al., 1997)

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\|\mathbf{x} - \mathbf{x}'\|^2 / (2\,\sigma^2)\right), \tag{1.73}$$

with $\sigma > 0$. Note that when using Gaussian kernels, for instance, the feature space $\mathcal{H}_k$ thus contains all superpositions of Gaussians on $\mathcal{X}$ (plus limit points), whereas by definition of Φ (1.69), only single bumps $k(\mathbf{x}, .)$ do have pre-images under Φ.

The main lesson from the study of kernel functions, is that the use of kernels can turn any algorithm that only depends on dot products into a nonlinear algorithm which is linear in feature space. In the time since this was explicitly pointed out (Schölkopf et al., 1998c) a number of such algorithms have been proposed: until then the applications of the kernel trick were a proof of the convergence of rbf network training by (Aizerman et al., 1964) and the nonlinear variant of the SV algorithm by Boser et al. (1992) (see Figure 1.3). To construct SV machines, one computes an optimal hyperplane in feature space. To this end, we substitute $\Phi(\mathbf{x}_i)$ for each training example $\mathbf{x}_i$. The weight vector (cf. (1.56)) then becomes an expansion in feature space. Note that $\mathbf{w}$ will typically no more correspond to the image of just a single vector from input space (cf. Schölkopf et al. (1998a) for a formula to compute the pre-image if it exists), in other words, $\mathbf{w}$ may not be directly accessible any more. However, since all patterns only occur in dot products, one can substitute Mercer kernels k for the dot products (Boser et al., 1992; Guyon et al., 1993), leading to decision functions of the more general form (cf. (1.60))

decision function

$$g(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{m} y_i \alpha_i \left(\Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}_i)\right) + b\right) = \operatorname{sgn}\left(\sum_{i=1}^{m} y_i \alpha_i\, k(\mathbf{x}, \mathbf{x}_i) + b\right) \tag{1.74}$$

and the following quadratic program (cf. (1.58)):

$$\text{maximize} \qquad W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \tag{1.75}$$

$$\text{subject to} \qquad \alpha_i \geq 0, \quad i = 1, \ldots, m, \text{ and } \sum_{i=1}^{m} \alpha_i y_i = 0. \tag{1.76}$$

soft margin and kernels

Recall that, as discussed in Section 1.1.4 a separating hyperplane may not always exist, even in the expanded feature space $\mathcal{F}$. To cope with this difficulty, slack variables were introduced to yield the *soft margin* optimal hyperplane problem (1.25). Incorporating kernels, and rewriting (1.25) in terms of Lagrange multipliers, this again leads to the problem of maximizing (1.75), but now subject to the constraints

$$0 \leq \alpha_i \leq C, \quad i = 1, \ldots, m, \text{ and } \sum_{i=1}^{m} \alpha_i y_i = 0. \tag{1.77}$$

The only difference from the separable case (1.76) is the upper bound C on the Lagrange multipliers α_i. This way, the influence of the individual patterns (which could always be outliers) gets limited. As above, the solution takes the form (1.74). The threshold b can be computed by exploiting the fact that for all SVs $\mathbf{x}_i$ with $\alpha_i < C$, the slack variable ξ_i is zero (this again follows from the Karush-Kuhn-Tucker complementarity conditions), and hence

$$\sum_{j=1}^{m} y_j \alpha_j \, k(\mathbf{x}_i, \mathbf{x}_j) + b = y_i. \tag{1.78}$$

If one uses an optimizer that works with the double dual (e.g. Vanderbei, 1997), one can also recover the value of the primal variable b directly from the corresponding double dual variable.

Finally, the algorithm can be modified such that it does not require the regularization constant C. Instead, one specifies an upper bound $0 \leq \nu \leq 1$ on the fraction of points allowed to lie in the margin (asymptotically, the number of SVs) (Schölkopf et al., 1998d). This leaves us with a homogeneous target function made up by the quadratic part of (1.75), and an additional lower bound constraint on the sum over all Lagrange multipliers.

1.3.3 Smoothness and Regularization

For kernel-based function expansions, one can show (Smola and Schölkopf, 1998b) that given a regularization operator P mapping the functions of the learning machine into some dot product space, the problem of minimizing the regularized risk

$$R_{\text{reg}}(f) := R_{\text{emp}}(f) + \frac{\lambda}{2} \|Pf\|^2 \tag{1.79}$$

regularized risk

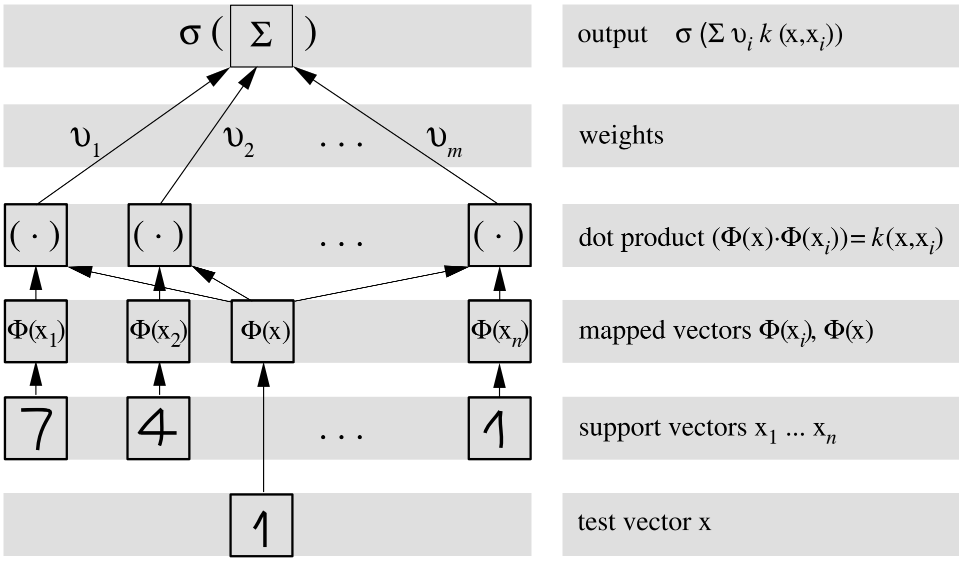

Figure 1.4 Architecture of SV machines. The input $\mathbf{x}$ and the Support Vectors $\mathbf{x}_i$ are nonlinearly mapped (by Φ) into a feature space $\mathcal{F}$, where dot products are computed. By the use of the kernel k, these two layers are in practice computed in one single step. The results are linearly combined by weights υ_i, found by solving a quadratic program (in pattern recognition, $\upsilon_i = y_i\alpha_i$; in regression estimation, $\upsilon_i = \alpha_i^* - \alpha_i$). The linear combination is fed into the function σ (in pattern recognition, $\sigma(x) = \operatorname{sgn}(x + b)$; in regression estimation, $\sigma(x) = x + b$).

(with a regularization parameter $\lambda \geq 0$) can be written as a constrained optimization problem. For particular choices of the loss function, it further reduces to a SV type quadratic programming problem. The latter thus is not specific to SV machines, but is common to a much wider class of approaches. What gets lost in the general case, however, is the fact that the solution can usually be expressed in terms of a small number of SVs (cf. also Girosi (1998), who establishes a connection between SV machines and basis pursuit denoising (Chen et al., 1995)). This specific feature of SV machines is due to the fact that the type of regularization and the class of functions that the estimate is chosen from are intimately related (Girosi et al., 1993; Smola and Schölkopf, 1998a; Smola et al., 1998): the SV algorithm is equivalent to minimizing the regularized risk $R_{\text{reg}}(f)$ on the set of functions

$$f(\mathbf{x}) = \sum_i \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b, \tag{1.80}$$

provided that k and P are interrelated by

$$k(\mathbf{x}_i, \mathbf{x}_j) = ((Pk)(\mathbf{x}_i, .) \cdot (Pk)(\mathbf{x}_j, .)) \,. \tag{1.81}$$

To this end, k is chosen as a Green's function of P^*P, for in that case, the right hand side of (1.81) equals $(k(\mathbf{x}_i, .) \cdot (P^*Pk)(\mathbf{x}_j, .)) = (k(\mathbf{x}_i, .) \cdot \delta_{\mathbf{x}_j}(.)) = k(\mathbf{x}_i, \mathbf{x}_j)$. For instance, an RBF kernel corresponds to regularization with a functional containing a specific differential operator.

In SV machines, the kernel thus plays a dual role: firstly, it determines the class of functions (1.80) that the solution is taken from; secondly, via (1.81), the kernel determines the type of regularization that is used. The next question, naturally, is what type of regularization (i.e. kernel) we should use in order to get the best generalization performance. Using bounds on covering numbers of Hilbert spaces (Carl and Stephani, 1990), one can show (Williamson et al., 1998b,a; Schölkopf et al., 1999) that the eigenspectrum of the matrix $k(x_i, x_j)$ is closely connected to the latter and also to the eigenspectrum of the kernel k.

regularization networks

For arbitrary expansions of f into basis functions, say f_i, the considerations about smoothness of the estimate still hold, provided $\|Pf\|$ is a norm in the space spanned by the basis functions f_i (otherwise one could find functions $f \in \operatorname{span}\{f_i\}$ with $\|Pf\| = 0$, however $f \neq 0$). In this case the existing bounds for kernel expansions can be readily applied to regularization networks as well (cf. e.g. (Williamson et al., 1998b; Smola, 1998) for details). However, one can show (Kimeldorf and Wahba, 1971; Cox and O'Sullivan, 1990), that such an expansion may not fully minimize the regularized risk functional (1.79). This is one of the reasons why often only kernel expansions are considered.

Gaussian processes

Finally it is worth while pointing out the connection between Gaussian Processes and Support Vector machines. The similarity is most obvious in regression, where the Support Vector solution is the maximum a posteriori estimate of the corresponding Bayesian inference scheme (Williams, 1998). In particular, the kernel k of Support Vector machines plays the role of a covariance function such that the prior probability of a function $f = \sum_i \alpha_i k(\mathbf{x}_i, \mathbf{x})$ is given by

$$P(f) \propto \exp\left(-\tfrac{1}{2}\|Pf\|^2\right) = \exp\left(-\tfrac{1}{2}\sum_{i,j} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j)\right). \tag{1.82}$$

Bayesian methods, however, require averaging over the posterior distribution $P(f|X, Y)$ in order to obtain the final estimate and to derive error bounds. In classification the situation is even more complicated, since we have Bernoulli distributed random variables for the labels of the classifier. See (Williams, 1998) for more details on this subject.

1.3.4 A Bound on the Leave-One-Out Estimate

Besides the bounds directly involving large margins, which are useful for stating uniform convergence results, one may also try to estimate $R(f)$ by using leave-one-out estimates. Denote by f_i the estimate obtained from $X\backslash\{\mathbf{x}_i\}, Y\backslash\{y_i\}$. Then

$$R_{\text{out}}(f) := \frac{1}{m}\sum_{i=1}^{m} c(\mathbf{x}_i, y_i, f_i(\mathbf{x}_i)) \tag{1.83}$$

One can show (cf. e.g. (Vapnik, 1979)) that the latter is an unbiased estimator of $R(f)$. Unfortunately, $R_{\text{out}}(f)$ is hard to compute and thus rarely used. In the case of Support Vector classification, however, an upper bound on $R_{\text{out}}(f)$ is not too difficult to obtain. Vapnik (1995b) showed that the fraction of Support Vectors is an upper bound on $R_{\text{out}}(f)$. Jaakkola and Haussler (1999) have generalized this result as follows

$$\begin{aligned} R_{\text{out}}(f) &\leq \frac{1}{m}\sum_{i=1}^{m} 1_{\left\{y_i \sum_{j\neq i} \alpha_j y_j k(\mathbf{x}_j, \mathbf{x}_i) + y_i b > 0\right\}} \\ &= \frac{1}{m}\sum_{i=1}^{m} 1_{\{y_i f(\mathbf{x}_i) - \alpha_i k(\mathbf{x}_i, \mathbf{x}_i)) > 0\}}. \end{aligned} \tag{1.84}$$

The latter can be obtained easily without explicitly solving the optimization problem again for the reduced samples. In particular, for kernels with $k(\mathbf{x}, \mathbf{x}) = 1$ like many RBF kernels the condition reduces to testing whether $y_i f(\mathbf{x}_i) - \alpha_i > 0$. The remaining problem is that $R_{\text{out}}(f)$ itself is a random variable and thus it does not immediately give a *bound* on $R(f)$. See also chapters 6 and 4 for futher details on how to exploit these bounds in practical cases.

1.4 Boosting

Freund and Schapire (1995) proposed the AdaBoost algorithm for combining classifiers produced by other learning algorithms. AdaBoost has been very successful in practical applications (see Section 1.5). It turns out that it is also a large margin technique.

Table 1.2 gives the pseudocode for the algorithm. It returns a convex combination of classifiers from a class G, by using a learning algorithm L that takes as input a training sample X, Y and a distribution D on X (not to be confused with the true distribution p), and returns a classifier from G. The algorithm L aims to minimize training error on X, Y, weighted according to D. That is, it aims to minimize

$$\sum_{i=1}^{m} D_i 1_{\{h(\mathbf{x}_i)\neq y_i\}}. \tag{1.85}$$

AdaBoost iteratively combines the classifiers returned by L. The idea behind AdaBoost is to start with a uniform weighting over the training sample, and progressively adjust the weights to emphasize the examples that have been frequently misclassified by the classifiers returned by L. These classifiers are combined with convex coefficients that depend on their respective weighted errors. The following theorem shows that Adaboost produces a large margin classifier, provided L is suc-

argument: `Training sample,` $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subset \mathcal{X}$, $Y = \{y_1, \ldots, y_m\} \subset \{\pm 1\}$
`Number of iterations,` T
returns: `Convex combination of functions from` G, $f = \sum_{t=1}^T \alpha_t g_t$.
function `AdaBoost`(X, Y, T)
  for all i `from` $i = 1, \ldots, m$
    $D_1(i) := 1/m$
  endfor
  for all t `from` $\{1, \ldots, T\}$
    $g_t := L(X, Y, D_t)$

$$\varepsilon_t := \sum_{i=1}^m D_t(i) 1_{g_t(x_i) \neq y_i}$$

$$\alpha_t := \frac{1}{2} \ln\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right)$$

$$Z_t := 2\sqrt{\varepsilon_t(1 - \varepsilon_t)}$$

    for all i `from` $i = 1, \ldots, m$

$$D_{t+1}(i) := \begin{cases} D_t(i) e^{-\alpha_t}/Z_t & \texttt{if } y_i = g_t(x_i) \\ D_t(i) e^{\alpha_t}/Z_t & \texttt{otherwise,} \end{cases}$$

    endfor
  endfor
  return $f = \dfrac{\sum_{t=1}^T \alpha_t g_t}{\sum_{i=1}^T \alpha_t}$.
end

Table 1.2 Pseudocode for the `Adaboost` algorithm. (L is a learning algorithm that chooses a classifier from G to minimize weighted training error.)

cessful at finding classifiers with small weighted training error. See (Schapire et al., 1998). Recall (1.30) that the margin error of a function f with respect to ρ on a sample X, Y is $R_\rho(f) = \frac{1}{m} \sum_{i=1}^m 1_{\{y_i f(\mathbf{x}_i) < \rho\}}$.

Theorem 1.12 Margin Error of AdaBoost
If, at iteration t, L returns a function with weighted training error $\varepsilon_t < 1/2$, then `AdaBoost` returns a function f that satisfies

$$R_\rho(f) \leq 2^T \prod_{t=1}^T \sqrt{\varepsilon_t^{1-\rho}(1 - \varepsilon_t)^{1+\rho}}. \tag{1.86}$$

In particular, if $\varepsilon_t \leq 1/2 - 2\rho$, then

$$R_\rho(f) < (1 - \rho^2)^{T/2}, \tag{1.87}$$

and this is less than ε for $T \geq (2/\rho^2) \ln(1/\varepsilon)$.

1.5 Empirical Results, Implementations, and Further Developments

Large margin classifiers are not only promising from the theoretical point of view. They also have proven to be competitive or superior to other learning algorithms in practical applications. In the following we will give references to such situations.

1.5.1 Boosting

Experimental results show that boosting is able to improve the performance of classifiers significantly. Extensive studies on the UC Irvine dataset, carried out by Freund and Schapire (1996) and Quinlan (1996) with tree classifiers show the performance of such methods. However, also other learning algorithms can benefit from boosting. Schwenk and Bengio (1998) achieve record performance on an OCR task on the UC Irvine database, using neural networks as the base classifiers. See Rätsch (1998) and chapter 7 for further results on the performance of improved versions of boosted classifiers.

1.5.2 Support Vector Machines

SV Machines perform particularly well in feature rich highdimensional problems. Schölkopf et al. (1995); Schölkopf et al. (1996, 1998b); Burges and Schölkopf (1997); Schölkopf (1997) achieve state of the art, or even record performance in several Optical Character Recognition (OCR) tasks such as the digit databases of the United Postal Service (USPS) and the National Institute of Standards and Technology (NIST). The latter can be obtained at

`http://www.research.att.com/~yann/ocr/mnist/`

Similar results have been obtained for face recognition by Oren et al. (1997); Osuna et al. (1997b) and object recognition (Blanz et al., 1996; Schölkopf, 1997). Finally, also on large noisy problems SV Machines are very competitive as shown in (Smola, 1998).

1.5.3 Implementation and Available Code

Whilst Boosting can be easily implemented by combining a base learner and following the pseudocode of table 1.2. Hence one only has to provide a base learning algorithm satisfying the properties of a weak learner, which defers all problems to the underlying algorithm.

`http://www.research.att.com/~yoav/adaboost/`

provides a Java applet demonstrating the basic properties of AdaBoost.

The central problem in Support Vector Machines is a quadratic programming problem. Unfortunately, off-the-shelf packages developed in the context of mathematical programming like MINOS (Murtagh and Saunders, 1993), LOQO (Vanderbei, 1994), OSL (IBM Corporation, 1992), or CPLEX (CPL, 1994) are often prohibitively expensive or unsuitable for optimization problems in more than several thousand variables (whilst the number of variables may be in the tens of thousands in practical applications). Furthermore these programs are often optimized to deal with sparse matrix entries, causing unneeded overhead when solving generic SV optimization problems (which are sparse in the solution, not in the matrix entries).

This situation led to the development of several quadratic optimization algorithms specifically designed to suit the needs of SV machines. Starting from simple subset selection algorithms as initially described by Vapnik (1979) and subsequently implemented in e.g. (Schölkopf et al., 1995), more advanced chunking methods were proposed (Osuna et al., 1997a) (see also (Joachims, 1999) for a detailed description of the algorithm) for splitting up the optimization problem into smaller subproblems that could be easily solved by standard optimization code. Other methods exploit constrained gradient descent techniques (Kaufmann, 1999), or minimize very small subproblems, such as the Sequential Minimal Optimization algorithm (SMO) by Platt (1999). See also chapter 5 for further methods for training a SV classifier. Implementations include SvmLight by Joachims (1999),

`http://www-ai.cs.uni-dortmund.de/thorsten/svm_light.html`

the Royal Holloway / ATT / GMD Support Vector Machine by Saunders et al. (1998), available at

`http://svm.dcs.rhbnc.ac.uk/`

and the implementation by Steve Gunn which can be downloaded from

`http://www.isis.ecs.soton.ac.uk/resources/svminfo/`.

The first two of these optimizers use the GMD (Smola) implementation of an interior point code along the lines of Vanderbei (1994) as the core optimization engine. It is available as a standalone package at

`http://www.svm.first.gmd.de/software.html`.

This site will also contain pointers to further toolboxes as they become available. Java applets for demonstration purposes can be found at

`http://http://svm.dcs.rhbnc.ac.uk/pagesnew/GPat.shtml`
`http://http://svm.research.bell-labs.com/SVT/SVMsvt.html.`

1.6 Notation

We conclude the introduction with a list of symbols which are used throughout the book, unless stated otherwise.

$\mathbb{N}$	the set of natural numbers
$\mathbb{R}$	the set of reals
X	a sample of input patterns
Y	a sample of output labels
$\mathcal{X}$	an abstract domain
ln	logarithm to base e
$\log_2$	logarithm to base 2
$(\mathbf{x} \cdot \mathbf{x}')$	inner product between vectors $\mathbf{x}$ and $\mathbf{x}'$
$\lVert.\rVert$	2-norm (Euclidean distance), $\lVert\mathbf{x}\rVert := \sqrt{(\mathbf{x} \cdot \mathbf{x})}$
$\lVert.\rVert_p$	p-norm , $\lVert\mathbf{x}\rVert_p := \left(\sum_{i=1}^N \lvert x_i\rvert^p\right)^{1/p}$
$\lVert.\rVert_\infty$	∞-norm , $\lVert\mathbf{x}\rVert_\infty := \max_{i=1}^N \lvert x_i\rvert$
ℓ_p	ℓ_p metric
$L_2(X)$	space of functions on X square integrable wrt. Borel–Lebesgue measure
$\mathbf{E}(\xi)$	expectation of random variable ξ
$\Pr(\cdot)$	probability of an event
N	dimensionality of input space
m	number of training examples
$\mathbf{x}_i$	input patterns
y_i	target values, or (in pattern recognition) classes
$\mathbf{w}$	weight vector
b	constant offset (or threshold)
h	VC dimension
f	a real valued function $f : \mathbb{R}^N \to \mathbb{R}$ (unthresholded)
F	a family of real valued functions f
g	a decision function $g : \mathbb{R}^N \to \{-1, 1\}$
F	a family of decision functions g
$\rho_f(\mathbf{x}, y)$	margin of function f on the example $(\mathbf{x}, y)$, i.e. $y\, f(\mathbf{x})$
ρ_f	minimum margin, i.e. $\min_{1 \le i \le m} \rho_f(\mathbf{x}_i, y_i)$

$c(\mathbf{x}, y, f(\mathbf{x}))$	cost function
$R(g)$	risk of g, i.e. expected fraction of errors
$R_{\text{emp}}(g)$	empirical risk of g, i.e. fraction of training errors
$R(f)$	risk of f
$R_{\text{emp}}(f)$	empirical risk of f
k	Mercer kernel
$\mathcal{F}$	Feature space induced by a kernel
$\mathbf{\Phi}$	map into feature space (induced by k)
α_i	Lagrange multiplier
$\boldsymbol{\alpha}$	vector of all Lagrange multipliers
ξ_i	slack variables
$\boldsymbol{\xi}$	vector of all slack variables
C	regularization constant for SV Machines
λ	regularization constant ($C = \frac{1}{\lambda}$)

References

F. Aidu and V. Vapnik. Estimation of probability density on the basis of the method of stochastic regularization. *Avtomatika i Telemekhanika*, 4:84–97, 1989.

M. Aizerman, E. Braverman, and L. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821 – 837, 1964.

K. S. Alexander. Probability inequalities for empirical processes and a law of the iterated logarithm. *Annals of Probability*, 12:1041–1067, 1984.

N. Alon, S. Ben-David, N. Cesa-Bianchi, and D. Haussler. Scale–sensitive Dimensions, Uniform Convergence, and Learnability. *Journal of the ACM*, 44(4):615–631, 1997.

M. Anthony and P. Bartlett. *Neural Network Learning: Theoretical Foundations.* Cambridge University Press, 1999. (to appear).

N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68:337 – 404, 1950.

P. Bartlett and J. Shawe-Taylor. Generalization performance of support vector machines and other pattern classifiers. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 43–54, Cambridge, MA, 1999. MIT Press.

P. L. Bartlett. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, 44(2):525–536, 1998.

P. L. Bartlett, S. R. Kulkarni, and S. E. Posner. Covering numbers for real-valued function classes. *IEEE Transactions on Information Theory*, 43(5):1721–1724, 1997.

K. P. Bennett and J. A. Blue. A support vector machine approach to decision trees. In *Proceedings of IJCNN'98*, pages 2396–2401, Anchorage, Alaska, 1997.

K. P. Bennett, D. Hui, and L. Auslender. On support vector decision trees for database marketing. Department of Mathematical Sciences Math Report No. 98-100, Rensselaer Polytechnic Institute, Troy, NY 12180, 1998. http://www.math.rpi.edu/~bennek/.

K. P. Bennett and O. L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1:23–34, 1992.

D. P. Bertsekas. *Nonlinear Programming.* Athena Scientific, Belmont, MA, 1995.

V. Blanz, B. Schölkopf, H. Bülthoff, C. Burges, V. Vapnik, and T. Vetter. Comparison of view-based object recognition algorithms using realistic 3D models. In C. von der Malsburg, W. von Seelen, J. C. Vorbrüggen, and B. Sendhoff, editors, *Artificial Neural Networks — ICANN'96*, pages 251 – 256, Berlin, 1996. Springer Lecture Notes in Computer Science, Vol. 1112.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, PA, 1992. ACM Press.

L. Bottou and V. N. Vapnik. Local learning algorithms. *Neural Computation*, 4(6): 888–900, 1992.

P. S. Bradley and O. L. Mangasarian. Feature selection via concave minimization and support vector machines. In J. Shavlik, editor, *Machine Learning Proceedings of the Fifteenth International Conference(ICML '98)*, pages 82–90, San Francisco, CA, 1998. Morgan Kaufmann. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/98-03.ps.Z.

E. J. Bredensteiner. *Optimization Methods in Data Mining and Machine Learning.* PhD thesis, Department of Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, NY, 1997.

E. J. Bredensteiner and K. P. Bennett. Feature minimization within decision trees. *Computational Optimization and Applications*, 10:110–126, 1997.

C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

C. J. C. Burges and B. Schölkopf. Improving the accuracy and speed of support vector learning machines. In M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 375–381, Cambridge, MA, 1997. MIT Press.

B. Carl and I. Stephani. *Entropy, compactness, and the approximation of operators.* Cambridge University Press, Cambridge, UK, 1990.

S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. Technical Report 479, Department of Statistics, Stanford University, 1995.

V. Cherkassky and F. Mulier. *Learning from Data — Concepts, Theory and Methods.* John Wiley & Sons, New York, 1998.

C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273 – 297, 1995.

R. Courant and D. Hilbert. *Methods of Mathematical Physics*, volume 1. Interscience Publishers, Inc, New York, 1953.

D. Cox and F. O'Sullivan. Asymptotic analysis of penalized likelihood and related estimators. *Ann. Statist.*, 18:1676–1695, 1990.

CPLEX Optimization Incorporated, Incline Village, Nevada. *Using the CPLEX Callable Library*, 1994.

Luc Devroye and Gábor Lugosi. Lower bounds in pattern recognition and learning. *Pattern Recognition*, 28, 1995.

S. P. Dirkse and M. C. Ferris. The PATH solver: A non-monotone stabilization scheme for mixed complementarity problems. *Optimization Methods and Software*, 5:123–156, 1995. ftp://ftp.cs.wisc.edu/tech-reports/reports/93/tr1179.ps.

R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* John Wiley & Sons, 1973.

N. Dunford and J. T. Schwartz. *Linear Operators Part II: Spectral Theory, Self Adjoint Operators in Hilbert Space.* Number VII in Pure and Applied Mathematics. John Wiley & Sons, New York, 1963.

A. Ehrenfeucht, D. Haussler, M. Kearns, and L. Valiant. A general lower bound on the number of examples needed for learning. *Information and Computation*, 82: 247–261, 1989.

M. C. Ferris and T. S. Munson. Interfaces to PATH 3.0: Design, implementation and usage. *Computational Optimization and Applications*, 12:207–227, 1999. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/97-12.ps.

M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3:95–110, 1956.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Computational Learning Theory: Eurocolt '95*, pages 23–37. Springer-Verlag, 1995.

Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proc. 13th International Conference on Machine Learning*, pages 148–146. Morgan Kaufmann, 1996.

F. Girosi. An equivalence between sparse approximation and support vector machines. *Neural Computation*, 10(6):1455–1480, 1998.

F. Girosi, M. Jones, and T. Poggio. Priors, stabilizers and basis functions: From regularization to radial, tensor and additive splines. A.I. Memo No. 1430, MIT, 1993.

H. Goldstein. *Classical Mechanics.* Addison-Wesley, Reading, MA, 1986.

L. Gurvits. A note on a scale-sensitive dimension of linear bounded functionals in banach spaces. In *Proceedings of Algorithm Learning Theory, ALT-97*, 1997. Also: NECI Technical Report, 1997.

I. Guyon, B. Boser, and V. Vapnik. Automatic capacity tuning of very large VC-dimension classifiers. In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems*, volume 5, pages 147–155. Morgan Kaufmann, San Mateo, CA, 1993.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

IBM Corporation. IBM optimization subroutine library guide and reference. *IBM Systems Journal*, 31, 1992. SC23-0519.

T. S. Jaakkola and D. Haussler. Probabilistic kernel regression models. In *Proceedings of the 1999 Conference on AI and Statistics*, 1999.

T. Joachims. Making large–scale SVM learning practical. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 169–184, Cambridge, MA, 1999. MIT Press.

W. Karush. Minima of functions of several variables with inequalities as side constraints. Master's thesis, Dept. of Mathematics, Univ. of Chicago, 1939.

L. Kaufmann. Solving the quadratic programming problem arising in support vector classification. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 147–168, Cambridge, MA, 1999. MIT Press.

G. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *J. Math. Anal. Applic.*, 33:82–95, 1971.

H. W. Kuhn and A. W. Tucker. Nonlinear programming. In *Proc.* 2^{nd} *Berkeley Symposium on Mathematical Statistics and Probabilistics*, pages 481–492, Berkeley, 1951. University of California Press.

Philip M. Long. The complexity of learning according to two models of a drifting environment. In *Proceedings of the 11th Annual Conference on Computational Learning Theory*, pages 116–125. ACM Press, 1998.

D. G. Luenberger. *Introduction to Linear and Nonlinear Programming*. Addison-Wesley, Reading, MA, 1973.

O. L. Mangasarian. Multi-surface method of pattern separation. *IEEE Transactions on Information Theory*, IT-14:801–807, 1968.

O. L. Mangasarian. *Nonlinear Programming*. SIAM, Philadelphia, PA, 1994.

O. L. Mangasarian. Mathematical programming in data mining. *Data Mining and Knowledge Discovery*, 42(1):183–201, 1997.

O. L. Mangasiarian. Linear and nonlinear separation of patterns by linear programming. *Operations Research*, 13:444–452, 1965.

MATLAB. *User's Guide*. The MathWorks, Inc., Natick, MA 01760, 1992.

J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philos. Trans. Roy. Soc. London*, A 209:415–446, 1909.

B. A. Murtagh and M. A. Saunders. MINOS 5.4 user's guide. Technical Report SOL 83.20, Stanford University, 1993.

M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio. Pedestrian detection using wavelet templates. In *Proc. Computer Vision and Pattern Recognition*, pages 193–199, Puerto Rico, 1997.

E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. In J. Principe, L. Gile, N. Morgan, and E. Wilson, editors, *Neural Networks for Signal Processing VII — Proceedings of the 1997 IEEE Workshop*, pages 276 – 285, New York, 1997a. IEEE.

E. Osuna, R. Freund, and F. Girosi. Training support vector machines: An application to face detection. In *Proc. Computer Vision and Pattern Recognition '97*, pages 130–136, 1997b.

J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 185–208, Cambridge, MA, 1999. MIT Press.

T. Poggio. On optimal nonlinear associative recall. *Biological Cybernetics*, 19: 201–209, 1975.

J. R. Quinlan. Bagging, boosting, and C4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, pages 725–730, Menlo Park, 1996. AAAI Press / MIT Press.

G. Rätsch. Ensemble learning for classification. Master's thesis, University of Potsdam, 1998. in German.

S. Saitoh. *Theory of Reproducing Kernels and its Applications.* Longman Scientific & Technical, Harlow, England, 1988.

C. Saunders, M. O. Stitson, J. Weston, L. Bottou, B. Schölkopf, and A. Smola. Support vector machine - reference manual. Technical Report CSD-TR-98-03, Department of Computer Science, Royal Holloway, University of London, Egham, TW20 0EX, UK, 1998. TR available as http://www.dcs.rhbnc.ac.uk/research/compint/areas/comp_learn/sv/pub/report98-03.ps; SVM available at http://svm.dcs.rhbnc.ac.uk/.

R. Schapire, Y. Freund, P. Bartlett, and W. Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics*, 1998. (To appear. An earlier version appeared in: D.H. Fisher, Jr. (ed.), Proceedings ICML97, Morgan Kaufmann.).

B. Schölkopf, C. Burges, and V. Vapnik. Extracting support data for a given task. In U. M. Fayyad and R. Uthurusamy, editors, *Proceedings, First International Conference on Knowledge Discovery & Data Mining.* AAAI Press, Menlo Park, CA, 1995.

B. Schölkopf. *Support Vector Learning.* R. Oldenbourg Verlag, Munich, 1997.

B. Schölkopf, S. Mika, A. Smola, G. Rätsch, and K.-R. Müller. Kernel PCA pattern reconstruction *via* approximate pre-images. In L. Niklasson, M. Bodén, and T. Ziemke, editors, *Proceedings of the 8th International Conference on Artificial Neural Networks*, Perspectives in Neural Computing, pages 147 – 152, Berlin, 1998a. Springer Verlag.

B. Schölkopf, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Generalization bounds via eigenvalues of the Gram matrix. Submitted to COLT99, 1999.

B. Schölkopf, P. Simard, A. Smola, and V. Vapnik. Prior knowledge in support vector kernels. In M. Jordan, M. Kearns, and S. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 640 – 646, Cambridge, MA, 1998b. MIT Press.

B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998c.

B. Schölkopf, A. Smola, R. Williamson, and P. Bartlett. New support vector algorithms. NeuroCOLT Technical Report NC-TR-98-031, Royal Holloway College, University of London, UK, 1998d.

B. Schölkopf, K. Sung, C. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik. Comparing support vector machines with gaussian kernels to radial basis function classifiers. A.I. Memo No. 1599, Massachusetts Institute of Techology, 1996.

B. Schölkopf, K. Sung, C. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik. Comparing support vector machines with gaussian kernels to radial basis function classifiers. *IEEE Trans. Sign. Processing*, 45:2758 – 2765, 1997.

H. Schwenk and Y. Bengio. Training methods for adaptive boosting of neural networks. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. The MIT Press, 1998.

J. Shawe-Taylor, P. L. Bartlett, R. C. Williamson, and M. Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE Transactions on Information Theory*, 1998. To appear. Also: NeuroCOLT Technical Report NC-TR-96-053, 1996, ftp://ftp.dcs.rhbnc.ac.uk/pub/neurocolt/tech_reports.

Hans Ulrich Simon. General bounds on the number of examples needed for learning probabilistic concepts. *J. of Comput. Syst. Sci.*, 52(2):239–254, 1996. Earlier version in 6th COLT, 1993.

A. Smola and B. Schölkopf. From regularization operators to support vector kernels. In M. Jordan, M. Kearns, and S. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 343 – 349, Cambridge, MA, 1998a. MIT Press.

A. Smola and B. Schölkopf. On a kernel-based method for pattern recognition, regression, approximation and operator inversion. *Algorithmica*, 22:211 – 231, 1998b.

A. Smola, B. Schölkopf, and K.-R. Müller. General cost functions for support vector regression. In T. Downs, M. Frean, and M. Gallagher, editors, *Proc. of the Ninth Australian Conf. on Neural Networks*, pages 79 – 83, Brisbane, Australia, 1998. University of Queensland.

A. J. Smola. *Learning with Kernels*. PhD thesis, Technische Universität Berlin, 1998.

M. Talagrand. Sharper bounds for gaussian and empirical processes. *Annals of Probability*, 22:28–76, 1994.

R. Vanderbei. LOQO: An interior point code for quadratic programming. Technical Report SOR 94-15, Princeton University, 1994.

R. J. Vanderbei. LOQO user's manual – version 3.10. Technical Report SOR-97-08, Princeton University, Statistics and Operations Research, 1997. Code available at http://www.princeton.edu/˜rvdb/.

V. Vapnik. *Estimation of Dependences Based on Empirical Data [in Russian].* Nauka, Moscow, 1979. (English translation: Springer Verlag, New York, 1982).

V. Vapnik. Inductive principles of statistics and learning theory. In P. Smolensky, M. C. Mozer, and D. E. Rumelhart, editors, *Mathematical Perspectives on Neural Networks.* Lawrence Erlbaum, Mahwah, NJ, 1995a.

V. Vapnik. *The Nature of Statistical Learning Theory.* Springer Verlag, New York, 1995b.

V. Vapnik. Structure of statistical learning theory. In A. Gammerman, editor, *Computational and Probabalistic Reasoning*, chapter 1. Wiley, Chichester, 1996.

V. Vapnik. *Statistical Learning Theory.* Wiley, New York, 1998. forthcoming.

V. Vapnik and A. Chervonenkis. A note on one class of perceptrons. *Automation and Remote Control*, 25, 1964.

V. Vapnik and A. Chervonenkis. Uniform convergence of frequencies of occurence of events to their probabilities. *Dokl. Akad. Nauk SSSR*, 181:915 – 918, 1968.

V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2): 264–280, 1971.

V. Vapnik and A. Chervonenkis. *Theory of Pattern Recognition [in Russian].* Nauka, Moscow, 1974. (German Translation: W. Wapnik & A. Tscherwonenkis, *Theorie der Zeichenerkennung*, Akademie-Verlag, Berlin, 1979).

V. Vapnik and A. Chervonenkis. Necessary and sufficient conditions for the uniform convergence of means to their expectations. *Theory of Probability and its Applications*, 26(3):532–553, 1981.

V. Vapnik and A. Chervonenkis. The necessary and sufficient conditions for consistency in the empirical risk minimization method. *Pattern Recognition and Image Analysis*, 1(3):283–305, 1991.

V. Vapnik, S. Golowich, and A. Smola. Support vector method for function approximation, regression estimation, and signal processing. In M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 281–287, Cambridge, MA, 1997. MIT Press.

V. Vapnik and A. Lerner. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24, 1963.

V. Vapnik, E. Levin, and Y. Le Cun. Measuring the VC-dimension of a learning machine. *Neural Computation*, 6(5):851–876, 1994.

G. Wahba. Convergence rates of certain approximate solutions to Fredholm integral equations of the first kind. *Journal of Approximation Theory*, 7:167 – 185, 1973.

G. Wahba. Support vector machines, reproducing kernel hilbert spaces and the randomized GACV. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 69–88, Cambridge, MA, 1999. MIT Press.

C. K. I. Williams. Prediction with gaussian processes: From linear regression to linear prediction and beyond. In M. I. Jordan, editor, *Learning and Inference in Graphical Models*. Kluwer, 1998. To appear. Also: Technical Report NCRG/97/012, Aston University.

R. Williamson, A. Smola, and B. Schölkopf. Entropy numbers, operators and support vector kernels. In *Advances in Neural Information Processing Systems 11*, 1998a. Submitted.

R. C. Williamson, A. J. Smola, and B. Schölkopf. Generalization performance of regularization networks and support vector machines via entropy numbers of compact operators. NeuroCOLT Technical Report NC-TR-98-019, Royal Holloway College, University of London, UK, 1998b.

P. Wolfe. A duality theorem for nonlinear programming. *Quarterly of Applied Mathematics*, 19:239–244, 1961.

Index

decision function, 20
  Bayes optimal, 2
  linear, 1
dimension
  fat shattering, 12
  VC, 10

error
  leave one out, 23
  margin, 9

feature
  space, 17

generalized support vector machine, 36
Green's function, 22
GSVM, 36
  linear programming, 39
  quadratic programming, 37

hyperplane
  canonical, 6
  optimal, 5
  soft margin, 8

Karush-Kuhn-Tucker conditions, 16
kernel, 17
  generalized, 35
  Mercer, 18
  polynomial, 17
  radial basis function, 19
  reproducing, 19
  sigmoid, 19

Lagrange multipliers, 15
Lagrangian, 15
linear discriminant, 3

margin, 4
  Euclidean, 6

notation, 27

Perceptron, 3
programming problem
  dual, 16, 20
  primal, 8, 15

regularization, 22
risk
  empirical, 9
  expected, 9
  regularized, 21

separating surface
  nonlinear, 35
slack variables, 8, 28
SRM, *see* structural risk minimization
support vector, 8, 16
  expansion, 16, 21
  mechanical interpretation, 17
  pattern recognition, 19
support vector machine
  generalized, 36
symbols, 27

threshold, 20