that the small training error does not guarantee a small test error. This illustrates how the bound can apply independent of assumptions about the underlying distribution $P(x, y)$: it always holds (provided that $h < m$), but it does not always make a nontrivial prediction. In order to get nontrivial predictions from (1.19), the function class must be *restricted* such that its capacity (e.g., VC dimension) is small enough (in relation to the available amount of data). At the same time, the class should be large enough to provide functions that are able to model the dependencies hidden in $P(x, y)$. The choice of the set of functions is thus crucial for learning from data. In the next section, we take a closer look at a class of functions which is particularly interesting for pattern recognition problems.

1.4 Hyperplane Classifiers

In the present section, we shall describe a hyperplane learning algorithm that can be performed in a dot product space (such as the feature space that we introduced earlier). As described in the previous section, to design learning algorithms whose statistical effectiveness can be controlled, one needs to come up with a class of functions whose capacity can be computed. Vapnik et al. [573, 566, 570] considered the class of hyperplanes in some dot product space $\mathcal{H}$,

$$\langle \mathbf{w}, \mathbf{x} \rangle + b = 0 \text{ where } \mathbf{w} \in \mathcal{H}, b \in \mathbb{R}, \tag{1.21}$$

corresponding to decision functions

$$f(\mathbf{x}) = \operatorname{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle + b), \tag{1.22}$$

and proposed a learning algorithm for problems which are separable by hyperplanes (sometimes said to be *linearly separable*), termed the *Generalized Portrait*, for constructing f from empirical data. It is based on two facts. First (see Chapter 7), among all hyperplanes separating the data, there exists a unique *optimal hyperplane*, distinguished by the maximum margin of separation between any training point and the hyperplane. It is the solution of

Optimal Hyperplane

$$\underset{\mathbf{w} \in \mathcal{H}, b \in \mathbb{R}}{\text{maximize}} \ \min \{ \|\mathbf{x} - \mathbf{x}_i\| \, | \mathbf{x} \in \mathcal{H}, \langle \mathbf{w}, \mathbf{x} \rangle + b = 0, i = 1, \ldots, m \}. \tag{1.23}$$

Second (see Chapter 5), the capacity (as discussed in Section 1.3) of the class of separating hyperplanes decreases with increasing margin. Hence there are theoretical arguments supporting the good generalization performance of the optimal hyperplane, cf. Chapters 5, 7, 12. In addition, it is *computationally* attractive, since we will show below that it can be constructed by solving a quadratic programming problem for which efficient algorithms exist (see Chapters 6 and 10).

Note that the form of the decision function (1.22) is quite similar to our earlier example (1.9). The ways in which the classifiers are trained, however, are different. In the earlier example, the normal vector of the hyperplane was trivially computed from the class means as $\mathbf{w} = \mathbf{c}_+ - \mathbf{c}_-$.

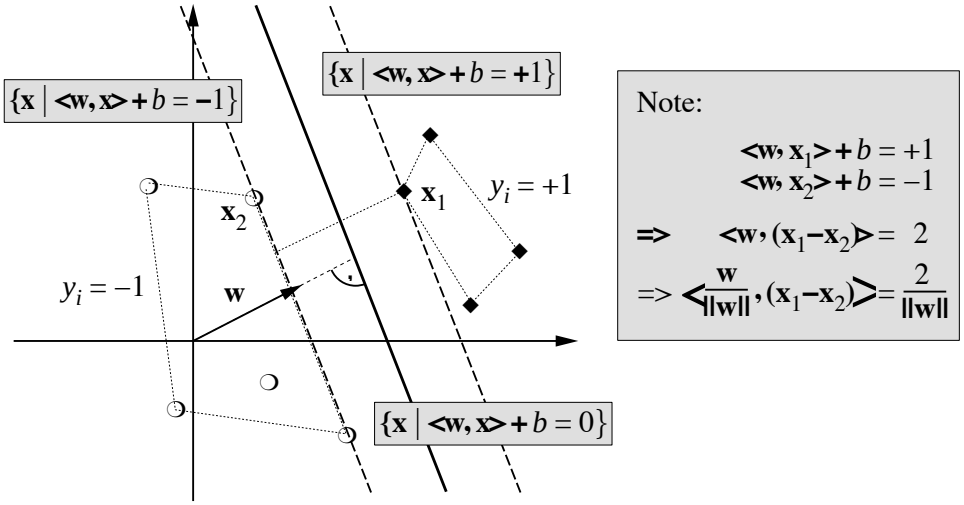

Figure 1.5 A binary classification toy problem: separate balls from diamonds. The *optimal hyperplane* (1.23) is shown as a solid line. The problem being separable, there exists a weight vector $\mathbf{w}$ and a threshold b such that $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) > 0$ $(i = 1, \ldots, m)$. Rescaling $\mathbf{w}$ and b such that the point(s) closest to the hyperplane satisfy $|\langle \mathbf{w}, \mathbf{x}_i \rangle + b| = 1$, we obtain a *canonical* form $(\mathbf{w}, b)$ of the hyperplane, satisfying $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1$. Note that in this case, the *margin* (the distance of the closest point to the hyperplane) equals $1/\|\mathbf{w}\|$. This can be seen by considering two points $\mathbf{x}_1, \mathbf{x}_2$ on opposite sides of the margin, that is, $\langle \mathbf{w}, \mathbf{x}_1 \rangle + b = 1, \langle \mathbf{w}, \mathbf{x}_2 \rangle + b = -1$, and projecting them onto the hyperplane normal vector $\mathbf{w}/\|\mathbf{w}\|$.

In the present case, we need to do some additional work to find the normal vector that leads to the largest margin. To construct the optimal hyperplane, we have to solve

$$\underset{\mathbf{w} \in \mathcal{H}, b \in \mathbb{R}}{\text{minimize}} \quad \tau(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 \tag{1.24}$$

$$\text{subject to} \quad y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \text{ for all } i = 1, \ldots, m. \tag{1.25}$$

Note that the constraints (1.25) ensure that $f(\mathbf{x}_i)$ will be $+1$ for $y_i = +1$, and -1 for $y_i = -1$. Now one might argue that for this to be the case, we don't actually need the "≥ 1" on the right hand side of (1.25). However, without it, it would not be meaningful to minimize the length of $\mathbf{w}$: to see this, imagine we wrote "> 0" instead of "≥ 1." Now assume that the solution is $(\mathbf{w}, b)$. Let us rescale this solution by multiplication with some $0 < \lambda < 1$. Since $\lambda > 0$, the constraints are still satisfied. Since $\lambda < 1$, however, the length of $\mathbf{w}$ has decreased. Hence $(\mathbf{w}, b)$ cannot be the minimizer of $\tau(\mathbf{w})$.

The "≥ 1" on the right hand side of the constraints effectively fixes the scaling of $\mathbf{w}$. In fact, any other positive number would do.

Let us now try to get an intuition for why we should be minimizing the length of $\mathbf{w}$, as in (1.24). If $\|\mathbf{w}\|$ were 1, then the left hand side of (1.25) would equal the distance from $\mathbf{x}_i$ to the hyperplane (cf. (1.23)). In general, we have to divide

$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)$ by $\|\mathbf{w}\|$ to transform it into this distance. Hence, if we can satisfy (1.25) for all $i = 1, \ldots, m$ with an $\mathbf{w}$ of minimal length, then the overall margin will be maximized.

A more detailed explanation of why this leads to the maximum margin hyperplane will be given in Chapter 7. A short summary of the argument is also given in Figure 1.5.

The function τ in (1.24) is called the *objective function*, while (1.25) are called *inequality constraints*. Together, they form a so-called *constrained optimization problem*. Problems of this kind are dealt with by introducing *Lagrange multipliers* $\alpha_i \geq 0$ and a *Lagrangian*[9]

Lagrangian

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i \left(y_i(\langle \mathbf{x}_i, \mathbf{w} \rangle + b) - 1 \right). \tag{1.26}$$

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 (in other words, a saddle point has to be found). Note that the constraint has been incorporated into the second term of the Lagrangian; it is not necessary to enforce it explicitly.

Let us try to get some intuition for this way of dealing with constrained optimization problems. If a constraint (1.25) is violated, then $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1 < 0$, 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 $\alpha_i \left(y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1 \right)$ from becoming an arbitrarily large negative number, the change in $\mathbf{w}$ and b will ensure that, provided the problem is separable, the constraint will eventually be satisfied. Similarly, one can understand that for all constraints which are not precisely met as equalities (that is, for which $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1 > 0$), 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 (KKT) complementarity conditions of optimization theory (Chapter 6).

KKT Conditions

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

$$\frac{\partial}{\partial b} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0 \text{ and } \frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0, \tag{1.27}$$

leads to

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \tag{1.28}$$

and

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i. \tag{1.29}$$

9. Henceforth, we use boldface Greek letters as a shorthand for corresponding vectors $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m)$.

Support Vector

The solution vector thus has an expansion (1.29) in terms of a subset of the training patterns, namely those patterns with non-zero α_i, called *Support Vectors (SVs)* (cf. (1.15) in the initial example). By the KKT conditions,

$$\alpha_i \left[y_i \left(\langle \mathbf{x}_i, \mathbf{w} \rangle + b \right) - 1 \right] = 0 \text{ for all } i = 1, \ldots, m, \tag{1.30}$$

the SVs lie on the margin (cf. Figure 1.5). All remaining training examples $(\mathbf{x}_j, y_j)$ are irrelevant: their constraint $y_j(\langle \mathbf{w}, \mathbf{x}_j \rangle + b) \geq 1$ (cf. (1.25)) could just as well be left out, and they do not appear in the expansion (1.29). This nicely captures our intuition of the problem: as the hyperplane (cf. Figure 1.5) is completely determined by the patterns closest to it, the solution should not depend on the other examples.

Dual Problem

By substituting (1.28) and (1.29) into the Lagrangian (1.26), one eliminates the primal variables $\mathbf{w}$ and b, arriving at the so-called *dual optimization problem*, which is the problem that one usually solves in practice:

$$\underset{\boldsymbol{\alpha} \in \mathbb{R}^m}{\text{maximize}} \quad W(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \tag{1.31}$$

$$\text{subject to} \quad \alpha_i \geq 0 \text{ for all } i = 1, \ldots, m \text{ and } \sum_{i=1}^{m} \alpha_i y_i = 0. \tag{1.32}$$

Decision Function

Using (1.29), the hyperplane decision function (1.22) can thus be written as

$$f(\mathbf{x}) = \operatorname{sgn} \left(\sum_{i=1}^{m} y_i \alpha_i \langle \mathbf{x}, \mathbf{x}_i \rangle + b \right), \tag{1.33}$$

where b is computed by exploiting (1.30) (for details, cf. Chapter 7).

The structure of the optimization problem closely resembles those that typically arise in Lagrange's formulation of mechanics (e.g., [206]). In the latter class of problem, it is also often the case that only a subset of constraints become 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, and those walls could just as well be removed.

Mechanical Analogy

Seen in this light, it is not too surprising that it is possible to give a mechanical interpretation of optimal margin hyperplanes [87]: If we assume that each SV $\mathbf{x}_i$ exerts a perpendicular force of size α_i and direction $y_j \cdot \mathbf{w}/\|\mathbf{w}\|$ on a solid plane sheet lying along the hyperplane, then the solution satisfies the requirements for mechanical stability. The constraint (1.28) states that the forces on the sheet sum to zero, and (1.29) implies that the torques also sum to zero, via $\sum_i \mathbf{x}_i \times y_i \alpha_i \mathbf{w}/\|\mathbf{w}\| = \mathbf{w} \times \mathbf{w}/\|\mathbf{w}\| = 0$.[10] This mechanical analogy illustrates the physical meaning of the term *Support* Vector.

10. Here, the $\times$ denotes the *vector* (or *cross*) *product*, satisfying $\mathbf{v} \times \mathbf{v} = 0$ for all $\mathbf{v} \in \mathcal{H}$.

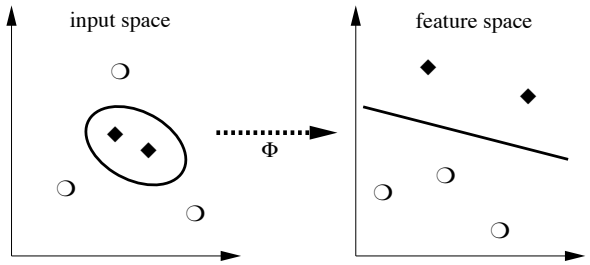

Figure 1.6 The idea of SVMs: map the training data into a higher-dimensional feature space via Φ, and construct a separating hyperplane with maximum margin there. This yields a nonlinear decision boundary in input space. By the use of a kernel function (1.2), it is possible to compute the separating hyperplane without explicitly carrying out the map into the feature space.

1.5 Support Vector Classification

We now have all the tools to describe SVMs (Figure 1.6). Everything in the last section was formulated in a dot product space. We think of this space as the feature space $\mathcal{H}$ of Section 1.1. To express the formulas in terms of the input patterns in $\mathcal{X}$, we thus need to employ (1.6), which expresses the dot product of bold face feature vectors $\mathbf{x}, \mathbf{x}'$ in terms of the kernel k evaluated on input patterns x, x',

$$k(x, x') = \langle \mathbf{x}, \mathbf{x}' \rangle . \tag{1.34}$$

This substitution, which is sometimes referred to as the *kernel trick*, was used by Boser, Guyon, and Vapnik [62] to extend the Generalized Portrait hyperplane classifier to nonlinear Support Vector Machines. Aizerman, Braverman, and Rozonoér [4] called $\mathcal{H}$ the *linearization space*, and used it in the context of the potential function classification method to express the dot product between elements of $\mathcal{H}$ in terms of elements of the input space.

The kernel trick can be applied since all feature vectors only occurred in dot products (see (1.31) and (1.33)). The weight vector (cf. (1.29)) then becomes an expansion in feature space, and therefore will typically no longer correspond to the Φ-image of a single input space vector (cf. Chapter 18). We obtain decision functions of the form (cf. (1.33))

Decision Function

$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^{m} y_i \alpha_i \langle \Phi(x), \Phi(x_i) \rangle + b\right) = \operatorname{sgn}\left(\sum_{i=1}^{m} y_i \alpha_i k(x, x_i) + b\right), \tag{1.35}$$

and the following quadratic program (cf. (1.31)):

$$\underset{\boldsymbol{\alpha} \in \mathbb{R}^m}{\text{maximize}} \quad W(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \tag{1.36}$$

$$\text{subject to} \quad \alpha_i \geq 0 \text{ for all } i = 1, \ldots, m, \quad \text{and} \quad \sum_{i=1}^{m} \alpha_i y_i = 0. \tag{1.37}$$