$$J(\mathbf{a}(k+1)) \simeq J(\mathbf{a}(k)) - \eta(k)\|\boldsymbol{\nabla} J\|^2 + \frac{1}{2}\eta^2(k)\boldsymbol{\nabla} J^t\mathbf{H}\boldsymbol{\nabla} J.$$

From this it follows (Problem 12) that $J(\mathbf{a}(k+1))$ can be minimized by the choice

$$\eta(k) = \frac{\|\boldsymbol{\nabla} J\|^2}{\boldsymbol{\nabla} J^t\mathbf{H}\boldsymbol{\nabla} J}, \tag{14}$$

where $\mathbf{H}$ depends on $\mathbf{a}$, and thus indirectly on k. This then is the optimal choice of $\eta(k)$ given the assumptions mentioned. Note that if the criterion function $J(\mathbf{a})$ is quadratic throughout the region of interest, then $\mathbf{H}$ is constant and η is a constant independent of k.

Newton's algorithm

An alternative approach, obtained by ignoring Eq. 12 and by choosing $\mathbf{a}(k+1)$ to minimize the second-order expansion, is *Newton's algorithm* where line 3 in Algorithm 1 is replaced by

$$\mathbf{a}(k+1) = \mathbf{a}(k) - \mathbf{H}^{-1}\boldsymbol{\nabla} J, \tag{15}$$

leading to the following algorithm:

Algorithm 2 (Newton descent)

```
1 begin initialize  $\mathbf{a}$, criterion $\theta$
2         do
3             $\mathbf{a} \leftarrow \mathbf{a} - \mathbf{H}^{-1}\boldsymbol{\nabla} J(\mathbf{a})$
4         until $\mathbf{H}^{-1}\boldsymbol{\nabla} J(\mathbf{a}) < \theta$
5     return $\mathbf{a}$
6 end
```

Simple gradient descent and Newton's algorithm are compared in Fig. 5.10.

Generally speaking, Newton's algorithm will usually give a greater improvement *per step* than the simple gradient descent algorithm, even with the optimal value of $\eta(k)$. However, Newton's algorithm is not applicable if the Hessian matrix $\mathbf{H}$ is singular. Furthermore, even when $\mathbf{H}$ is nonsingular, the $O(d^3)$ time required for matrix inversion on each iteration can easily offset the descent advantage. In fact, it often takes less time to set $\eta(k)$ to a constant η that is smaller than necessary and make a few more corrections than it is to compute the optimal $\eta(k)$ at each step (Computer exercise 1).

5.5 Minimizing the Perceptron Criterion Function

5.5.1 The Perceptron Criterion Function

Consider now the problem of constructing a criterion function for solving the linear inequalities $\mathbf{a}^t\mathbf{y}_i > 0$. The most obvious choice is to let $J(\mathbf{a};\ \mathbf{y}_1, ..., \mathbf{y}_n)$ be the number of samples misclassified by $\mathbf{a}$. However, because this function is piecewise constant, it is obviously a poor candidate for a gradient search. A better choice is the *Perceptron criterion function*

$$J_p(\mathbf{a}) = \sum_{\mathbf{y}\in\mathcal{Y}}(-\mathbf{a}^t\mathbf{y}), \tag{16}$$

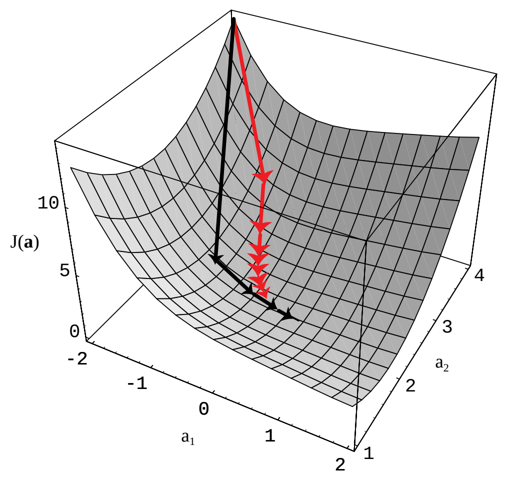


Figure 5.10: The sequence of weight vectors given by a simple gradient descent method (red) and by Newton's (second order) algorithm (black). Newton's method typically leads to greater improvement per step, even when using optimal learning rates for both methods. However the added computational burden of inverting the Hessian matrix used in Newton's method is not always justified, and simple descent may suffice.

where $\mathcal{Y}(\mathbf{a})$ is the set of samples *misclassified* by $\mathbf{a}$. (If no samples are misclassified, $\mathcal{Y}$ is empty and we define J_p to be zero.) Since $\mathbf{a}^t\mathbf{y} \leq 0$ if $\mathbf{y}$ is misclassified, $J_p(\mathbf{a})$ is never negative, being zero only if $\mathbf{a}$ is a solution vector, or if $\mathbf{a}$ is on the decision boundary. Geometrically, $J_p(\mathbf{a})$ is proportional to the sum of the distances from the misclassified samples to the decision boundary. Figure 5.11 illustrates J_p for a simple two-dimensional example.

Since the jth component of the gradient of J_p is $\partial J_p/\partial a_j$, we see from Eq. 16 that

$$\nabla J_p = \sum_{\mathbf{y}\in\mathcal{Y}} (-\mathbf{y}), \tag{17}$$

and hence the update rule becomes

$$\mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k) \sum_{\mathbf{y}\in\mathcal{Y}_k} \mathbf{y}, \tag{18}$$

where $\mathcal{Y}_k$ is the set of samples misclassified by $\mathbf{a}(k)$. Thus the Perceptron algorithm is:

Algorithm 3 (Batch Perceptron)

```
1 begin initialize  $\mathbf{a}, \eta(\cdot)$, criterion $\theta, k = 0$
2           do  $k \leftarrow k + 1$
3               $\mathbf{a} \leftarrow \mathbf{a} + \eta(k) \sum_{\mathbf{y}\in\mathcal{Y}_k} \mathbf{y}$
4           until $\eta(k) \sum_{\mathbf{y}\in\mathcal{Y}_k} \mathbf{y} < \theta$
5     return $\mathbf{a}$
6 end
```

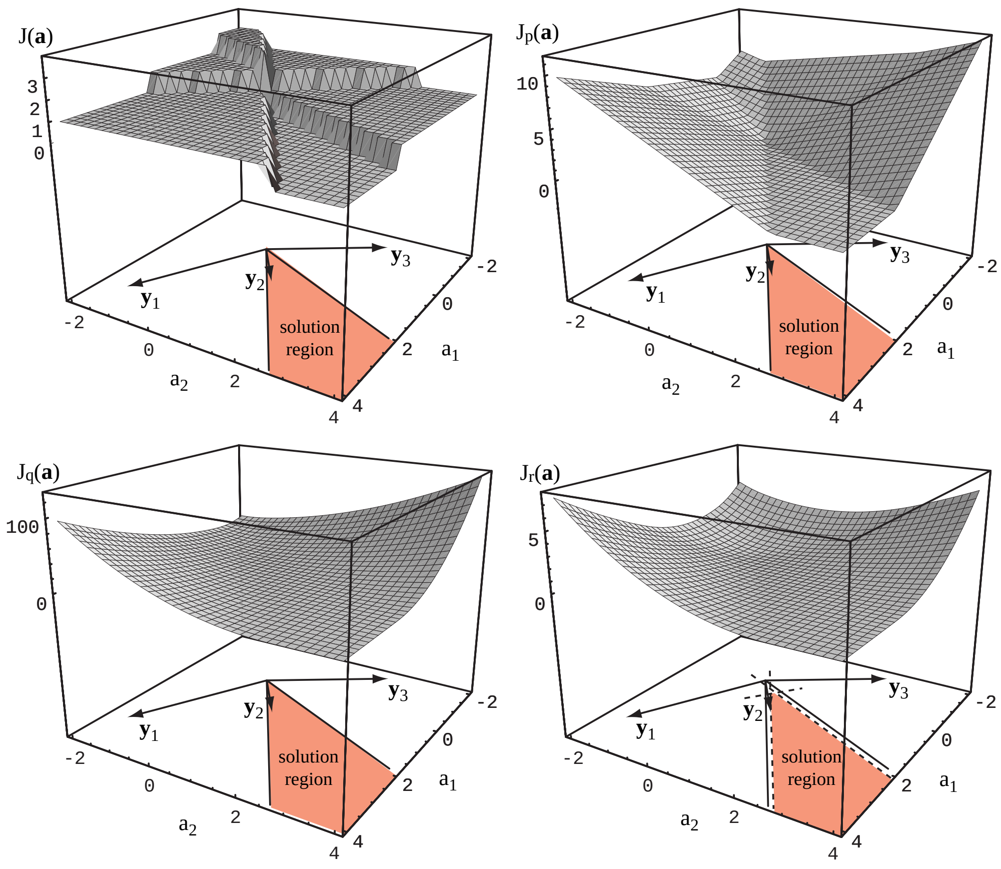


Figure 5.11: Four learning criteria as a function of weights in a linear classifier. At the upper left is the total number of patterns misclassified, which is piecewise constant and hence unacceptable for gradient descent procedures. At the upper right is the Perceptron criterion (Eq. 16), which is piecewise linear and acceptable for gradient descent. The lower left is squared error (Eq. 32), which has nice analytic properties and is useful even when the patterns are not linearly separable. The lower right is the square error with margin (Eq. 33). A designer may adjust the margin b in order to force the solution vector to lie toward the middle of the $b = 0$ solution region in hopes of improving generalization of the resulting classifier.

BATCH TRAINING

Thus, the batch Perceptron algorithm for finding a solution vector can be stated very simply: the next weight vector is obtained by adding some multiple of the sum of the misclassified samples to the present weight vector. We use the term "batch" to refer to the fact that (in general) a large group of samples is used when computing each weight update. (We shall soon see alternate methods based on single samples.) Figure 5.12 shows how this algorithm yields a solution vector for a simple two-dimensional example with $\mathbf{a}(1) = \mathbf{0}$, and $\eta(k) = 1$. We shall now show that it will yield a solution for any linearly separable problem.

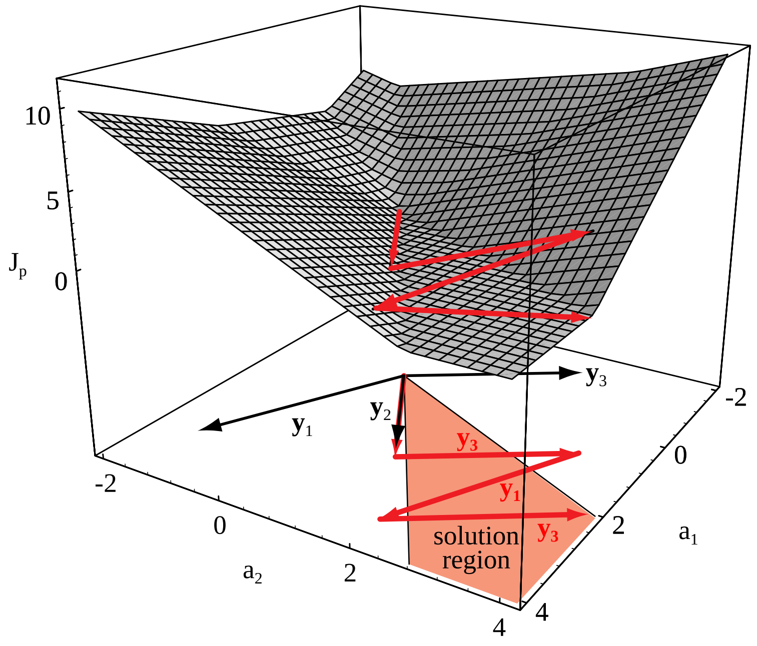


Figure 5.12: The Perceptron criterion, J_p is plotted as a function of the weights a_1 and a_2 for a three-pattern problem. The weight vector begins at $\mathbf{0}$, and the algorithm sequentially adds to it vectors equal to the "normalized" misclassified patterns themselves. In the example shown, this sequence is $\mathbf{y}_2, \mathbf{y}_3, \mathbf{y}_1, \mathbf{y}_3$, at which time the vector lies in the solution region and iteration terminates. Note that the second update (by $\mathbf{y}_3$) takes the candidate vector *farther* from the solution region than after the first update (cf. Theorem 5.1. (In an alternate, batch method, *all* the misclassified points are added at each iteration step leading to a smoother trajectory in weight space.)

5.5.2 Convergence Proof for Single-Sample Correction

We shall begin our examination of convergence properties of the Perceptron algorithm with a variant that is easier to analyze. Rather than testing $\mathbf{a}(k)$ on all of the samples and basing our correction of the set $\mathcal{Y}_k$ of misclassified training samples, we shall consider the samples in a sequence and shall modify the weight vector whenever it misclassifies a *single* sample. For the purposes of the convergence proof, the detailed nature of the sequence is unimportant as long as every sample appears in the sequence infinitely often. The simplest way to assure this is to repeat the samples cyclically, though from a practical point of view random selection is often to be preferred (Sec. 5.8.5). Clearly neither the batch nor this single-sample version of the Perceptron algorithm are on-line since we must store and potentially revisit all of the training patterns.

Two further simplifications help to clarify the exposition. First, we shall temporarily restrict our attention to the case in which $\eta(k)$ is constant — the so-called *fixed-increment* case. It is clear from Eq. 18 that if $\eta(t)$ is constant it merely serves to scale the samples; thus, in the fixed-increment case we can take $\eta(t) = 1$ with no loss in generality. The second simplification merely involves notation. When the samples

FIXED INCREMENT

are considered sequentially, some will be misclassified. Since we shall only change the weight vector when there is an error, we really need only pay attention to the misclassified samples. Thus we shall denote the sequence of samples using superscripts, i.e., by $\mathbf{y}^1,\ \mathbf{y}^2, ...,\ \mathbf{y}^k, ...$, where each $\mathbf{y}^k$ is one of the n samples $\mathbf{y}_1, ..., \mathbf{y}_n$, and where each $\mathbf{y}^k$ is misclassified. For example, if the samples $\mathbf{y}_1$, $\mathbf{y}_2$, and $\mathbf{y}_3$ are considered cyclically, and if the marked samples

$$\overset{\downarrow}{\mathbf{y}_1},\ \mathbf{y}_2,\ \overset{\downarrow}{\mathbf{y}_3},\ \overset{\downarrow}{\mathbf{y}_1},\ \overset{\downarrow}{\mathbf{y}_2},\ \mathbf{y}_3,\ \mathbf{y}_1,\ \overset{\downarrow}{\mathbf{y}_2},\ ... \tag{19}$$

FIXED-INCREMENT RULE

are misclassified, then the sequence $\mathbf{y}^1,\ \mathbf{y}^2,\ \mathbf{y}^3,\ \mathbf{y}^4,\ \mathbf{y}^5, ...$ denotes the sequence $\mathbf{y}_1,\ \mathbf{y}_3,\ \mathbf{y}_1,\ \mathbf{y}_2,\ \mathbf{y}_2, ...$ With this understanding, the *fixed-increment rule* for generating a sequence of weight vectors can be written as

$$\left.\begin{array}{ll}\mathbf{a}(1) & \text{arbitrary} \\ \mathbf{a}(k+1) = \mathbf{a}(k) + \mathbf{y}^k & k \geq 1\end{array}\right\} \tag{20}$$

where $\mathbf{a}^t(k)\mathbf{y}^k \leq 0$ for all k. If we let n denote the total number of patterns, the algorithm is:

Algorithm 4 (Fixed-increment single-sample Perceptron)

```
1 begin initialize  $\mathbf{a}, k = 0$
2         do  $k \leftarrow (k+1) \text{mod} n$
3             if  $\mathbf{y}_k$ is misclassified by $\mathbf{a}$  then  $\mathbf{a} \leftarrow \mathbf{a} - \mathbf{y}_k$
4         until all patterns properly classified
5    return $\mathbf{a}$
6 end
```

The fixed-increment Perceptron rule is the simplest of many algorithms that have been proposed for solving systems of linear inequalities. Geometrically, its interpretation in weight space is particularly clear. Since $\mathbf{a}(k)$ misclassifies $\mathbf{y}^k$, $\mathbf{a}(k)$ is not on the positive side of the $\mathbf{y}^k$ hyperplane $\mathbf{a}^t\mathbf{y}^k = 0$. The addition of $\mathbf{y}^k$ to $\mathbf{a}(k)$ moves the weight vector directly toward and perhaps across this hyperplane. Whether the hyperplane is crossed or not, the new inner product $\mathbf{a}^t(k+1)\mathbf{y}^k$ is larger than the old inner product $\mathbf{a}^t(k)\mathbf{y}^k$ by the amount $\|\mathbf{y}^k\|^2$, and the correction is clearly moving the weight vector in a good direction (Fig. 5.13).

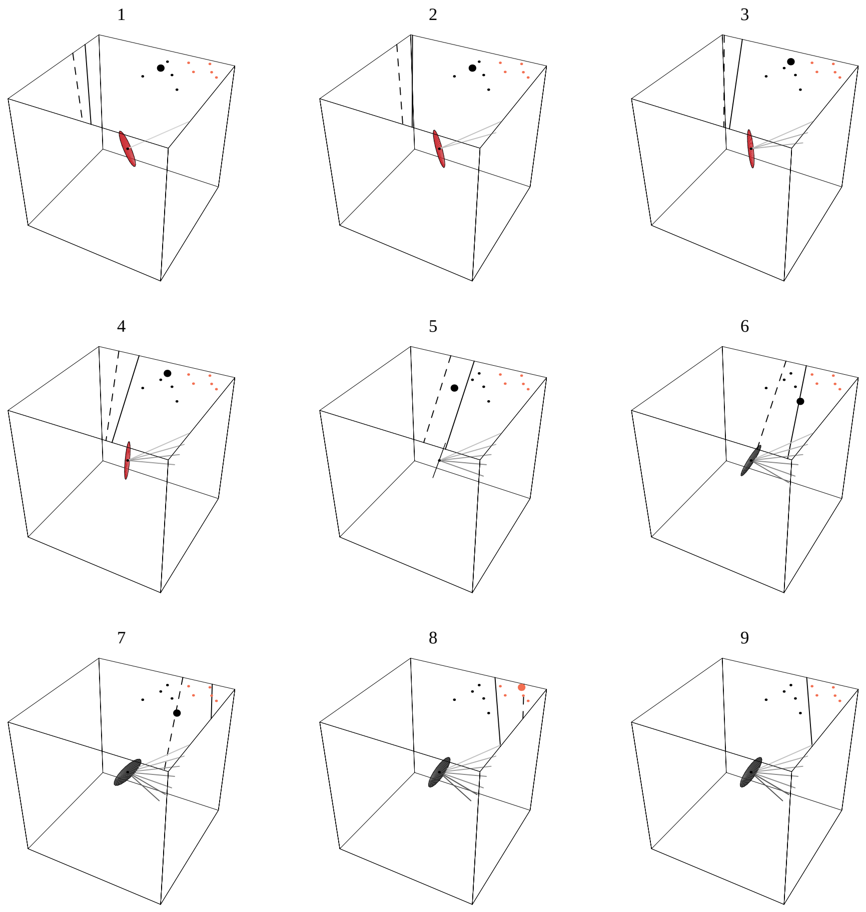


Figure 5.13: Samples from two categories, ω_1 (black) and ω_2 (red) are shown in augmented feature space, along with an augmented weight vector $\mathbf{a}$. At each step in a fixed-increment rule, one of the misclassified patterns, $\mathbf{y}^k$, is shown by the large dot. A correction $\Delta\mathbf{a}$ (proportional to the pattern vector $\mathbf{y}^k$) is added to the weight vector — towards an ω_1 point or away from an ω_2 point. This changes the decision boundary from the dashed position (from the previous update) to the solid position. The sequence of resulting $\mathbf{a}$ vectors is shown, where later values are shown darker. In this example, by step 9 a solution vector has been found and the categories successfully separated by the decision boundary shown.

Clearly this algorithm can only terminate if the samples are linearly separable; we now prove that indeed it terminates so long as the samples are linearly separable.

Theorem 5.1 (Perceptron Convergence) *If training samples are linearly separable then the sequence of weight vectors given by Algorithm 4 will terminate at a solution vector.*

Proof:

In seeking a proof, it is natural to try to show that each correction brings the weight vector closer to the solution region. That is, one might try to show that if $\hat{\mathbf{a}}$ is any solution vector, then $\|\mathbf{a}(k+1) - \hat{\mathbf{a}}\|$ is smaller than $\|\mathbf{a}(k) - \hat{\mathbf{a}}\|$. While this turns out not to be true in general (cf. steps 6 & 7 in Fig. 5.13), we shall see that it is true for solution vectors that are sufficiently long.

Let $\hat{\mathbf{a}}$ be any solution vector, so that $\hat{\mathbf{a}}^t\mathbf{y}_i$ is strictly positive for all i, and let α be a positive scale factor. From Eq. 20,

$$\mathbf{a}(k+1) - \alpha\hat{\mathbf{a}} = (\mathbf{a}(k) - \alpha\hat{\mathbf{a}}) + \mathbf{y}^k$$

and hence

$$\|\mathbf{a}(k+1) - \alpha\hat{\mathbf{a}}\|^2 = \|\mathbf{a}(k) - \alpha\hat{\mathbf{a}}\|^2 + 2(\mathbf{a}(k) - \alpha\hat{\mathbf{a}})^t\mathbf{y}^k + \|\mathbf{y}^k\|^2.$$

Since $\mathbf{y}^k$ was misclassified, $\mathbf{a}^t(k)\mathbf{y}^k \leq 0$, and thus

$$\|\mathbf{a}(k+1) - \alpha\hat{\mathbf{a}}\|^2 \leq \|\mathbf{a}(k) - \alpha\hat{\mathbf{a}}\|^2 - 2\alpha\hat{\mathbf{a}}^t\mathbf{y}^k + \|\mathbf{y}^k\|^2.$$

Because $\hat{\mathbf{a}}^t\mathbf{y}^k$ is strictly positive, the second term will dominate the third if α is sufficiently large. In particular, if we let β be the maximum length of a pattern vector,

$$\beta^2 = \max_i \|\mathbf{y}_i\|^2, \tag{21}$$

and γ be the smallest inner product of the solution vector with any pattern vector, i.e.,

$$\gamma = \min_i \left[\hat{\mathbf{a}}^t\mathbf{y}_i\right] > 0, \tag{22}$$

then we have the inequality

$$\|\mathbf{a}(k+1) - \alpha\hat{\mathbf{a}}\|^2 \leq \|\mathbf{a}(k) - \alpha\hat{\mathbf{a}}\|^2 - 2\alpha\gamma + \beta^2.$$

If we choose

$$\alpha = \frac{\beta^2}{\gamma}, \tag{23}$$

we obtain

$$\|\mathbf{a}(k+1) - \alpha\hat{\mathbf{a}}\|^2 \leq \|\mathbf{a}(k) - \alpha\hat{\mathbf{a}}\|^2 - \beta^2.$$

Thus, the squared distance from $\mathbf{a}(k)$ to $\alpha\hat{\mathbf{a}}$ is reduced by at least β^2 at each correction, and after k corrections

$$\|\mathbf{a}(k+1) - \alpha\hat{\mathbf{a}}\|^2 \leq \|\mathbf{a}(k) - \alpha\hat{\mathbf{a}}\|^2 - k\beta^2. \tag{24}$$

Since the squared distance cannot become negative, it follows that the sequence of corrections must terminate after no more than k_0 corrections, where

$$k_0 = \frac{\|\mathbf{a}(1) - \alpha\hat{\mathbf{a}}\|^2}{\beta^2}. \tag{25}$$

Since a correction occurs whenever a sample is misclassified, and since each sample appears infinitely often in the sequence, it follows that when corrections cease the resulting weight vector must classify all of the samples correctly. ■

The number k_0 gives us a bound on the number of corrections. If $\mathbf{a}(1) = \mathbf{0}$, we get the following particularly simple expression for k_0:

$$k_0 = \frac{\alpha^2\|\hat{\mathbf{a}}\|^2}{\beta^2} = \frac{\beta^2\alpha^2\|\hat{\mathbf{a}}\|^2}{\gamma^2} = \frac{\max\limits_i \|\mathbf{y}_i\|^2\|\hat{\mathbf{a}}\|^2}{\min\limits_i [\mathbf{y}_i^t\hat{\mathbf{a}}]^2}. \tag{26}$$

The denominator in Eq. 26 shows that the difficulty of the problem is essentially determined by the samples most nearly orthogonal to the solution vector. Unfortunately, it provides no help when we face an unsolved problem, since the bound is expressed in terms of a solution vector which is unknown. In general, it is clear that linearly-separable problems can be made arbitrarily difficult to solve by making the samples almost coplanar (Computer exercise 2). Nevertheless, if the training samples are linearly separable, the fixed-increment rule will yield a solution after a finite number of corrections.

5.5.3 Some Direct Generalizations

The fixed increment rule can be generalized to provide a variety of related algorithms. We shall briefly consider two variants of particular interest. The first variant introduces a *variable increment* $\eta(k)$ and a margin b, and calls for a correction whenever $\mathbf{a}^t(k)\mathbf{y}^k$ fails to excede the margin. The update is given by

VARIABLE INCREMENT

$$\left.\begin{array}{ll} \mathbf{a}(1) & \text{arbitrary} \\ \mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k)\mathbf{y}^k & k \geq 1, \end{array}\right\} \tag{27}$$

where now $\mathbf{a}^t(k)\mathbf{y}^k \leq b$ for all k. Thus for n patterns, our algorithm is:

Algorithm 5 (Variable increment Perceptron with margin)

```
1 begin initialize  $\mathbf{a}$, criterion $\theta$, margin $b, \eta(\cdot), k = 0$
2           do  $k \leftarrow k + 1$
3               if  $\mathbf{a}^t\mathbf{y}_k + b < 0$  then  $\mathbf{a} \leftarrow \mathbf{a} - \eta(k)\mathbf{y}_k$
4           until $\mathbf{a}^t\mathbf{y}_k + b \leq 0$ for all $k$
5    return $\mathbf{a}$
6 end
```

It can be shown that if the samples are linearly separable and if

$$\eta(k) \geq 0, \tag{28}$$

$$\lim_{m\to\infty} \sum_{k=1}^{m} \eta(k) = \infty \tag{29}$$

and

$$\lim_{m\to\infty} \frac{\sum\limits_{k=1}^{m} \eta^2(k)}{\left(\sum\limits_{k=1}^{m} \eta(k)\right)^2} = 0, \tag{30}$$

then $\mathbf{a}(k)$ converges to a solution vector $\mathbf{a}$ satisfying $\mathbf{a}^t\mathbf{y}_i > b$ for all i (Problem 18). In particular, these conditions on $\eta(k)$ are satisfied if $\eta(k)$ is a positive constant, or if it decreases like $1/k$.

Another variant of interest is our original gradient descent algorithm for J_p,

$$\left.\begin{array}{ll} \mathbf{a}(1) & \text{arbitrary} \\ \mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k) \sum\limits_{\mathbf{y}\in\mathcal{Y}_k} \mathbf{y}, & \end{array}\right\} \tag{31}$$

where $\mathcal{Y}_k$ is the set of training samples misclassified by $\mathbf{a}(k)$. It is easy to see that this algorithm will also yield a solution once one recognizes that if $\hat{\mathbf{a}}$ is a solution vector for $\mathbf{y}_1, ..., \mathbf{y}_n$, then it correctly classifies the correction vector

$$\mathbf{y}^k = \sum_{\mathbf{y}\in\mathcal{Y}_k} \mathbf{y}.$$

In greater detail, then, the algorithm is

Algorithm 6 (Batch variable increment Perceptron)

```
1  begin initialize $\mathbf{a}, \eta(\cdot), k = 0$
2          do $k \leftarrow k + 1$
3              $\mathcal{Y}_k = \{\}$
4              $j = 0$
5              do $j \leftarrow j + 1$
6                  if $\mathbf{y}_j$ is misclassified then Append $\mathbf{y}_j$ to $\mathcal{Y}_k$
7              until $j = n$
8              $\mathbf{a} \leftarrow \mathbf{a} + \eta(k) \sum_{\mathbf{y}\in\mathcal{Y}_k} \mathbf{y}$
9          until $\mathcal{Y}_k = \{\}$
10 return $\mathbf{a}$
11 end
```

The benefit of batch gradient descent is that the trajectory of the weight vector is smoothed, compared to that in corresponding single-sample algorithms (e.g., Algorithm 5), since at each update the full set of misclassified patterns is used — the local statistical variations in the misclassified patterns tend to cancel while the large-scale trend does not. Thus, if the samples are linearly separable, all of the possible correction vectors form a linearly separable set, and if $\eta(k)$ satisfies Eqs. 28–30, the sequence of weight vectors produced by the gradient descent algorithm for $J_p(\cdot)$ will always converge to a solution vector.

It is interesting to note that the conditions on $\eta(k)$ are satisfied if $\eta(k)$ is a positive constant, if it decreases as $1/k$, or even if it increases as k. Generally speaking, one would prefer to have $\eta(k)$ become smaller as time goes on. This is particularly true if there is reason to believe that the set of samples is not linearly separable, since it reduces the disruptive effects of a few "bad" samples. However, in the separable case it is a curious fact that one can allow $\eta(k)$ to become larger and still obtain a solution.

This observation brings out one of the differences between theoretical and practical attitudes. From a theoretical viewpoint, it is interesting that we can obtain a solution in a finite number of steps for any finite set of separable samples, for any initial weight vector $\mathbf{a}(1)$, for any nonnegative margin b, and for any scale factor $\eta(k)$ satisfying Eqs. 28–30. From a practical viewpoint, we want to make wise choices for these quantities. Consider the margin b, for example. If b is much smaller than $\eta(k)\|\mathbf{y}^k\|^2$, the amount by which a correction increases $\mathbf{a}^t(k)\mathbf{y}^k$, it is clear that it will have little effect at all. If it is much larger than $\eta(k)\|\mathbf{y}^k\|^2$, many corrections will be needed to satisfy the conditions $\mathbf{a}^t(k)\mathbf{y}^k > b$. A value close to $\eta(k)\|\mathbf{y}^k\|^2$ is often a useful compromise. In addition to these choices for $\eta(k)$ and b, the scaling of the components of $\mathbf{y}^k$ can also have a great effect on the results. The possession of a convergence theorem does not remove the need for thought in applying these techniques.

A close descendant of the Perceptron algorithm is the Winnow algorithm, which has applicability to separable training data. The key difference is that while the weight vector returned by the Perceptron algorithm has components a_i $(i = 0, ...d)$, in Winnow they are scaled according to 2sinh$[a_i]$. In one version, the balanced Winnow algorithm, there are separate "positive" and "negative" weight vectors, $\mathbf{a}^+$ and $\mathbf{a}^-$, each associated with one of the two categories to be learned. Corrections on the positive weight are made if and only if a training pattern in ω_1 is misclassified; conversely, corrections on the negative weight are made if and only if a training pattern in ω_2 is misclassified.

WINNOW ALGORITHM

Algorithm 7 (Balanced Winnow)

```
1 begin initialize $\mathbf{a}^+, \mathbf{a}^-, \eta(\cdot), k \leftarrow 0, \alpha > 1$
2          if sign$[\mathbf{a}^{+t}\mathbf{y}_k - \mathbf{a}^{-t}\mathbf{y}_k] \neq z_k$ (pattern misclassified)
3          then if $z_k = +1$ then $a_i^+ \leftarrow \alpha^{+y_i} a_i^+$; $a_i^- \leftarrow \alpha^{-y_i} a_i^-$ for all $i$
4               if $z_k = -1$ then $a_i^+ \leftarrow \alpha^{-y_i} a_i^+$; $a_i^- \leftarrow \alpha^{+y_i} a_i^-$ for all $i$
5    return $\mathbf{a}^+, \mathbf{a}^-$
6 end
```

There are two main benefits of such a version of the Winnow algorithm. The first is that during training each of the two consituent weight vectors moves in a uniform direction and this means the "gap," determined by these two vectors, can never increase in size for separable data. This leads to a convergence proof that, while somewhat more complicated, is nevertheless more general than the Perceptron convergence theorem (cf. Bibliography). The second benefit is that convergence is generally faster than in a Perceptron, since for proper setting of learning rate, each constituent weight does not overshoot its final value. This benefit is especially pronounced whenever a large number of irrelevant or redundant features are present (Computer exercise 6).

5.6 Relaxation Procedures

5.6.1 The Descent Algorithm

The criterion function J_p is by no means the only function we can construct that is minimized when $\mathbf{a}$ is a solution vector. A close but distinct relative is

$$J_q(\mathbf{a}) = \sum_{\mathbf{y}\in\mathcal{Y}} (\mathbf{a}^t\mathbf{y})^2, \tag{32}$$

where $\mathcal{Y}(\mathbf{a})$ again denotes the set of training samples misclassified by $\mathbf{a}$. Like J_p, J_q focuses attention on the misclassified samples. Its chief difference is that its gradient is continuous, whereas the gradient of J_p is not. Thus, J_q presents a smoother surface to search (Fig. 5.11). Unfortunately, J_q is so smooth near the boundary of the solution region that the sequence of weight vectors can converge to a point on the boundary. It is particularly embarrassing to spend some time following the gradient merely to reach the boundary point $\mathbf{a} = \mathbf{0}$. Another problem with J_q is that its value can be dominated by the longest sample vectors. Both of these problems are avoided by the criterion function

$$J_r(\mathbf{a}) = \frac{1}{2} \sum_{\mathbf{y} \in \mathcal{Y}} \frac{(\mathbf{a}^t\mathbf{y} - b)^2}{\|\mathbf{y}\|^2}, \tag{33}$$

where now $\mathcal{Y}(\mathbf{a})$ is the set of samples for which $\mathbf{a}^t\mathbf{y} \leq b$. (If $\mathcal{Y}(\mathbf{a})$ is empty, we define J_r to be zero.) Thus, $J_r(\mathbf{a})$ is never negative, and is zero if and only if $\mathbf{a}^t\mathbf{y} \geq b$ for all of the training samples. The gradient of J_r is given by

$$\boldsymbol{\nabla} J_r = \sum_{\mathbf{y} \in \mathcal{Y}} \frac{\mathbf{a}^t\mathbf{y} - b}{\|\mathbf{y}\|^2}\mathbf{y},$$

and the update rule

$$\left.\begin{array}{ll} \mathbf{a}(1) & \text{arbitrary} \\ \mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k) \sum\limits_{\mathbf{y} \in \mathcal{Y}} \frac{b - \mathbf{a}^t\mathbf{y}}{\|\mathbf{y}\|^2}\mathbf{y}. \end{array}\right\} \tag{34}$$

Thus the relaxation algorithm becomes

Algorithm 8 (Batch relaxation with margin)

```
1  begin initialize $\mathbf{a}, \eta(\cdot), k = 0$
2      do $k \leftarrow k + 1$
3          $\mathcal{Y}_k = \{\}$
4          $j = 0$
5          do $j \leftarrow j + 1$
6              if $\mathbf{y}_j$ is misclassified then Append $\mathbf{y}_j$ to $\mathcal{Y}_k$
7          until $j = n$
8          $\mathbf{a} \leftarrow \mathbf{a} + \eta(k) \sum_{\mathbf{y} \in \mathcal{Y}} \frac{b - \mathbf{a}^t\mathbf{y}}{\|\mathbf{y}\|^2}\mathbf{y}$
9      until $\mathcal{Y}_k = \{\}$
10 return $\mathbf{a}$
11 end
```

As before, we find it easier to prove convergence when the samples are considered one at a time rather than jointly, i.e., single-sample rather than batch. We also limit our attention to the fixed-increment case, $\eta(k) = \eta$. Thus, we are again led to consider a sequence $\mathbf{y}^1, \mathbf{y}^2, ...$ formed from those samples that call for the weight vector to be corrected. The single-sample correction rule analogous to Eq. 33 is

$$\left.\begin{array}{ll} \mathbf{a}(1) & \text{arbitrary} \\ \mathbf{a}(k+1) = \mathbf{a}(k) + \eta \frac{b - \mathbf{a}^t(k)\mathbf{y}^k}{\|\mathbf{y}^k\|^2}\mathbf{y}^k, \end{array}\right\} \tag{35}$$

where $\mathbf{a}^t(k)\mathbf{y}^k \leq b$ for all k. The algorithm is:

Algorithm 9 (Single-sample relaxation with margin)

```
1 begin initialize $\mathbf{a}, \eta(\cdot), k = 0$
2         do $k \leftarrow k + 1$
3             if $\mathbf{y}_k$ is misclassified then $\mathbf{a} \leftarrow \mathbf{a} + \eta(k)\frac{b-\mathbf{a}^t\mathbf{y}}{\|\mathbf{y}_k\|^2}\mathbf{y}_k$
4               until all patterns properly classified
5   return $\mathbf{a}$
6 end
```

This algorithm is known as the *single-sample relaxation rule with margin*, and it has a simple geometrical interpretation. The quantity

$$r(k) = \frac{b - \mathbf{a}^t(k)\mathbf{y}^k}{\|\mathbf{y}^k\|} \tag{36}$$

is the distance from $\mathbf{a}(k)$ to the hyperplane $\mathbf{a}^t\mathbf{y}^k = b$. Since $\mathbf{y}^k/\|\mathbf{y}^k\|$ is the unit normal vector for the hyperplane, Eq. 35 calls for $\mathbf{a}(k)$ to be moved a certain fraction η of the distance from $\mathbf{a}(k)$ to the hyperplane. If $\eta = 1$, $\mathbf{a}(k)$ is moved exactly to the hyperplane, so that the "tension" created by the inequality $\mathbf{a}^t(k)\mathbf{y}^k \leq b$ is "relaxed" (Fig. 5.14). From Eq. 35, after a correction,

$$\mathbf{a}^t(k+1)\mathbf{y}^k - b = (1-\eta)(\mathbf{a}^t(k)\mathbf{y}^k - b). \tag{37}$$

If $\eta < 1$, then $\mathbf{a}^t(k+1)\mathbf{y}^k$ is still less than b, while if $\eta > 1$, then $\mathbf{a}^t(k+1)\mathbf{y}^k$ is greater than b. These conditions are referred to as *underrelaxation* and *overrelaxation*, respectively. In general, we shall restrict η to the range $0 < \eta < 2$ (Figs. 5.14 & 5.15).

UNDER-RELAXATION

OVER-RELAXATION

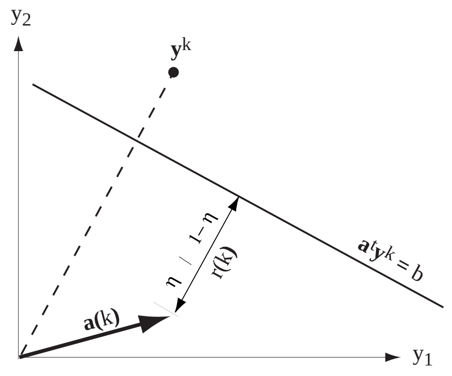


Figure 5.14: In each step of a basic relaxation algorithm, the weight vector is moved a proportion η of the way towards the hyperplane defined by $\mathbf{a}^t\mathbf{y}^k = b$.

5.6.2 Convergence Proof

When the relaxation rule is applied to a set of linearly separable samples, the number of corrections may or may not be finite. If it is finite, then of course we have obtained a solution vector. If it is not finite, we shall see that $\mathbf{a}(k)$ converges to a limit vector on the boundary of the solution region. Since the region in which $\mathbf{a}^t\mathbf{y} \geq b$ is contained in a larger region where $\mathbf{a}^t\mathbf{y} > 0$ if $b > 0$, this implies that $\mathbf{a}(k)$ will enter this larger region at least once, eventually remaining there for all k greater than some finite k_0.

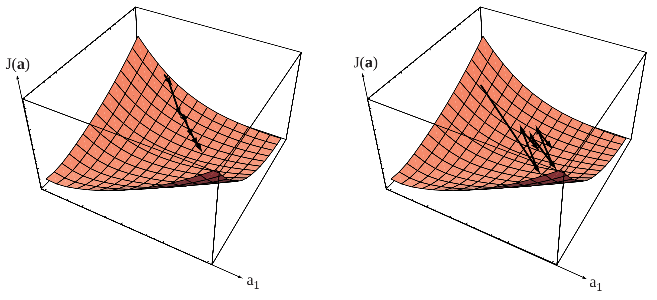


Figure 5.15: At the left, underrelaxation ($\eta < 1$) leads to needlessly slow descent, or even failure to converge. Overrelaxation ($1 < \eta < 2$, shown in the middle) describes overshooting; nevertheless convergence will ultimately be achieved.

The proof depends upon the fact that if $\hat{\mathbf{a}}$ is *any* vector in the solution region — i.e., any vector satisfying $\hat{\mathbf{a}}^t\mathbf{y}_i > b$ for all i — then at each step $\mathbf{a}(k)$ gets closer to $\hat{\mathbf{a}}$. This fact follows at once from Eq. 35, since

$$\begin{aligned}\|\mathbf{a}(k+1)-\hat{\mathbf{a}}\|^2 &= \|\mathbf{a}(k)-\hat{\mathbf{a}}\|^2 - 2\eta\frac{(b-\mathbf{a}^t(k)\mathbf{y}^k)}{\|\mathbf{y}^k\|^2}(\hat{\mathbf{a}}-\mathbf{a}(k))^t\mathbf{y}^k \\ &\quad +\eta^2\frac{(b-\mathbf{a}^t(k)\mathbf{y}^k)^2}{\|\mathbf{y}^k\|^2}\end{aligned} \tag{38}$$

and

$$(\hat{\mathbf{a}}-\mathbf{a}(k))^t\mathbf{y}^k > b-\mathbf{a}^t(k)\mathbf{y}^k \geq 0, \tag{39}$$

so that

$$\|\mathbf{a}(k+1)-\hat{\mathbf{a}}\|^2 \leq \|\mathbf{a}(k)-\hat{\mathbf{a}}\|^2 - \eta(2-\eta)\frac{(b-\mathbf{a}^t(k)\mathbf{y}^k)^2}{\|\mathbf{y}^k\|^2}. \tag{40}$$

Since we restrict η to the range $0 < \eta < 2$, it follows that $\|\mathbf{a}(k+1)-\hat{\mathbf{a}}\| \leq \|\mathbf{a}(k)-\hat{\mathbf{a}}\|$. Thus, the vectors in the sequence $\mathbf{a}(1), \mathbf{a}(2), ...$ get closer and closer to $\hat{\mathbf{a}}$, and in the limit as k goes to infinity the distance $\|\mathbf{a}(k)-\hat{\mathbf{a}}\|$ approaches some limiting distance $r(\hat{\mathbf{a}})$. This means that as k goes to infinity $\mathbf{a}(k)$ is confined to the surface of a hypersphere with center $\hat{\mathbf{a}}$ and radius $r(\hat{\mathbf{a}})$. Since this is true for any $\hat{\mathbf{a}}$ in the solution region, the limiting $\mathbf{a}(k)$ is confined to the intersection of the hyperspheres centered about all of the possible solution vectors.

We now show that the common intersection of these hyperspheres is a single point on the boundary of the solution region. Suppose first that there are at least two points $\mathbf{a}'$ and $\mathbf{a}''$ on the common intersection. Then $\|\mathbf{a}'-\hat{\mathbf{a}}\| = \|\mathbf{a}''-\hat{\mathbf{a}}\|$ for every $\hat{\mathbf{a}}$ in the solution region. But this implies that the solution region is contained in the $(\hat{d}-1)$-dimensional hyperplane of points equidistant from $\mathbf{a}'$ to $\mathbf{a}''$, whereas we know that the solution region is $\hat{d}$-dimensional. (Stated formally, if $\hat{\mathbf{a}}^t\mathbf{y}_i > 0$ for $i = 1, ..., n$, then for any $\hat{d}$-dimensional vector $\mathbf{v}$, we have $(\hat{\mathbf{a}}+\epsilon\mathbf{v})^t\mathbf{y} > 0$ for $i = 1, ..., n$ if ϵ is sufficiently small.) Thus, $\mathbf{a}(k)$ converges to a single point $\mathbf{a}$. This point is certainly not inside the solution region, for then the sequence would be finite. It is not outside either, since each correction causes the weight vector to move η times its distance from the boundary plane, thereby preventing the vector from being bounded away from the boundary forever. Hence the limit point must be on the boundary.

5.7 Nonseparable Behavior

The Perceptron and relaxation procedures give us a number of simple methods for finding a separating vector when the samples are linearly separable. All of these methods are called *error-correcting procedures*, because they call for a modification of the weight vector when and only when an error is encountered. Their success on separable problems is largely due to this relentless search for an error-free solution. In practice, one would only consider the use of these methods if there was reason to believe that the error rate for the optimal linear discriminant function is low.

ERROR-CORRECTING PROCEDURE

Of course, even if a separating vector is found for the training samples, it does not follow that the resulting classifier will perform well on independent test data. A moment's reflection will show that *any* set of fewer than $2\hat{d}$ samples is likely to be linearly separable — a matter we shall return to in Chap. **??**. Thus, one should use several times that many design samples to overdetermine the classifier, thereby ensuring that the performance on training and test data will be similar. Unfortunately, sufficiently large design sets are almost certainly *not* linearly separable. This makes it important to know how the error-correction procedures will behave when the samples are nonseparable.

Since no weight vector can correctly classify every sample in a nonseparable set (by definition), it is clear that the corrections in an error-correction procedure can never cease. Each algorithm produces an infinite sequence of weight vectors, any member of which may or may not yield a useful "solution." The exact nonseparable behavior of these rules has been studied thoroughly in a few special cases. It is known, for example, that the length of the weight vectors produced by the fixed-increment rule are bounded. Empirical rules for terminating the correction procedure are often based on this tendency for the length of the weight vector to fluctuate near some limiting value. From a theoretical viewpoint, if the components of the samples are integer-valued, the fixed-increment procedure yields a finite-state process. If the correction process is terminated at some arbitrary point, the weight vector may or may not be in a good state. By averaging the weight vectors produced by the correction rule, one can reduce the risk of obtaining a bad solution by accidentally choosing an unfortunate termination time.

A number of similar heuristic modifications to the error-correction rules have been suggested and studied empirically. The goal of these modifications is to obtain acceptable performance on nonseparable problems while preserving the ability to find a separating vector on separable problems. A common suggestion is the use of a variable increment $\eta(k)$, with $\eta(k)$ approaching zero as k approaches infinity. The rate at which $\eta(k)$ approaches zero is quite important. If it is too slow, the results will still be sensitive to those training samples that render the set nonseparable. If it is too fast, the weight vector may converge prematurely with less than optimal results. One way to choose $\eta(k)$ is to make it a function of recent performance, decreasing it as performance improves. Another way is to program $\eta(k)$ by a choice such as $\eta(k) = \eta(1)/k$. When we examine stochastic approximation techniques, we shall see that this latter choice is the theoretical solution to an analogous problem. Before we take up this topic, however, we shall consider an approach that sacrifices the ability to obtain a separating vector for good compromise performance on both separable and nonseparable problems.

5.8 Minimum Squared Error Procedures

5.8.1 Minimum Squared Error and the Pseudoinverse

The criterion functions we have considered thus far have focussed their attention on the misclassified samples. We shall now consider a criterion function that involves *all* of the samples. Where previously we have sought a weight vector $\mathbf{a}$ making all of the inner products $\mathbf{a}^t\mathbf{y}_i$ positive, now we shall try to make $\mathbf{a}^t\mathbf{y}_i = b_i$, where the b_i are some arbitrarily specified positive constants. Thus, we have replaced the problem of finding the solution to a set of linear inequalities with the more stringent but better understood problem of finding the solution to a set of linear equations.

The treatment of simultaneous linear equations is simplified by introducing matrix notation. Let $\mathbf{Y}$ be the n-by-$\hat{d}$ matrix ($\hat{d} = d+1$) whose ith row is the vector $\mathbf{y}_i^t$, and let $\mathbf{b}$ be the column vector $\mathbf{b} = (b_1, ..., b_n)^t$. Then our problem is to find a weight vector $\mathbf{a}$ satisfying

$$\begin{pmatrix} Y_{10} & Y_{11} & \cdots & Y_{1d} \\ Y_{20} & Y_{21} & \cdots & Y_{2d} \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ Y_{n0} & Y_{n1} & \cdots & Y_{nd} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_d \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ \vdots \\ b_n \end{pmatrix}$$

$$\text{or} \qquad \mathbf{Ya} = \mathbf{b}. \tag{41}$$

If $\mathbf{Y}$ were nonsingular, we could write $\mathbf{a} = \mathbf{Y}^{-1}\mathbf{b}$ and obtain a formal solution at once. However, $\mathbf{Y}$ is rectangular, usually with more rows than columns. When there are more equations than unknowns, $\mathbf{a}$ is overdetermined, and ordinarily no exact solution exists. However, we can seek a weight vector $\mathbf{a}$ that minimizes some function of the error between $\mathbf{Ya}$ and $\mathbf{b}$. If we define the error vector $\mathbf{e}$ by

$$\mathbf{e} = \mathbf{Ya} - \mathbf{b} \tag{42}$$

then one approach is to try to minimize the squared length of the error vector. This is equivalent to minimizing the sum-of-squared-error criterion function

$$J_s(\mathbf{a}) = \|\mathbf{Ya} - \mathbf{b}\|^2 = \sum_{i=1}^{n} (\mathbf{a}^t\mathbf{y}_i - b_i)^2. \tag{43}$$

The problem of minimizing the sum of squared error is a classical one. It can be solved by a gradient search procedure, as we shall see in Sect. **??**. A simple closed-form solution can also be found by forming the gradient

$$\boldsymbol{\nabla} J_s = \sum_{i=1}^{n} 2(\mathbf{a}^t\mathbf{y}_i - b_i)\mathbf{y}_i = 2\mathbf{Y}^t(\mathbf{Ya} - \mathbf{b}) \tag{44}$$

and setting it equal to zero. This yields the necessary condition

$$\mathbf{Y}^t\mathbf{Y}\mathbf{a} = \mathbf{Y}^t\mathbf{b}, \tag{45}$$

and in this way we have converted the problem of solving $\mathbf{Ya} = \mathbf{b}$ to that of solving $\mathbf{Y}^t\mathbf{Ya} = \mathbf{Y}^t\mathbf{b}$. This celebrated equation has the great advantage that the $\hat{d}$-by-$\hat{d}$ matrix $\mathbf{Y}^t\mathbf{Y}$ is square and often nonsingular. If it is nonsingular, we can solve for $\mathbf{a}$ uniquely as

$$\begin{aligned}\mathbf{a} &= (\mathbf{Y}^t\mathbf{Y})^{-1}\mathbf{Y}^t\mathbf{b}\\ &= \mathbf{Y}^\dagger\mathbf{b},\end{aligned} \tag{46}$$

where the $\hat{d}$-by-n matrix

$$\mathbf{Y}^\dagger \equiv (\mathbf{Y}^t\mathbf{Y})^{-1}\mathbf{Y}^t \tag{47}$$

is called the *pseudoinverse* of $\mathbf{Y}$. Note that if $\mathbf{Y}$ is square and nonsingular, the pseudoinverse coincides with the regular inverse. Note also that $\mathbf{Y}^\dagger\mathbf{Y} = \mathbf{I}$, but $\mathbf{YY}^\dagger \neq \mathbf{I}$ in general. However, a minimum-squared-error (MSE) solution always exists. In particular, if $\mathbf{Y}^\dagger$ is defined more generally by

PSEUDO-INVERSE

$$\mathbf{Y}^\dagger \equiv \lim_{\epsilon\to 0}(\mathbf{Y}^t\mathbf{Y} + \epsilon\mathbf{I})^{-1}\mathbf{Y}^t, \tag{48}$$

it can be shown that this limit always exists, and that $\mathbf{a} = \mathbf{Y}^\dagger\mathbf{b}$ is an MSE solution to $\mathbf{Ya} = \mathbf{b}$.

The MSE solution depends on the margin vector $\mathbf{b}$, and we shall see that different choices for $\mathbf{b}$ give the solution different properties. If $\mathbf{b}$ is fixed arbitrarily, there is no reason to believe that the MSE solution yields a separating vector in the linearly separable case. However, it is reasonable to hope that by minimizing the squared-error criterion function we might obtain a useful discriminant function in both the separable and the nonseparable cases. We shall now examine two properties of the solution that support this hope.

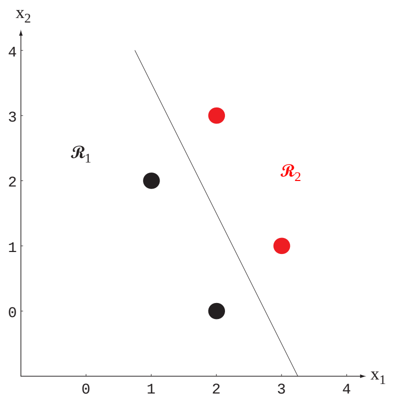
Example 1: Constructing a linear classifier by matrix pseudoinverse

Suppose we have the following two-dimensional points for two categories: ω_1: $(1,2)^t$ and $(2,0)^t$, and ω_2: $(3,1)^t$ and $(2,3)^t$, as shown in black and red, respectively, in the figure.
Our matrix $\mathbf{Y}$ is therefore

$$\mathbf{Y} = \begin{pmatrix} 1 & 1 & 2\\ 1 & 2 & 0\\ -1 & -3 & -1\\ -1 & -2 & -3 \end{pmatrix}$$

and after a few simple calculations we find that its pseudoinverse is

$$\mathbf{Y}^\dagger \equiv \lim_{\epsilon\to 0}(\mathbf{Y}^t\mathbf{Y} + \epsilon\mathbf{I})^{-1}\mathbf{Y}^t = \begin{pmatrix} 5/4 & 13/12 & 3/4 & 7/12\\ -1/2 & -1/6 & -1/2 & -1/6\\ 0 & -1/3 & 0 & -1/3 \end{pmatrix}$$



Four training points and the decision boundary $\mathbf{a}^t \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix} = 0$, where $\mathbf{a}$ was found by means of a pseudoinverse technique.

We arbitrarily let all the margins be equal, i.e., $\mathbf{b} = (1,1,1,1)^t$. Our solution is $\mathbf{a} = \mathbf{Y}^\dagger\mathbf{b} = (11/3, -4/3, -2/3)^t$, and leads to the decision boundary shown in the figure. Other choices for $\mathbf{b}$ would typically lead to different decision boundaries, of course.

5.8.2 Relation to Fisher's Linear Discriminant

In this section we shall show that with the proper choice of the vector $\mathbf{b}$, the MSE discriminant function $\mathbf{a}^t\mathbf{y}$ is directly related to Fisher's linear discriminant. To do this, we must return to the use of linear rather than generalized linear discriminant functions. We assume that we have a set of n d-dimensional samples $\mathbf{x}_1, ..., \mathbf{x}_n$, n_1 of which are in the subset $\mathcal{D}_1$ labelled ω_1, and n_2 of which are in the subset $\mathcal{D}_2$ labelled ω_2. Further, we assume that a sample $\mathbf{y}_i$ is formed from $\mathbf{x}_i$ by adding a threshold component $x_0 = 1$ to make an *augmented pattern vector*. Further, if the sample is labelled ω_2, then the entire pattern vector is multiplied by -1 — the "normlization" we saw in Sect. 5.4.1. With no loss in generality, we can assume that the first n_1 samples are labelled ω_1 and the second n_2 are labelled ω_2. Then the matrix $\mathbf{Y}$ can be partitioned as follows:

AUGMENTED PATTERN VECTOR

$$\mathbf{Y} = \begin{bmatrix} \mathbf{1}_1 & \mathbf{X}_1 \\ -\mathbf{1}_2 & -\mathbf{X}_2 \end{bmatrix},$$

where $\mathbf{1}_i$ is a column vector of n_i ones, and $\mathbf{X}_i$ is an n_i-by-d matrix whose rows are the samples labelled ω_i. We partition $\mathbf{a}$ and $\mathbf{b}$ correspondingly, with

$$\mathbf{a} = \begin{bmatrix} w_0 \\ \mathbf{w} \end{bmatrix}$$

and with

$$\mathbf{b} = \begin{bmatrix} \frac{n}{n_1}\mathbf{1}_1 \\ \frac{n}{n_2}\mathbf{1}_2 \end{bmatrix}.$$

We shall now show that this special choice for **b** links the MSE solution to Fisher's linear discriminant.

We begin by writing Eq. 47 for **a** in terms of the partitioned matrices:

$$\begin{bmatrix} \mathbf{1}_1^t & -\mathbf{1}_2^t \\ \mathbf{X}_1^t & -\mathbf{X}_2^t \end{bmatrix} \begin{bmatrix} \mathbf{1}_1 & \mathbf{X}_1 \\ -\mathbf{1}_2 & -\mathbf{X}_2 \end{bmatrix} \begin{bmatrix} w_0 \\ \mathbf{w} \end{bmatrix} = \begin{bmatrix} \mathbf{1}_1^t & -\mathbf{1}_2^t \\ \mathbf{X}_1^t & -\mathbf{X}_2^t \end{bmatrix} \begin{bmatrix} \frac{n}{n_1}\mathbf{1}_1 \\ \frac{n}{n_2}\mathbf{1}_2 \end{bmatrix}. \qquad (49)$$

By defining the sample means $\mathbf{m}_i$ and the pooled sample scatter matrix $\mathbf{S}_W$ as

$$\mathbf{m}_i = \frac{1}{n_i}\sum_{\mathbf{x}\in\mathcal{D}_i}\mathbf{x} \qquad i = 1, 2 \qquad (50)$$

and

$$\mathbf{S}_W = \sum_{i=1}^{2}\sum_{\mathbf{x}\in\mathcal{D}_i}(\mathbf{x}-\mathbf{m}_i)(\mathbf{x}-\mathbf{m}_i)^t, \qquad (51)$$

we can multiply the matrices of Eq. 49 and obtain

$$\begin{bmatrix} n & (n_1\mathbf{m}_1 + n_2\mathbf{m}_2)^t \\ (n_1\mathbf{m}_1 + n_2\mathbf{m}_2) & \mathbf{S}_W + n_1\mathbf{m}_1\mathbf{m}_1^t + n_2\mathbf{m}_2\mathbf{m}_2^t \end{bmatrix} \begin{bmatrix} w_0 \\ \mathbf{w} \end{bmatrix} = \begin{bmatrix} 0 \\ n(\mathbf{m}_1 - \mathbf{m}_2) \end{bmatrix}.$$

This can be viewed as a pair of equations, the first of which can be solved for w_0 in terms of **w**:

$$w_0 = -\mathbf{m}^t\mathbf{w}, \qquad (52)$$

where **m** is the mean of all of the samples. Substituting this in the second equation and performing a few algebraic manipulations, we obtain

$$\Big[\frac{1}{n}\mathbf{S}_W + \frac{n_1 n_2}{n^2}(\mathbf{m}_1-\mathbf{m}_2)(\mathbf{m}_1-\mathbf{m}_2)^t\Big]\mathbf{w} = \mathbf{m}_1 - \mathbf{m}_2. \qquad (53)$$

Since the vector $(\mathbf{m}_1-\mathbf{m}_2)(\mathbf{m}_1-\mathbf{m}_2)^t\mathbf{w}$ is in the direction of $\mathbf{m}_1 - \mathbf{m}_2$ for any value of **w**, we can write

$$\frac{n_1 n_2}{n^2}(\mathbf{m}_1-\mathbf{m}_2)(\mathbf{m}_1-\mathbf{m}_2)^t\mathbf{w} = (1-\alpha)(\mathbf{m}_1-\mathbf{m}_2),$$

where α is some scalar. Then Eq. 53 yields

$$\mathbf{w} = \alpha n \mathbf{S}_W^{-1}(\mathbf{m}_1 - \mathbf{m}_2), \qquad (54)$$

which, except for an unimportant scale factor, is identical to the solution for Fisher's linear discriminant. In addition, we obtain the threshold weight w_0 and the following decision rule: Decide ω_1 if $\mathbf{w}^t(\mathbf{x}-\mathbf{m}) > 0$; otherwise decide ω_2.

5.8.3 Asymptotic Approximation to an Optimal Discriminant

Another property of the MSE solution that recommends its use is that if $\mathbf{b} = \mathbf{1}_n$ it approaches a minimum mean-squared-error approximation to the Bayes discriminant function

$$g_0(\mathbf{x}) = P(\omega_1|\mathbf{x}) - P(\omega_2|\mathbf{x}) \tag{55}$$

in the limit as the number of samples approaches infinity. To demonstrate this fact, we must assume that the samples are drawn independently, identically distributed (i.i.d.) according to the probability law

$$p(\mathbf{x}) = p(\mathbf{x}|\omega_1)P(\omega_1) + p(\mathbf{x}|\omega_2)P(\omega_2). \tag{56}$$

In terms of the augmented vector $\mathbf{y}$, the MSE solution yields the series expansion $g(\mathbf{x}) = \mathbf{a}^t\mathbf{y}$, where $\mathbf{y} = \mathbf{y}(\mathbf{x})$. If we define the mean-squared approximation error by

$$\epsilon^2 = \int [\mathbf{a}^t\mathbf{y} - g_0(\mathbf{x})]^2 p(\mathbf{x})\, d\mathbf{x}, \tag{57}$$

then our goal is to show that ϵ^2 is minimized by the solution $\mathbf{a} = \mathbf{Y}^\dagger \mathbf{1}_n$.

The proof is simplified if we preserve the distinction between category ω_1 and category ω_2 samples. In terms of the unnormalized data, the criterion function J_s becomes

$$\begin{aligned} J_s(\mathbf{a}) &= \sum_{\mathbf{y}\in\mathcal{Y}_1} (\mathbf{a}^t\mathbf{y} - 1)^2 + \sum_{\mathbf{y}\in\mathcal{Y}_2} (\mathbf{a}^t\mathbf{y} + 1)^2 \\ &= n\Big[\frac{n_1}{n}\frac{1}{n_1}\sum_{\mathbf{y}\in\mathcal{Y}_1} (\mathbf{a}^t\mathbf{y} - 1)^2 + \frac{n_2}{n}\frac{1}{n_2}\sum_{\mathbf{y}\in\mathcal{Y}_2} (\mathbf{a}^t\mathbf{y} + 1)^2\Big]. \end{aligned} \tag{58}$$

Thus, by the law of large numbers, as n approaches infinity $(1/n)J_s(\mathbf{a})$ approaches

$$\bar{J}(\mathbf{a}) = P(\omega_1)\mathcal{E}_1[(\mathbf{a}^t\mathbf{y} - 1)^2] + P(\omega_2)\mathcal{E}_2[(\mathbf{a}^t\mathbf{y} + 1)^2], \tag{59}$$

with probability one, where

$$\mathcal{E}_1[(\mathbf{a}^t\mathbf{y} - 1)^2] = \int (\mathbf{a}^t\mathbf{y} - 1)^2 p(\mathbf{x}|\omega_1)\, d\mathbf{x}$$

and

$$\mathcal{E}_2[(\mathbf{a}^t\mathbf{y} + 1)^2] = \int (\mathbf{a}^t\mathbf{y} + 1)^2 p(\mathbf{x}|\omega_2)\, d\mathbf{x}.$$

Now, if we recognize from Eq. 55 that

$$g_0(\mathbf{x}) = \frac{p(\mathbf{x}, \omega_1) - p(\mathbf{x}, \omega_2)}{p(\mathbf{x})}$$

we see that

$$\begin{aligned}\bar{J}(\mathbf{a}) &= \int (\mathbf{a}^t\mathbf{y}-1)^2 p(\mathbf{x},\omega_1)\ d\mathbf{x} + \int (\mathbf{a}^t\mathbf{y}+1)^2 p(\mathbf{x},\omega_2)\ d\mathbf{x} \\ &= \int (\mathbf{a}^t\mathbf{y})^2 p(\mathbf{x})\ d\mathbf{x} - 2\int \mathbf{a}^t\mathbf{y} g_0(\mathbf{x}) p(\mathbf{x})\ d\mathbf{x} + 1 \\ &= \underbrace{\int [\mathbf{a}^t\mathbf{y} - g_0(\mathbf{x})]^2 p(\mathbf{x})\ d\mathbf{x}}_{\epsilon^2} + \underbrace{\left[1 - \int g_0^2(\mathbf{x}) p(\mathbf{x})\ d\mathbf{x}\right]}_{\text{indep. of } \mathbf{a}}. \end{aligned} \tag{60}$$

The second term in this sum is independent of the weight vector **a**. Hence, the **a** that minimizes J_s also minimizes ϵ^2 — the mean-squared-error between $\mathbf{a}^t\mathbf{y}$ and $g(\mathbf{x})$ (Fig. 5.16). In Chap. **??** we shall see that analogous properties also holds for many multilayer networks.

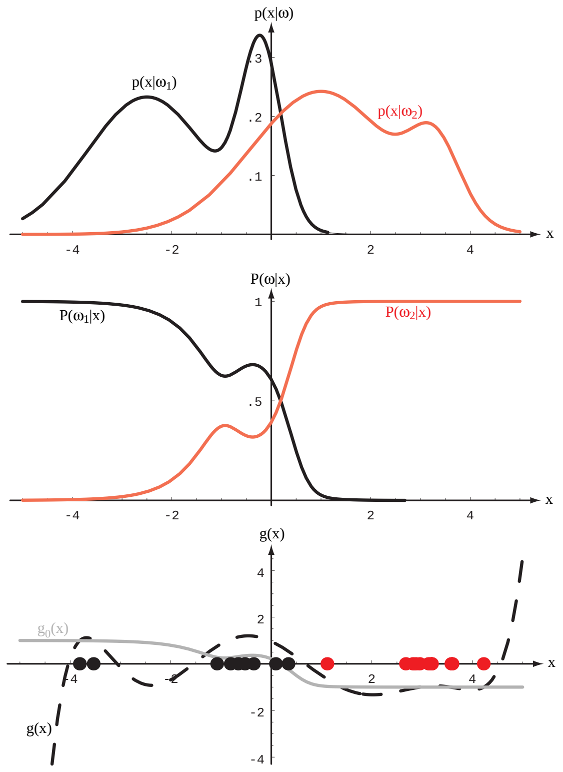


Figure 5.16: The top figure shows two class-conditional densities, and the middle figure the posteriors, assuming equal priors. Minimizing the MSE error also minimizes the mean-squared-error between $\mathbf{a}^t\mathbf{y}$ and the discriminant function $g(\mathbf{x})$ (here a 7th-order polynomial) measured over the data distribution, as shown at the bottom. Note that the resulting $g(x)$ best approximates $g_0(x)$ in the regions where the data points lie.

This result gives considerable insight into the MSE procedure. By approximating $g_0(\mathbf{x})$, the discriminant function $\mathbf{a}^t\mathbf{y}$ gives direct information about the posterior probabilities $P(\omega_1|\mathbf{x}) = (1+g_0)/2$ and $P(\omega_2|\mathbf{x}) = (1-g_0)/2$. The quality of the approximation depends on the functions $y_i(\mathbf{x})$ and the number of terms in the expansion $\mathbf{a}^t\mathbf{y}$. Unfortunately, the mean-square-error criterion places emphasis on points where $p(\mathbf{x})$ is larger, rather than on points near the decision surface $g_0(\mathbf{x}) = 0$. Thus, the discriminant function that "best" approximates the Bayes discriminant does not necessarily minimize the probability of error. Despite this property, the MSE solution has interesting properties, and has received considerable attention in the literature. We shall encounter the mean-square approximation of $g_0(\mathbf{x})$ again when we consider stochastic approximation methods and multilayer neural networks.

5.8.4 The Widrow-Hoff Procedure

We remarked earlier that $J_s(\mathbf{a}) = \|\mathbf{Ya} - \mathbf{b}\|^2$ could be minimized by a gradient descent procedure. Such an approach has two advantages over merely computing the pseudoinverse: (1) it avoids the problems that arise when $\mathbf{Y}^t\mathbf{Y}$ is singular, and (2) it avoids the need for working with large matrices. In addition, the computation involved is effectively a feedback scheme which automatically copes with some of the computational problems due to roundoff or truncation. Since $\boldsymbol{\nabla} J_s = 2\mathbf{Y}^t(\mathbf{Ya} - \mathbf{b})$, the obvious update rule is

$$\left.\begin{array}{l} \mathbf{a}(1) \qquad \text{arbitrary} \\ \mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k)\mathbf{Y}^t(\mathbf{Ya}_k - \mathbf{b}). \end{array}\right\}$$

In Problem 24 you are asked to show that if $\eta(k) = \eta(1)/k$, where $\eta(1)$ is any positive constant, then this rule generates a sequence of weight vectors that converges to a limiting vector $\mathbf{a}$ satisfying

$$\mathbf{Y}^t(\mathbf{Ya} - \mathbf{b}) = 0.$$

Thus, the descent algorithm always yields a solution regardless of whether or not $\mathbf{Y}^t\mathbf{Y}$ is singular.

While the $\hat{d}$-by-$\hat{d}$ matrix $\mathbf{Y}^t\mathbf{Y}$ is usually smaller than the $\hat{d}$-by-n matrix $\mathbf{Y}^\dagger$, the storage requirements can be reduced still further by considering the samples sequentially and using the *Widrow-Hoff* or *LMS rule* (least-mean-squared):

LMS RULE

$$\left.\begin{array}{l} \mathbf{a}(1) \qquad \text{arbitrary} \\ \mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k)(b_k - \mathbf{a}(k)^t\mathbf{y}^k)\mathbf{y}^k, \end{array}\right\} \qquad (61)$$

or in algorithm form:

Algorithm 10 (LMS)

```
1 begin initialize $\mathbf{a}, \mathbf{b}$, criterion $\theta, \eta(\cdot), k = 0$
2         do $k \leftarrow k + 1$
3             $\mathbf{a} \leftarrow \mathbf{a} + \eta(k)(b_k - \mathbf{a}^t\mathbf{y}^k)\mathbf{y}^k$
4     until $\eta(k)(b_k - \mathbf{a}^t\mathbf{y}^k)\mathbf{y}^k < \theta$
5     return $\mathbf{a}$
6 end
```

At first glance this descent algorithm appears to be essentially the same as the relaxation rule. The primary difference is that the relaxation rule is an error-correction rule, so that $\mathbf{a}^t(k)\mathbf{y}^k$ does not equal b_k, and thus the corrections never cease. Therefore, $\eta(k)$ must decrease with k to obtain convergence, the choice $\eta(k) = \eta(1)/k$ being common. Exact analysis of the behavior of the Widrow-Hoff rule in the deterministic case is rather complicated, and merely indicates that the sequence of weight vectors tends to converge to the desired solution. Instead of pursuing this topic further, we shall turn to a very similar rule that arises from a stochastic descent procedure. We note, however, that the solution need not give a separating vector, even if one exists, as shown in Fig. 5.17 (Computer exercise 10).

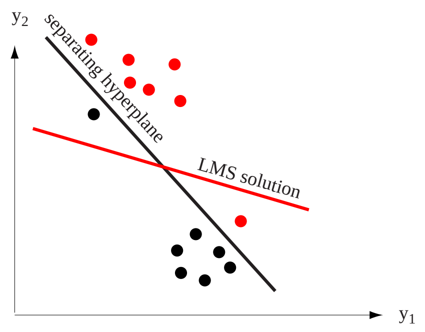


Figure 5.17: The LMS algorithm need not converge to a separating hyperplane, even if one exists. Since the LMS solution minimizes the sum of the squares of the distances of the training points to the hyperplane, for this exmple the plane is rotated clockwise compared to a separating hyperplane.

5.8.5 Stochastic Approximation Methods

All of the iterative descent procedures we have considered thus far have been described in deterministic terms. We are given a particular set of samples, and we generate a particular sequence of weight vectors. In this section we digress briefly to consider an MSE procedure in which the samples are drawn randomly, resulting in a random sequence of weight vectors. We will return in Chap. **??** to the theory of stochastic approximation though here some of the main ideas will be presented without proof.

Suppose that samples are drawn independently by selecting a state of nature with probability $P(\omega_i)$ and then selecting an $\mathbf{x}$ according to the probability law $p(\mathbf{x}|\omega_i)$. For each $\mathbf{x}$ we let θ be its *label*, with $\theta = +1$ if $\mathbf{x}$ is labelled ω_1 and $\theta = -1$ if $\mathbf{x}$ is labelled ω_2. Then the data consist of an infinite sequence of independent pairs $(\mathbf{x}, \theta_1)$, $(\mathbf{x}_2, \theta_2), ...,$ $(\mathbf{x}_k, \theta_k), ...$ Even though the label variable θ is binary-valued it can be thought of as a noisy version of the Bayes discriminant function $g_0(\mathbf{x})$. This follows from the observation that

$$P(\theta = 1|\mathbf{x}) = P(\omega_1|\mathbf{x}),$$

and

$$P(\theta = -1|\mathbf{x}) = P(\omega_2|\mathbf{x}),$$

so that the conditional mean of θ is given by

$$\mathcal{E}_{\theta|\mathbf{x}}[\theta] = \sum_{\theta} \theta P(\theta|\mathbf{x}) = P(\omega_1|\mathbf{x}) - P(\omega_2|\mathbf{x}) = g_0(\mathbf{x}). \qquad (62)$$

Suppose that we wish to approximate $g_0(\mathbf{x})$ by the finite series expansion

$$g(\mathbf{x}) = \mathbf{a}^t\mathbf{y} = \sum_{i=1}^{\hat{d}} a_i y_i(\mathbf{x}),$$

where both the basis functions $y_i(\mathbf{x})$ and the number of terms $\hat{d}$ are known. Then we can seek a weight vector $\hat{\mathbf{a}}$ that minimizes the mean-squared approximation error

$$\epsilon^2 = \mathcal{E}[(\mathbf{a}^t\mathbf{y} - g_0(\mathbf{x}))^2]. \tag{63}$$

Minimization of ϵ^2 would appear to require knowledge of Bayes discriminant $g_0(\mathbf{x})$. However, as one might have guessed from the analogous situation in Sect. 5.8.3, it can be shown that the weight vector $\hat{\mathbf{a}}$ that minimizes ϵ^2 also minimizes the criterion function

$$J_m(\mathbf{a}) = \mathcal{E}[(\mathbf{a}^t\mathbf{y} - \theta)^2]. \tag{64}$$

This should also be plausible from the fact that θ is essentially a noisy version of $g_0(\mathbf{x})$ (Fig. **??**). Since the gradient is

$$\boldsymbol{\nabla} J_m = 2\mathcal{E}[(\mathbf{a}^t\mathbf{y} - \theta)\mathbf{y}], \tag{65}$$

we can obtain the closed-form solution

$$\hat{\mathbf{a}} = \mathcal{E}[\mathbf{y}\mathbf{y}^t]^{-1}\mathcal{E}[\theta\mathbf{y}]. \tag{66}$$

Thus, one way to use the samples is to estimate $\mathcal{E}[\mathbf{y}\mathbf{y}^t]$ and $\mathcal{E}[\theta\mathbf{y}]$, and use Eq. 66 to obtain the MSE optimal linear discriminant. An alternative is to minimize $J_m(\mathbf{a})$ by a gradient descent procedure. Suppose that in place of the true gradient we substitute the noisy version $2(\mathbf{a}^t\mathbf{y}_k - \theta_k)\mathbf{y}_k$. This leads to the update rule

$$\mathbf{a}(k+1) = \mathbf{a}(k) + \eta(\theta_k - \mathbf{a}^t(k)\mathbf{y}_k)\mathbf{y}_k, \tag{67}$$

which is basically just the Widrow-Hoff rule. It can be shown (Problem **??**) that if $\mathcal{E}[\mathbf{y}\mathbf{y}^t]$ is nonsingular and if the coefficients $\eta(k)$ satisfy

$$\lim_{m\to\infty} \sum_{k=1}^{m} \eta(k) = +\infty \tag{68}$$

and

$$\lim_{m\to\infty} \sum_{k=1}^{m} \eta^2(k) < \infty \tag{69}$$

then $\mathbf{a}(k)$ converges to $\hat{\mathbf{a}}$ in mean square:

$$\lim_{k\to\infty} \mathcal{E}[\|\mathbf{a}(k) - \hat{\mathbf{a}}\|^2] = 0. \tag{70}$$

The reasons we need these conditions on $\eta(k)$ are simple. The first condition keeps the weight vector from converging so fast that a systematic error will remain forever uncorrected. The second condition ensures that random fluctuations are eventually suppressed. Both conditions are satisfied by the conventional choice $\eta(k) = 1/k$.

Unfortunately, this kind of programmed decrease of $\eta(k)$, independent of the problem at hand, often leads to very slow convergence.

Of course, this is neither the only nor the best descent algorithm for minimizing J_m. For example, if we note that the matrix of second partial derivatives for J_m is given by

$$D = 2\mathcal{E}[\mathbf{y}\mathbf{y}^t],$$

we see that Newton's rule for minimizing J_m (Eq. 15) is

$$\mathbf{a}(k+1) = \mathbf{a}(k) + \mathcal{E}[\mathbf{y}\mathbf{y}^t]^{-1}\mathcal{E}[(\theta - \mathbf{a}^t\mathbf{y})\mathbf{y}].$$

A stochastic analog of this rule is

$$\mathbf{a}(k+1) = \mathbf{a}(k) + \mathbf{R}_{k+1}(\theta_k - \mathbf{a}^t(k)\mathbf{y}_k)\mathbf{y}_k. \tag{71}$$

with

$$\mathbf{R}_{k+1}^{-1} = \mathbf{R}_k^{-1} + \mathbf{y}_k\mathbf{y}_k^t, \tag{72}$$

or, equivalently,[*]

$$\mathbf{R}_{k+1} = \mathbf{R}_k - \frac{\mathbf{R}_k\mathbf{y}_k(\mathbf{R}_k\mathbf{y}_k)^t}{1 + \mathbf{y}_k^t\mathbf{R}_k\mathbf{y}_k}. \tag{73}$$

This rule also produces a sequence of weight vectors that converges to the optimal solution in mean square. Its convergence is faster, but it requires more computation per step (Computer exercise 8).

These gradient procedures can be viewed as methods for minimizing a criterion function, or finding the zero of its gradient, in the presence of noise. In the statistical literature, functions such as J_m and ∇J_m that have the form $\mathcal{E}[f(\mathbf{a}, \mathbf{x})]$ are called *regression functions*, and the iterative algorithms are called *stochastic approximation procedures.* Two well known ones are the Kiefer-Wolfowitz procedure for minimizing a regression function, and the Robbins-Monro procedure for finding a root of a regression function. Often the easiest way to obtain a convergence proof for a particular descent or approximation procedure is to show that it satisfies the convergence conditions for these more general procedures. Unfortunately, an exposition of these methods in their full generality would lead us rather far afield, and we must close this digression by referring the interested reader to the literature.

REGRESSION FUNCTION

STOCHASTIC APPROXI-MATION

5.9 The Ho-Kashyap Procedures

5.9.1 The Descent Procedure

The procedures we have considered thus far differ in several ways. The Perceptron and relaxation procedures find separating vectors if the samples are linearly separable, but do not converge on nonseparable problems. The MSE procedures yield a weight vector whether the samples are linearly separable or not, but there is no guarantee

[*] This recursive formula for computing R_k, which is roughly $(1/k)\mathcal{E}[\mathbf{y}\mathbf{y}^t]^{-1}$, cannot be used if R_k is singular. The equivalence of Eq. 72 and Eq. 73 follows from Problem **??** of Chap. **??**.