1.2 A Simple Pattern Recognition Algorithm

We are now in the position to describe a pattern recognition learning algorithm that is arguably one of the simplest possible. We make use of the structure introduced in the previous section; that is, we assume that our data are embedded into a dot product space $\mathcal{H}$.[3] Using the dot product, we can measure distances in this space. The basic idea of the algorithm is to assign a previously unseen pattern to the class with closer mean.

We thus begin by computing the means of the two classes in feature space;

$$\mathbf{c}_+ = \frac{1}{m_+} \sum_{\{i|y_i=+1\}} \mathbf{x}_i, \tag{1.7}$$

$$\mathbf{c}_- = \frac{1}{m_-} \sum_{\{i|y_i=-1\}} \mathbf{x}_i, \tag{1.8}$$

where m_+ and m_- are the number of examples with positive and negative labels, respectively. We assume that both classes are non-empty, thus $m_+, m_- > 0$. We assign a new point $\mathbf{x}$ to the class whose mean is closest (Figure 1.1). This geometric construction can be formulated in terms of the dot product $\langle \cdot, \cdot \rangle$. Half way between $\mathbf{c}_+$ and $\mathbf{c}_-$ lies the point $\mathbf{c} := (\mathbf{c}_+ + \mathbf{c}_-)/2$. We compute the class of $\mathbf{x}$ by checking whether the vector $\mathbf{x} - \mathbf{c}$ connecting $\mathbf{c}$ to $\mathbf{x}$ encloses an angle smaller than $\pi/2$ with the vector $\mathbf{w} := \mathbf{c}_+ - \mathbf{c}_-$ connecting the class means. This leads to

$$\begin{aligned} y &= \operatorname{sgn} \langle (\mathbf{x} - \mathbf{c}), \mathbf{w} \rangle \\ &= \operatorname{sgn} \langle (\mathbf{x} - (\mathbf{c}_+ + \mathbf{c}_-)/2), (\mathbf{c}_+ - \mathbf{c}_-) \rangle \\ &= \operatorname{sgn} (\langle \mathbf{x}, \mathbf{c}_+ \rangle - \langle \mathbf{x}, \mathbf{c}_- \rangle + b). \end{aligned} \tag{1.9}$$

Here, we have defined the offset

$$b := \frac{1}{2}(\|\mathbf{c}_-\|^2 - \|\mathbf{c}_+\|^2), \tag{1.10}$$

with the norm $\|\mathbf{x}\| := \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$. If the class means have the same distance to the origin, then b will vanish.

Note that (1.9) induces a decision boundary which has the form of a hyperplane (Figure 1.1); that is, a set of points that satisfy a constraint expressible as a linear equation.

It is instructive to rewrite (1.9) in terms of the input patterns x_i, using the kernel k to compute the dot products. Note, however, that (1.6) only tells us how to compute the dot products between vectorial representations $\mathbf{x}_i$ of inputs x_i. We therefore need to express the vectors $\mathbf{c}_i$ and $\mathbf{w}$ in terms of $\mathbf{x}_1, \ldots, \mathbf{x}_m$.

Decision Function

To this end, substitute (1.7) and (1.8) into (1.9) to get the *decision function*

3. For the definition of a dot product space, see Section B.2.

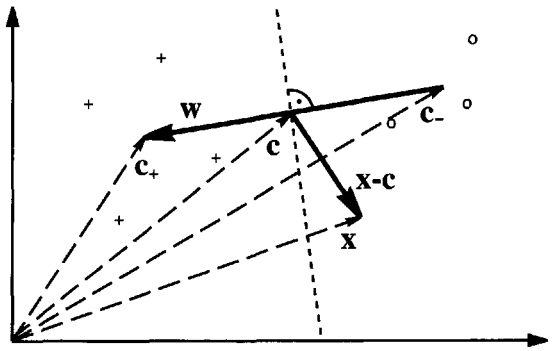

Figure 1.1 A simple geometric classification algorithm: given two classes of points (depicted by 'o' and '+'), compute their means $\mathbf{c}_+, \mathbf{c}_-$ and assign a test pattern $\mathbf{x}$ to the one whose mean is closer. This can be done by looking at the dot product between $\mathbf{x} - \mathbf{c}$ (where $\mathbf{c} = (\mathbf{c}_+ + \mathbf{c}_-)/2$) and $\mathbf{w} := \mathbf{c}_+ - \mathbf{c}_-$, which changes sign as the enclosed angle passes through $\pi/2$. Note that the corresponding decision boundary is a hyperplane (the dotted line) orthogonal to $\mathbf{w}$.

$$y = \operatorname{sgn}\left(\frac{1}{m_+}\sum_{\{i|y_i=+1\}}\langle \mathbf{x}, \mathbf{x}_i\rangle - \frac{1}{m_-}\sum_{\{i|y_i=-1\}}\langle \mathbf{x}, \mathbf{x}_i\rangle + b\right)$$
$$= \operatorname{sgn}\left(\frac{1}{m_+}\sum_{\{i|y_i=+1\}}k(x, x_i) - \frac{1}{m_-}\sum_{\{i|y_i=-1\}}k(x, x_i) + b\right). \tag{1.11}$$

Similarly, the offset becomes

$$b := \frac{1}{2}\left(\frac{1}{m_-^2}\sum_{\{(i,j)|y_i=y_j=-1\}}k(x_i, x_j) - \frac{1}{m_+^2}\sum_{\{(i,j)|y_i=y_j=+1\}}k(x_i, x_j)\right). \tag{1.12}$$

Surprisingly, it turns out that this rather simple-minded approach contains a well-known statistical classification method as a special case. Assume that the class means have the same distance to the origin (hence $b = 0$, cf. (1.10)), and that k can be viewed as a probability density when one of its arguments is fixed. By this we mean that it is positive and has unit integral,[4]

$$\int_{\mathcal{X}} k(x, x')dx = 1 \text{ for all } x' \in \mathcal{X}. \tag{1.13}$$

In this case, (1.11) takes the form of the so-called Bayes classifier separating the two classes, subject to the assumption that the two classes of patterns were generated by sampling from two probability distributions that are correctly estimated by the

4. In order to state this assumption, we have to require that we can define an integral on $\mathcal{X}$.

Parzen windows estimators of the two class densities,

$$p_+(x) := \frac{1}{m_+} \sum_{\{i|y_i=+1\}} k(x, x_i) \text{ and } p_-(x) := \frac{1}{m_-} \sum_{\{i|y_i=-1\}} k(x, x_i), \tag{1.14}$$

Parzen Windows

where $x \in \mathcal{X}$.

Given some point x, the label is then simply computed by checking which of the two values $p_+(x)$ or $p_-(x)$ is larger, which leads directly to (1.11). Note that this decision is the best we can do if we have no prior information about the probabilities of the two classes.

The classifier (1.11) is quite close to the type of classifier that this book deals with in detail. Both take the form of kernel expansions on the input domain,

$$y = \operatorname{sgn}\left(\sum_{i=1}^{m} \alpha_i k(x, x_i) + b\right). \tag{1.15}$$

In both cases, the expansions correspond to a separating hyperplane in a feature space. In this sense, the α_i can be considered a *dual representation* of the hyperplane's normal vector [223]. Both classifiers are example-based in the sense that the kernels are centered on the training patterns; that is, one of the two arguments of the kernel is always a training pattern. A test point is classified by comparing it to all the training points that appear in (1.15) with a nonzero weight.

More sophisticated classification techniques, to be discussed in the remainder of the book, deviate from (1.11) mainly in the selection of the patterns on which the kernels are centered and in the choice of weights α_i that are placed on the individual kernels in the decision function. It will no longer be the case that *all* training patterns appear in the kernel expansion, and the weights of the kernels in the expansion will no longer be uniform within the classes — recall that in the current example, cf. (1.11), the weights are either $(1/m_+)$ or $(-1/m_-)$, depending on the class to which the pattern belongs.

In the feature space representation, this statement corresponds to saying that we will study normal vectors $\mathbf{w}$ of decision hyperplanes that can be represented as general linear combinations (i.e., with non-uniform coefficients) of the training patterns. For instance, we might want to remove the influence of patterns that are very far away from the decision boundary, either since we expect that they will not improve the generalization error of the decision function, or since we would like to reduce the computational cost of evaluating the decision function (cf. (1.11)). The hyperplane will then only depend on a subset of training patterns called *Support Vectors*.

1.3 Some Insights From Statistical Learning Theory

With the above example in mind, let us now consider the problem of pattern recognition in a slightly more formal setting [559, 152, 186]. This will allow us to indicate the factors affecting the design of "better" algorithms. Rather than just