CHAPTER 4

Single-Unit Mappings and the Perceptron

We should not introduce errors through sloppiness; we should do it carefully and systematically.

Edsger Dijkstra

4.1 INTRODUCTION

In Chapter 3, the characteristics of a single unit were considered. In this chapter, we continue to study unit-specific computation, as a prelude to understanding larger (multi-unit) networks. Specifically, we consider the utility of a single element in implementing useful mappings. We begin with a linear unit and then extend the analysis to the nonlinear case. From the analysis of the simple, single-unit ANNs of this chapter, a great deal of information extendible to more sophisticated structures can be gained. In Chapters 5 and 6 the concept is extended to layers of units and multilayer structures. We emphasize the early work of Widrow [WH60] in single-element, nonlinear adaptive systems such as Adaline. Much of the Adaline presentation in this chapter follows [WL90].

4.2 LINEAR SEPARABILITY

In this section we explore the concept and utility of *linearly separable input point sets* and related solution procedures. We begin by considering a single-unit mapping that separates a set of d-dimensional inputs $\{i_1, i_2, \ldots, i_d\}$ into two distinct outputs. The input samples of a $c = 2$ class training set are visualized as points in R^d. Each point is represented as a $d \times 1$ vector, $\underline{i}$.[1] Note that some configurations of input vectors are separable by a (possibly nonunique) hyperplane. Although this is not true for an arbitrary configuration of input samples (this is considered in depth in the following discussion), the computational and conceptual advantages of a linear decision boundary often

[1] Or $\underset{\sim}{i}^{d\times 1}$ to emphasize the vector dimension.

motivate us to consider it as a special case of a mapping network. Perhaps the single most important point of this chapter is that a class of unit that forms its net activation using a weighted linear combination of inputs (usually followed by a hardlimiter activation function) implements a hyperplanar decision boundary. Therefore, it is restricted in applicability to problems that are "linearly separable." This is a problem that has plagued pattern recognition and early ANN development efforts. A priori knowledge that a problem is linearly separable thus makes the choice of an ANN architecture straightforward.

DEFINITION. If a linear decision boundary (hyperplanar decision boundary) exists that correctly classifies all the training samples in H for a $c = 2$ class problem, the samples are said to be *linearly separable* [Sch92].

The hyperplane, denoted H_{ij}, is defined by parameters $\underline{w}$ and w_o in a linear constraint of the form

$$g(\underline{i}) = \underline{w}^T \underline{i} - w_o = 0 \tag{4.1}$$

$g(\underline{i})$ separates R^d into positive and negative regions R_p and R_n, where

$$g(\underline{i}) = \underline{w}^T \underline{i} - w_o = \begin{cases} > 0 & \underline{i} \in R_p \\ 0 & \underline{i} \in H_{ij} \\ < 0 & \underline{i} \in R_n \end{cases} \tag{4.2}$$

Problems that are not linearly separable are sometimes referred to as *nonlinearly separable* or *topologically complex*.

LEMMA 4.1. If the n training samples of a $c = 2$ class problem in H are linearly separable by a hyperplane in d-dimensional space R^d, a single-layer feedforward network is capable of correctly classifying the samples.

Recall from Chapter 2 that a linear decision *boundary* is defined by a hyperplane through point $\underline{i}_s$:

$$\underline{w}^T (\underline{i} - \underline{i}_s) = 0 \tag{4.3}$$

Rearranging gives

$$\underline{w}^T \underline{i} + b = 0 \tag{4.4}$$

or

$$\hat{\underline{w}}^T \hat{\underline{i}} = 0 \tag{4.5}$$

with the homogeneous vectors

$$\hat{\underline{w}} = \begin{pmatrix} \underline{w} \\ b \end{pmatrix} \tag{4.6}$$

and

$$\hat{\underline{i}} = \begin{pmatrix} \underline{i} \\ 1 \end{pmatrix} \tag{4.7}$$

$\hat{\underline{w}}$ is the separating or solution vector determining the orientation ($\underline{w}$) and location (b) of a hyperplane in d-space.

4.2.1 Probability of a Linearly Separable Problem

As shown in [Cov65], of the 2^n possible dichotomies of n points in R^d, the fraction of these that are achievable using a hyperplane (so-called linear dichotomies) is given by the function

$$f(n,d) = \begin{cases} 1 & n \leq d+1 \\ \dfrac{2}{2^n}\displaystyle\sum_{i=0}^{d}\binom{n-1}{i} & n > d+1 \end{cases} \tag{4.8}$$

At $n = 2(d+1)$ (this is sometimes referred to as the capacity of a d-dimensional hyperplane) $f(n,d) = \frac{1}{2}$. Thus, half the possible dichotomies are achievable with a linear discriminant function. The probability, as a function of $d+1$ and n, is shown in Figure 4.1.[2]

Remarks concerning Figure 4.1:

- Notice that as the ratio $n/(d+1) \to 0$, the probability of a linearly separable solution approaches 1. This makes sense intuitively, since the number of weights then far exceeds the number of patterns in H, and thus the degrees of freedom available exceed those required by the constraints in H.
- Similarly, as $n/(d+1)$ gets large, the probability of a linearly separable solution decreases, since we exceed the capacity of the hyperplanar implementation.
- A corollary to the preceding remark is that in cases where a linearly separable H is not given, we may increase d (and augment the input vectors as well) to achieve a linearly separable solution. This is shown in several of the examples in this chapter.

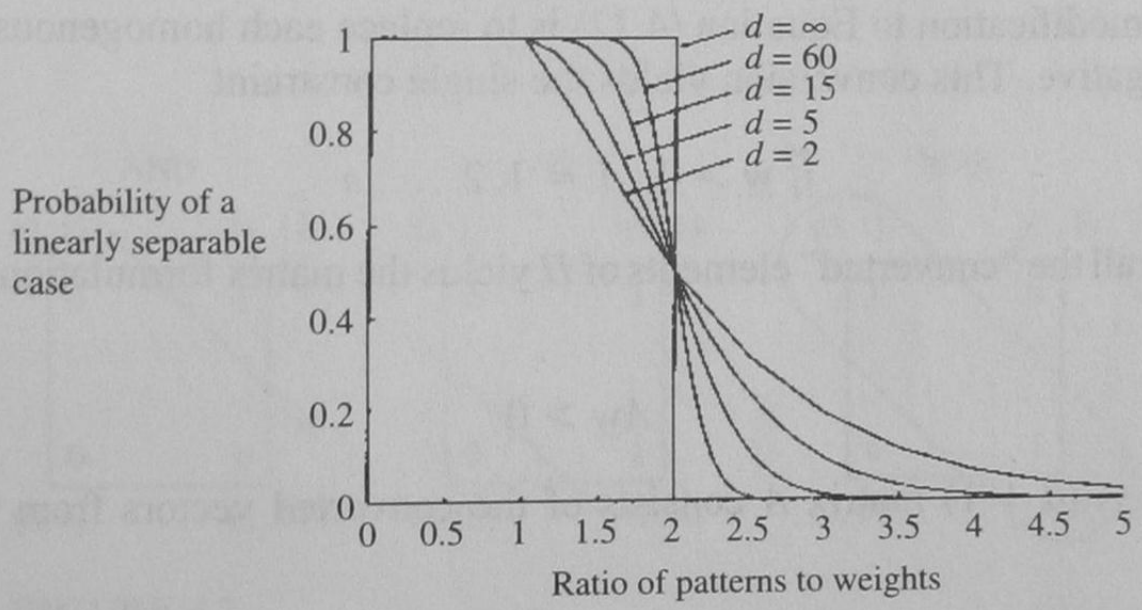


FIGURE 4.1
Probability of a linearly separable solution as a function of the ratio of input patterns to unit weights $(n/d+1)$, including bias.

[2]The reader may be confused by the use of $d+1$ as opposed to d. Here we allow the unit to have a bias input, or simply bias, represented by the $(d+1)$th weight.

4.2.2 Linear Programming and Linear Separability

Problem formulation

Assume that a $c = 2$ class training set $H = \{\underline{i}_i\}, i = 1, 2, \ldots, n$, may be partitioned into H_1 and H_2, where H_i consists only of samples labeled w_i. The goal is to determine a separating plane H_{12} determined by parameters $\underline{w}_{12}$ and w_o, where, for each $\underline{i}_i$ in H,

$$\underline{w}_{12}^T \underline{i}_i - w_o = \begin{cases} > 0 & \underline{i}_i \in H_1 \\ < 0 & \underline{i}_i \in H_2 \end{cases} \tag{4.9}$$

This plane is characterized by $d + 1$ parameters, namely, the d elements of $\underline{w}_{12}$ (the normal) and w_o, and is implementable using a single $d + 1$ input unit. Defining

$$\underline{w} = \begin{pmatrix} \underline{w}_{12} \\ w_o \end{pmatrix} \tag{4.10}$$

and converting each $\underline{i}_i$ in H to a $(d+1)\times 1$ vector by adding -1 as the $(d+1)$th element yields the standard homogenous coordinate representation (see, for example, [Sch89]):

$$\hat{\underline{i}}_i = \begin{pmatrix} \underline{i}_i \\ -1 \end{pmatrix} \tag{4.11}$$

Noting that $\underline{w}^T \hat{\underline{i}}_i^T$ is a scalar quantity, Equation (4.9) can be rewritten as

$$\hat{\underline{i}}_i^T \underline{w} = \begin{cases} > 0 & \hat{\underline{i}}_i \in H_1 \\ < 0 & \hat{\underline{i}}_i \in H_2 \end{cases} \tag{4.12}$$

A desirable modification to Equation (4.12) is to replace each homogenous vector $\hat{\underline{i}}_i$ in H_2 by its negative. This conversion yields the single constraint

$$\hat{\underline{i}}_i^T \underline{w} > 0 \quad i = 1, 2, \ldots, n \tag{4.13}$$

Considering all the "converted" elements of H yields the matrix formulation of Equation (4.13) as

$$A\underline{w} > \underline{0} \tag{4.14}$$

where the $n \times (d + 1)$ matrix A consists of the converted vectors from the training set as

$$A = \begin{pmatrix} \hat{\underline{i}}_1^T \\ \hat{\underline{i}}_2^T \\ \vdots \\ \hat{\underline{i}}_n^T \end{pmatrix} \tag{4.15}$$

and $\underline{0}$ is a vector of all zero elements.

Linear separability as a linear programming problem

Equation (4.14) shows the problem of seeking a hyperplanar decision boundary as a linear programming problem. It has been shown [Man65] that linear separability is equivalent to solving a linear programming problem. A necessary and sufficient condition for linear separability of the input sets H_1 and H_2 is

$$\psi(H_1, H_2) > 0 \tag{4.16}$$

where $\psi(H_1, H_2)$ is the solution of the linear programming problem

$$\psi(H_1, H_2) = \max_{\underline{w},\alpha,\beta} \left\{ (\alpha - \beta) \left| H_1 \begin{pmatrix} \underline{w} \\ \alpha \end{pmatrix} \geq 0, H_2 \begin{pmatrix} \underline{w} \\ \beta \end{pmatrix} \leq 0, -\underline{1} \leq \underline{w} \leq \underline{1} \right. \right\} \tag{4.17}$$

where $\underline{1}$ is a column vector of 1s. A necessary and sufficient condition for linear inseparability of the pattern sets H_1 and H_2 is that $\psi(H_1, H_2) = 0$. The proof follows based on the observation that $\underline{w} = 0, \alpha = 0, \beta = 0$ is a feasible solution and hence $\psi(H_1, H_2) \geq 0$. The solution vector $\hat{\underline{w}}$ is computed using

$$\hat{\underline{w}} = \begin{pmatrix} \underline{w} \\ \frac{1}{2}(\alpha - \beta) \end{pmatrix} \tag{4.18}$$

4.2.3 Examples

Logic functions and decision regions

Referring to Figure 4.2, we observe that some common $d = 2$ logic functions, i.e., OR and AND, are linearly separable and XOR is not.[3] Figure 4.3 shows how augmenting the input space with an additional dimension (input) converts the problem to a linearly separable formulation. An important note, however, is that the additional input must be computable from the other inputs.[4] One solution structure is shown in Figure 4.4.

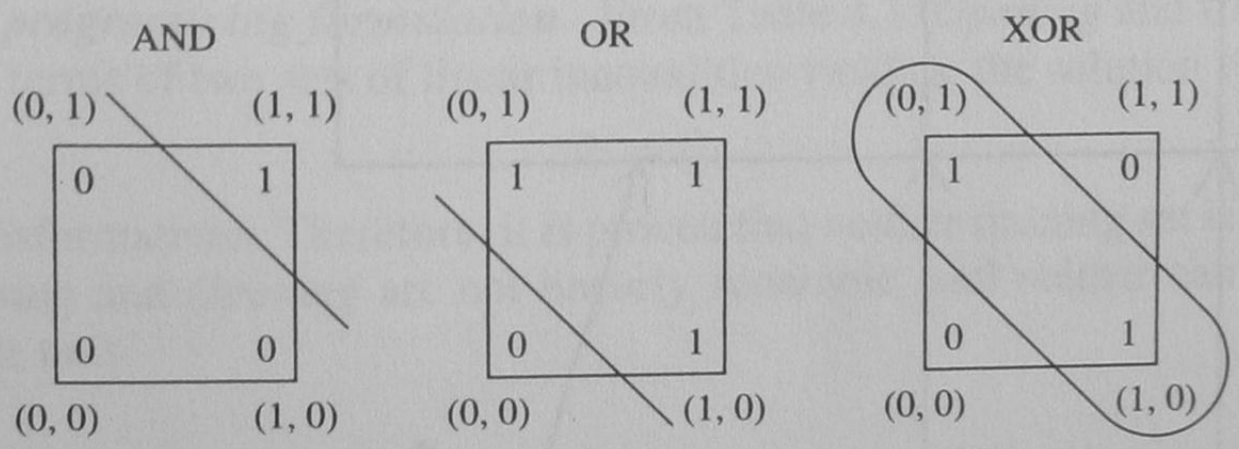


FIGURE 4.2
Simple $d = 2$ logic functions. Note that XOR is not linearly separable.

[3]This is a well-worn example of nonlinear separability.

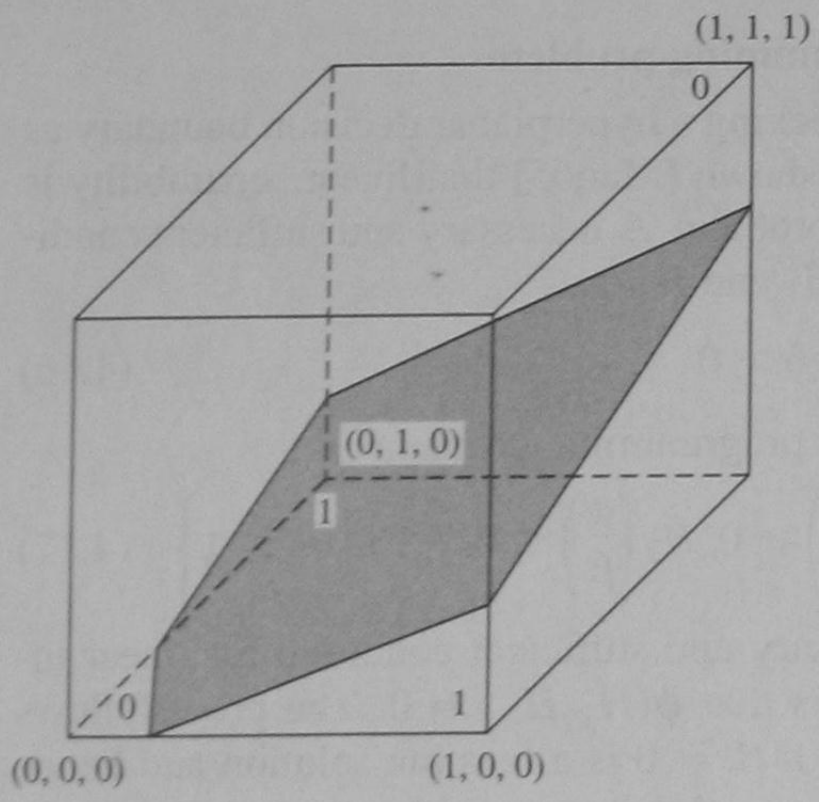


FIGURE 4.3
Solution to the $d = 2$ XOR problem by input augmentation and use of $d = 3$ space.

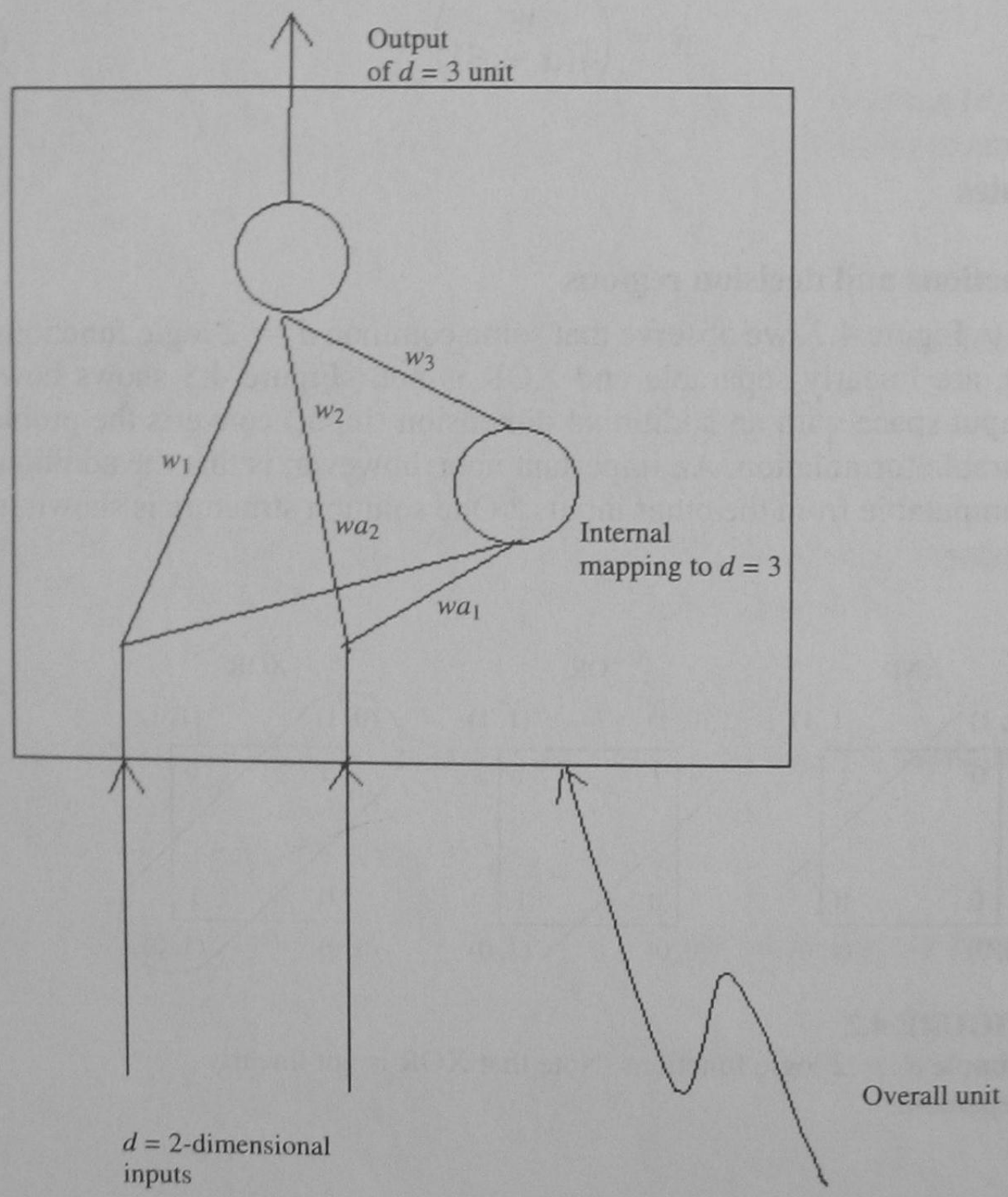
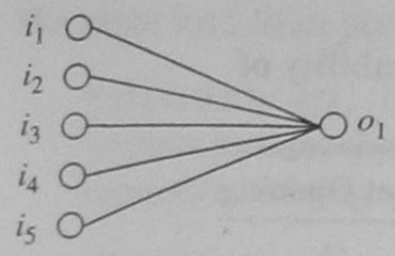


FIGURE 4.4
A structure for solution of the $d = 2$ XOR problem by mapping to $d = 3$ space.



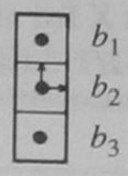


FIGURE 4.5
Minimal 5–1 network for testing linear separability of *Closing* and *Opening* operators (using a 3×1 structuring element, B).

Morphological signal processing

This example is taken from a broader study of ANN application to signal processing [HS94].

Opening and *Closing* are commonly encountered signal processing operators that are especially useful in 2-D (image) processing [Sch89]. A training set H must be designed that specifies the transformations *Opening* and *Closing*. This training set must be complete; i.e., it must include 2^n samples for n-dimensional input. Using a 3×1 structuring element, the single-unit configuration is shown in Figure 4.5. Here $n = 5$ results in a training set size of 32 samples.

Closing. For the given network, o is computed as

$$o = (i_1 \cup i_2 \cup i_3) \cap (i_2 \cup i_3 \cup i_4) \cap (i_3 \cup i_4 \cup i_5)$$

with operators defined as

$\cup$: logical OR
$\cap$: logical AND

This may be reduced to

$$o = i_3 \cup (i_2 \cap i_4) \cup (i_1 \cap i_4) \cup (i_2 \cap i_5) \tag{4.19}$$

Opening. The output o for the given network is computed as

$$o = (i_1 \cap i_2 \cap i_3) \cup (i_2 \cap i_3 \cap i_4) \cup (i_3 \cap i_4 \cap i_5) \tag{4.20}$$

The training sets X for *Closing* and *Opening* based on Equations (4.19) and (4.20) are illustrated in Table 4.1 for a $-1/1$ data representation.

Linear programming formulation. From Table 4.1, *Opening* and *Closing* can be specified in terms of two sets of linear inequalities yielding the solution vectors[5]

$$\underline{\hat{w}} = \underline{0} \tag{4.21}$$

for both transformations. Therefore, it is proven that neither training set is linearly separable. *Closing* and *Opening* are not linearly separable, and neither can be achieved with a single unit.

4.2.4 Alternative Constraint on a Nonlinearly Separable H

Derivation and proof

The constraint of Equation (4.14) yields another approach to determining nonlinearly separable training data. We show two interpretations that are modifications of [SRK94].

---

[5]It is left to the student to verify this.

TABLE 4.1
Minimal 1-D training sets X for testing the linear separability of *Closing* and *Opening*: A−1/1 data representation

Input					Target Closing	Target Opening
−1	−1	−1	−1	−1	−1	−1
−1	−1	−1	−1	1	−1	−1
−1	−1	−1	1	−1	−1	−1
−1	−1	−1	1	1	−1	−1
−1	−1	1	−1	−1	1	−1
−1	−1	1	−1	1	1	−1
−1	−1	1	1	−1	1	−1
−1	−1	1	1	1	1	1
−1	1	−1	−1	−1	−1	−1
−1	1	−1	−1	1	1	−1
−1	1	−1	1	−1	1	−1
−1	1	−1	1	1	1	−1
−1	1	1	−1	−1	1	−1
−1	1	1	−1	1	1	−1
−1	1	1	1	−1	1	1
−1	1	1	1	1	1	1
1	−1	−1	−1	−1	−1	−1
1	−1	−1	−1	1	−1	−1
1	−1	−1	1	−1	1	−1
1	−1	−1	1	1	1	−1
1	−1	1	−1	−1	1	−1
1	−1	1	−1	1	1	−1
1	−1	1	1	−1	1	−1
1	−1	1	1	1	1	1
1	1	−1	−1	−1	−1	−1
1	1	−1	−1	1	1	−1
1	1	−1	1	−1	1	−1
1	1	−1	1	1	1	−1
1	1	1	−1	−1	1	1
1	1	1	−1	1	1	1
1	1	1	1	−1	1	1
1	1	1	1	1	1	1

Recall the constraint of Section 4.2.2:

$$A\underline{w} > \underline{0} \tag{4.22}$$

where matrix A consists of the "preprocessed" inputs

$$A = \begin{pmatrix} \hat{\underline{i}}_1^T \\ \hat{\underline{i}}_2^T \\ \vdots \\ \hat{\underline{i}}_n^T \end{pmatrix} \tag{4.23}$$

We state and then prove and use the following:

THEOREM 4.1. If the nullspace (see Chapter 2) of A^T (in Equation 4.23) contains any vectors besides $\underline{0}$ whose elements are all the same sign, then A represents a nonlinearly separable training set.

An alternative and equivalent viewpoint, which follows from the definition of the nullspace of A^T, is that if there exists a set of coefficients, q_i, such that

$$\hat{\underline{i}}_1 q_1 + \hat{\underline{i}}_2 q_2 + \cdots + \hat{\underline{i}}_n q_n = \underline{0} \tag{4.24}$$

where $q_i \geq 0$ and at least one q_i is positive, then the set of training vectors is nonlinearly separable.

The proof is by contradiction. Equation (4.24) may be written as

$$\sum_{i=1}^{n} q_i \, \hat{\underline{i}}_i = \underline{0} \tag{4.25}$$

Similarly, the solution to Equation (4.13) requires

$$\hat{\underline{i}}_i^T \underline{w} > 0 \quad i = 1, 2, \ldots, n \tag{4.26}$$

Assume the existence of a linearly separable solution to Equation (4.26). Also, assume the existence of a set of q_i satisfying Equation (4.25). Using Equations (4.25) and (4.26), we get

$$\underline{w}^T \sum_{i=1}^{n} q_i \, \hat{\underline{i}}_i = \sum_{i=1}^{n} \underline{w}^T q_i \, \hat{\underline{i}}_i = \sum_{i=1}^{n} q_i \, \underline{w}^T \hat{\underline{i}}_i = 0 \tag{4.27}$$

which is clearly a contradiction to the premise in Equation (4.26).

Numerical examples

XOR. Here we show a few examples of the utility of the preceding derivation. Refer to the $d = 2$ XOR formulation in Table 4.2. Here ϵ is some positive quantity[6] used to enforce the right-hand side of Equation (4.26). The vectors have been preprocessed using Equation (4.11).

Looking at the four rows of Table 4.2 (i.e., the $\underline{i}_i^T$), note that the sum of these vectors is $\underline{0}$. Alternatively, there exists a set of coefficients $q_i = 1$, $i = 1, \ldots, 4$, that satisfies

TABLE 4.2
$d = 2$ XOR training set for linear separability analysis

Initial training set				Converted training set			
Input (i_1 i_2 i_{bias})			Output (o_1)	Input (i_1 i_2 i_{bias})			Output (o_1)
0	0	−1	<0	0	0	1	ϵ
0	1	−1	>0	0	1	−1	ϵ
1	0	−1	>0	1	0	−1	ϵ
1	1	−1	<0	−1	−1	1	ϵ

[6] We could simply choose $\epsilon = 1$ for simplicity, since the length of $\underline{w}$ is not constrained.

Equation (4.24). Thus, this set is nonlinearly separable. The following MATLAB results confirm this by showing the nullspace of A.

```
A =

     0     0     1
     0     1    -1
     1     0    -1
    -1    -1     1


> N=null(A')

N =

   -0.5000
   -0.5000
   -0.5000
   -0.5000
```

There are no sign changes; thus H is not linearly separable. Since the nullspace of A^T is the span of this vector, multiplying by -2 shows that the nullspace includes

$$\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

This set of coefficients, i.e., $q_i = 1 \; \forall i$, satisfies Equation (4.24).

OR. In a similar fashion, we show H for the OR function (known to be linearly separable from the simple example of Section 4.2.3) in Table 4.3. Notice that there is no *positive* set of coefficients that satisfies Equation (4.24); the following MATLAB results confirm this.

```
A =

     0     0     1
     0     1    -1
     1     0    -1
     1     1    -1

> N=null(A')

N =
```

TABLE 4.3
$d = 2$ OR training set for linear separability analysis

Initial training set				Converted training set			
Input (i_1 i_2 i_{bias})			Output (o_1)	Input (i_1 i_2 i_{bias})			Output (o_1)
0	0	−1	<0	0	0	1	ϵ
0	1	−1	>0	0	1	−1	ϵ
1	0	−1	>0	1	0	−1	ϵ
1	1	−1	>0	1	1	−1	ϵ

Linearly and nonlinearly separable subsets of H

A useful corollary to Theorem 1 allows us to see substructures within H. Specifically, we can identify vectors in H that may be responsible for the nonlinearly separable behavior of the training set.

COROLLARY 1. A vector in H can cause H to be nonlinearly separable if it has a corresponding positive q_i in Equation (4.24).

Referring to the previous XOR example, we note that all vectors may participate in the nonseparable formulation.

4.3 TECHNIQUES TO DIRECTLY OBTAIN LINEAR UNIT PARAMETERS

4.3.1 Batch (Pseudoinverse) Solution to Single-Unit Design

Equation (4.14) from Section 4.2.4 is a set of *n linear inequalities*. Many solution procedures exist, including linear programming. A solution is developed that is based on converting Equation (4.14) into a linear constraint by defining a vector of user-chosen "offsets," $\underline{b}$, as

$$\underline{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \quad b_i > 0 \tag{4.28}$$

Thus, Equation (4.14) becomes

$$A\underline{w} = \underline{b} \tag{4.29}$$

and a solution for the parameters of the separating plane, $\underline{w}$, is obtained by forming the pseudoinverse of A:

$$\underline{\hat{w}} = A^{\dagger}\underline{b} \tag{4.30}$$

Another approach for solving the system of linear inequalities given by Equation (4.12), or equivalently, (4.13), is to view these equations as n constraints in $(d + 1)$-dimensional space. Each equation of the form of (4.13) together with a user-chosen

offset or "margin" may be written as

$$\hat{\underline{i}}_i^T \underline{w} - b_i > 0 \quad i = 1, 2, \ldots, n \tag{4.31}$$

From Chapter 2, each of the n linear inequality constraints in Equation (4.31) may be visualized *by viewing* $\hat{\underline{i}}_i^T$ *as the normal vector to a* $(d+1)$*-dimensional hyperplane that partitions* R^{d+1}. A requirement for a solution is that the solution vector, $\underline{w}$, must lie in the positive half, R_p, of R^{d+1}, at a distance of $|b|/\|\hat{\underline{i}}_i^T\|$ from the boundary. Moreover, the intersection of the n half spaces of R^{d+1} defined by Equation (4.31) is the overall *solution region* for $\underline{w}$. In problems that are not linearly separable, this region does not exist. Conversely, in linearly separable solutions with nonunique separating planes, this region contains an infinite number of solution points. In addition, by setting the margins $b_i = 0, i = 1, 2, \ldots, n$, we find the largest solution region by solving

$$\hat{\underline{i}}_i^T \underline{w} > 0 \quad i = 1, 2, \ldots, n \tag{4.32}$$

4.3.2 Iterative (Gradient Descent) Solution Procedures

Using a gradient approach (Chapter 2), an iterative procedure to determine $\underline{w}$ may be found. The form is

$$\underline{w}^{(n+1)} = \underline{w}^{(n)} - \alpha_n \left. \frac{\partial J(\underline{w})}{\partial w} \right|_{w = w^{(n)}} \cdot \tag{4.33}$$

where α_n controls the adjustment at each iteration. The iteration procedure requires a stopping criterion. Examples are

$$\|\underline{w}^{(n+1)} - \underline{w}^{(n)}\| < \epsilon \tag{4.34}$$

where ϵ is a user-chosen tolerance, or

$$n = n_{\max} \tag{4.35}$$

where $n_{\max}$ is the (predetermined) maximum number of iterations, or

$$J(\underline{w}^{(n)}) \leq J_T \tag{4.36}$$

where J_T is an error threshold and $J(\underline{w})$ is an overall measure of classification error.

Many forms for the error, $J(\underline{w})$, are possible. For example, a vector $\underline{w}$, where

$$\hat{\underline{i}}_i^T \underline{w} < 0 \tag{4.37}$$

misclassifies sample $\hat{\underline{i}}_i^T$. Therefore, one error measure, the *perceptron criterion function*, is

$$J_p(\underline{w}) = - \sum_{\hat{\underline{i}} \in X_{\text{ERR}}(\underline{w})} (\hat{\underline{i}}_i^T \underline{w}) \tag{4.38}$$

where $X_{\text{ERR}}(\underline{w})$ is the set of *samples misclassified by* $\underline{w}$. Note that this set will vary from iteration to iteration in the solution procedure. If $X_{\text{ERR}}(\underline{w}) = \varnothing$, then $J_p(\underline{w}) = 0$ and

the minimum of the error function is obtained. Since

$$\nabla_{\underline{w}} J_p(\underline{w}) = -\sum_{\hat{\underline{i}}_i \in X_{\mathrm{ERR}}(\underline{w})} \hat{\underline{i}}_i \tag{4.39}$$

the iterative procedure of Equation (4.33) becomes

$$\underline{w}^{(n+1)} = \underline{w}^{(n)} + \alpha_n \sum_{\hat{\underline{i}}_i \in X_{\mathrm{ERR}}(\underline{w}^{(n)})} \hat{\underline{i}}_i \tag{4.40}$$

Notice that when $X_{\mathrm{ERR}}(\underline{w}^{(n)}) = \varnothing$, the adjustments to $\underline{w}^{(n)}$ cease.

Equation (4.40) suggests that at each iteration the entire set of samples misclassified by $\underline{w}^{(n)}$ can be used to form the correction at the next iteration. This entails a consideration of the entire training set for each adjustment of $\underline{w}$ and represents *training by epoch*. Another alternative is to adjust $\underline{w}$ as soon as a single classification error is made. This represents *training by sample* and may be viewed as a "correct as soon as possible" strategy. It is often unclear whether training by epoch or training by sample is preferable, and this concern carries over into our training of certain similar neural network structures in a later chapter. In the case of training by sample, Equation (4.40) becomes

$$\underline{w}^{(n+1)} = \underline{w}^{(n)} + \alpha_n \hat{\underline{i}}_i \tag{4.41}$$

where $\hat{\underline{x}}_i$ is the first sample misclassified by $\underline{w}^{(n)}$.

4.4 PERCEPTRONS AND ADALINE/MADALINE UNITS AND NETWORKS

4.4.1 Perceptron/Adaline Overview

The α-perceptron consists of two layers. The first is a remapping stage with fixed weights and sparse interconnections to the input, and the second is a regular feedforward network layer with adaptable weights and hardlimiter activation function. Only the second layer is generally referred to as a *perceptron*. The unit in the second layer is called a *linear threshold unit—linear* because of the computation of the activation value (inner product) and *threshold* to relate to the type of activation function (hardlimiter). Training of a perceptron is possible with the *perceptron learning rule*. Associated with the rule is the *perceptron convergence theorem* [MP69], [DH73], [KS91], which is of particular importance here since it proves convergence in the case that a solution exists.

> **THEOREM 4.2. PERCEPTRON CONVERGENCE THEOREM.** If there exists a set of connection weights $\hat{\underline{w}}^*$ that is able to perform the transformation T, the perceptron learning rule will converge to some solution $\hat{\underline{w}}$ (which may not be the same as $\hat{\underline{w}}^*$) in a finite number of steps for any initial choice of weights.

According to the theorem, training with the perceptron learning rule leads to a zero-error solution if the training set is linearly separable. An upper bound on the number of iterations necessary to achieve this solution can be computed. Unfortunately, this bound is expressed in terms of the unknown solution vector $\hat{\underline{w}}^*$.

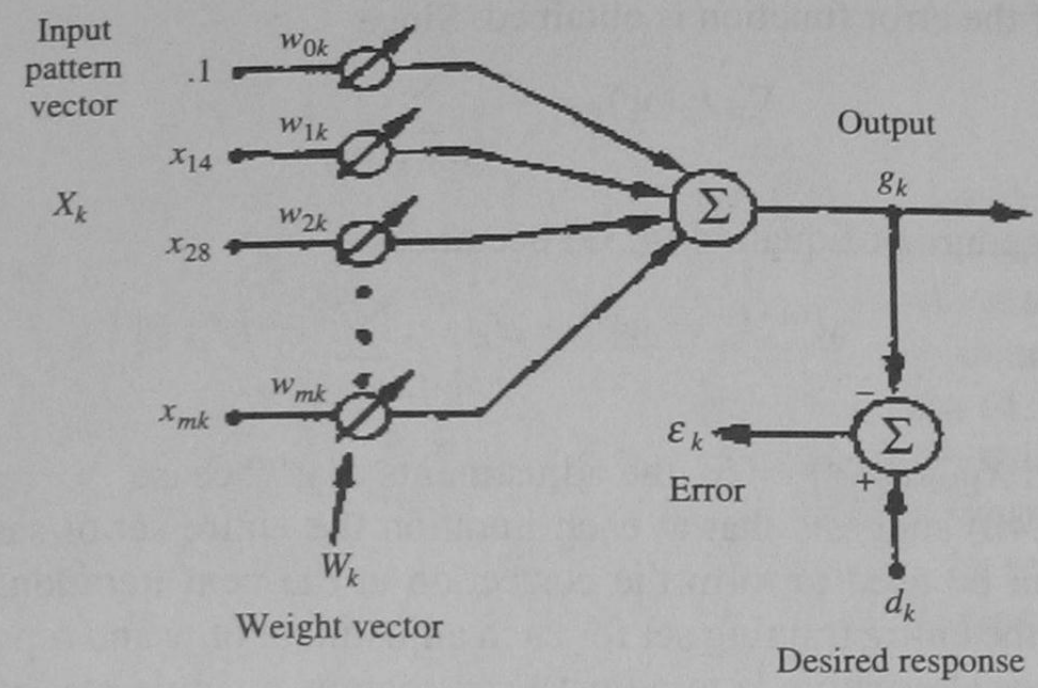


FIGURE 4.6
Adaptive linear combiner structure; note the use of linear error [WL90].

As shown in Figure 4.6,[7] the basis for the Adaline element is a single unit whose net activation is computed using a WLIC approach, i.e.,

$$net_i = \sum_j w_{ij} i_j = \underline{w}^T \underline{i} \tag{4.42}$$

The unit output is computed by using a hardlimiter, threshold-type nonlinearity, namely, the signum function: for unit i with output o_i,

$$o_i = \begin{cases} +1 & net_i \geq 0 \\ -1 & net_i < 0 \end{cases} \tag{4.43}$$

It is fundamental to note that the unit has a *binary output*; however, the formation of net_i (as well as weight adjustments in the training algorithm) is based on the linear portion of the unit, i.e., the mapping obtained prior to application of the nonlinear activation function. Figure 4.7 shows a more complete picture of the overall Adaline structure, and Figure 4.8 shows a simple two-input ($d = 2$) case with a unit bias.

Figure 4.9 shows the implementation of the AND, OR, and MAJ functions using a single Adaline unit.

4.4.2 α-LMS Algorithms

Here we briefly summarize the family of least mean squared (LMS) algorithms, with emphasis on simplicity and their relationship to gradient descent.

The LMS algorithm is based on *error correction* and *training by sample*. We first consider the following objective:

---

[7]Observe that, in figures excerpted from [WL90], the inputs are designated using the vector $\underline{x}$ rather than $\underline{i}$, as in our notation.

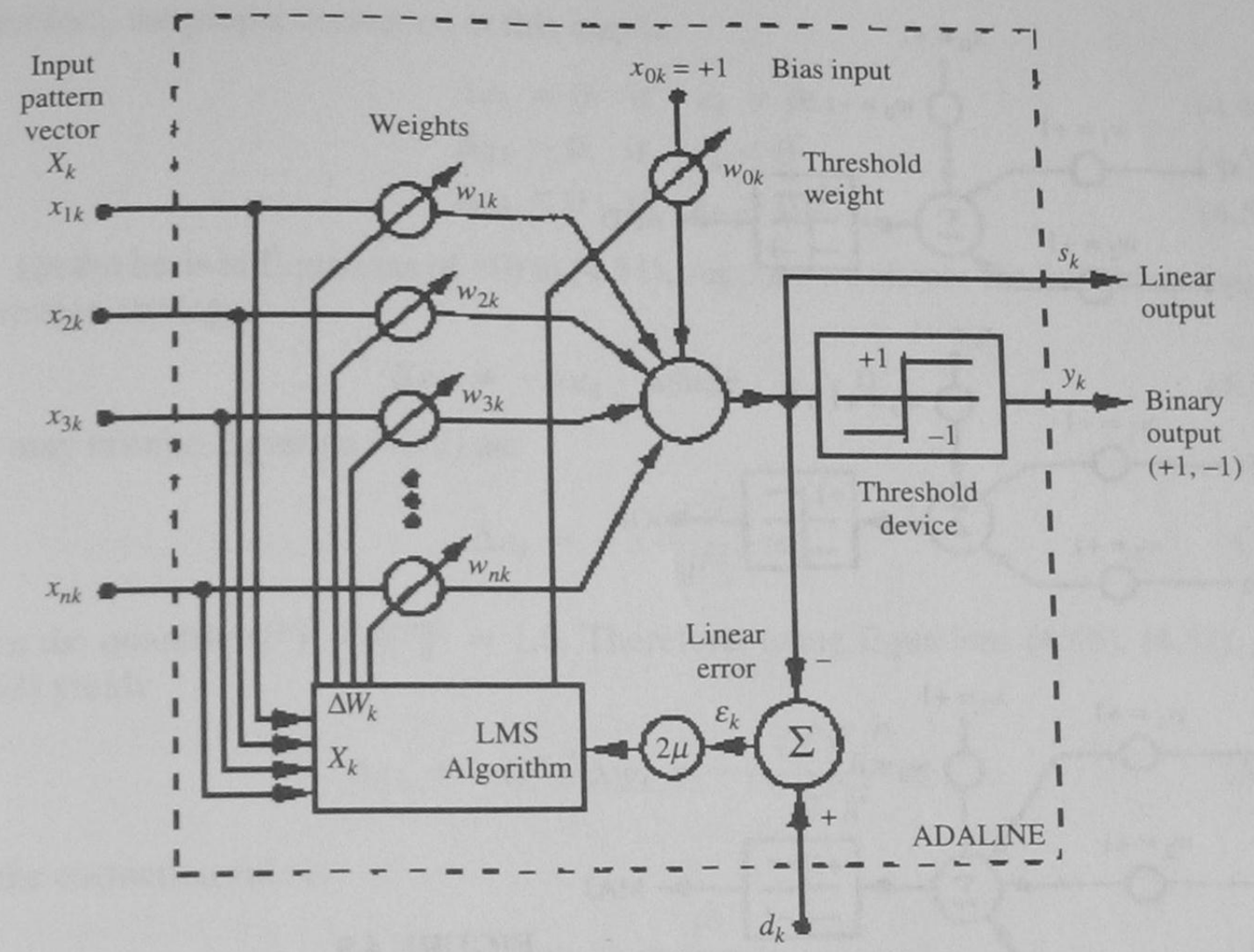


FIGURE 4.7
Adaline element, showing nonlinearity and training algorithm; note that the correction is based on the unit net activation ("Linear Output") and not the squashed ("Binary") output [WL90].

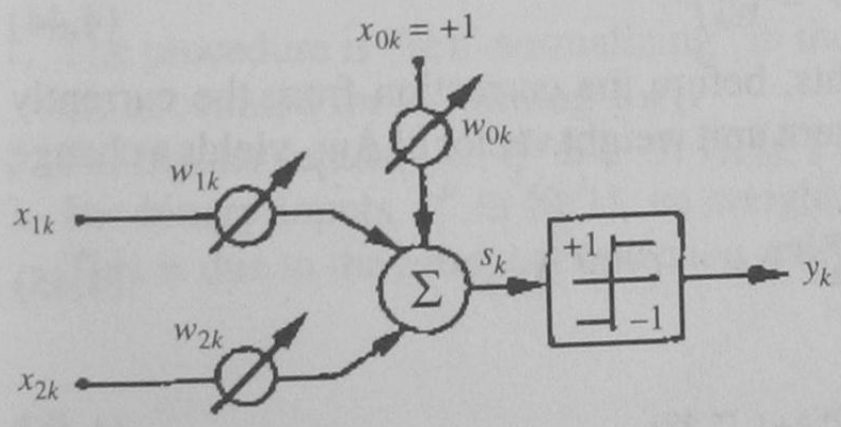


FIGURE 4.8
Two-input Adaline example [WL90].

Minimum disturbance principle: The system should adapt to reduce the output error for the *current training pattern*, with *minimal disturbance* to responses already learned.

The significance of this principle is that the unit weight correction that is ideal for the current pattern may be counterproductive in terms of other training set patterns that were previously used (or those yet to be processed). In other words, using training by sample, it is possible to "undo" some of the previous successful unit training with corrections made on subsequent samples.

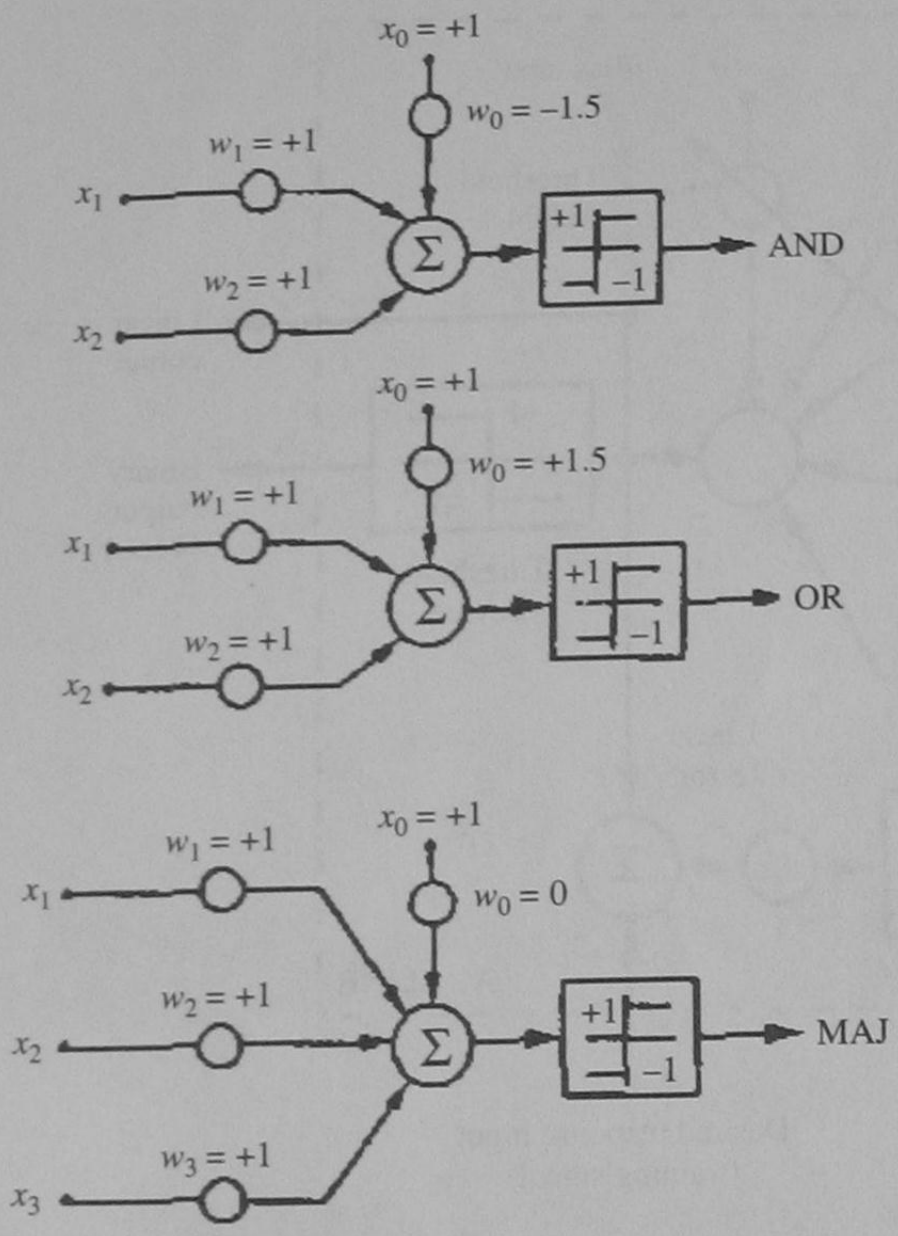


FIGURE 4.9
Adaline Implementation of AND, OR, MAJ Functions [WL90].

Derivation

Assume a training set of the form $H = \{\underline{i}^p, t^p\}$ is given, where t^p is the desired response, or "target," corresponding to input $\underline{i}^p$. Define the linear error at time[8] k to be

$$e_k \triangleq d^p - \underline{w}_k^T \underline{i}^p \tag{4.44}$$

where $\underline{w}_k$ is the current set of unit weights, before the correction from the currently presented pattern pair. A change in the current unit weight vector of $\Delta\underline{w}_k$ yields a change in the error

$$\Delta e = \Delta(d^p - \underline{w}_k^T \underline{i}^p) = -(\underline{i}^p)^T \Delta\underline{w}_k \tag{4.45}$$

If $\Delta\underline{w}_k$ is used as the weight change,

$$\Delta\underline{w}_k = \underline{w}_{k+1} - \underline{w}_k \tag{4.46}$$

or

$$\underline{w}_{k+1} = \underline{w}_k + \Delta\underline{w}_k \tag{4.47}$$

Since $\underline{i}^p$ and d^p are fixed, observe

$$e_k = \begin{cases} >0 & d^p > \underline{w}_k^T \underline{i}^p \\ 0 & d^p = \underline{w}_k^T \underline{i}^p \\ <0 & d^p < \underline{w}_k^T \underline{i}^p \end{cases} \tag{4.48}$$

[8]Or iteration.

Therefore, the proper correction at this step is

$$\Delta e_k = 0 \quad \text{if} \quad e_k = 0 \tag{4.49}$$

$$\Delta e_k > 0 \quad \text{if} \quad e_k < 0 \tag{4.50}$$

$$\Delta e_k < 0 \quad \text{if} \quad e_k > 0 \tag{4.51}$$

On the basis of Equations (4.49) to (4.51), suppose we choose the following weight correction strategy:

$$\Delta e_k = -\alpha e_k \quad \text{where} \quad \alpha > 0 \tag{4.52}$$

We may rewrite Equation (4.52) as:

$$\Delta e_k = -\alpha \frac{(\underline{i}^p)^T \underline{i}^p}{\|\underline{i}^p\|^2} e_k \tag{4.53}$$

since the quantity $(\underline{i}^p)^T \underline{i}^p / \|\underline{i}^p\|^2 = 1.0$. Therefore, using Equations (4.45), (4.52), and (4.53) yields

$$\Delta e_k = -(\underline{i}^p)^T \Delta \underline{w}_k = -\alpha \frac{(\underline{i}^P)^T \underline{i}^P}{\|\underline{i}^P\|^2} e_k \tag{4.54}$$

So the correction rule is

$$\Delta \underline{w}_k = \alpha \frac{\underline{i}^p}{\|\underline{i}^p\|^2} e_k \tag{4.55}$$

Assessment of the LMS procedure

Typically,

$$0.1 < \alpha < 1.0 \tag{4.56}$$

We observe several things regarding Equation (4.55):

1. The procedure is "self-normalizing" in the sense that the effect of variations in $\|\underline{i}^p\|$ are accounted for in forming $\Delta \underline{w}_k$.
2. For bipolar inputs, i.e., $i_i^p \in \{-1, 1\}$, $\|\underline{i}^p\|$ is constant over all inputs $\underline{i}$.
3. For binary inputs, $i_i^p \in \{0, 1\}$, no weights corresponding to $i_i^p = 0$ are changed.[9] This is due to the *product correction rule* nature of the algorithm and is not ideal.

4.5 MULTILAYER PERCEPTRONS (MLPs)

As we have seen, the WLIC-T unit characteristic, including the Adaline form, suffers from the inability to implement nonlinearly separable training set mappings. Numerous attempts have been made to overcome this limitation, some of which are described here. Of particular interest is the use of *layers of Adaline units*, often referred to as multilayer perceptrons, or MLPs. One of the biggest shortcomings of MLPs, however, is

[9] Another viewpoint is that "zeros don't count" in the correction procedure

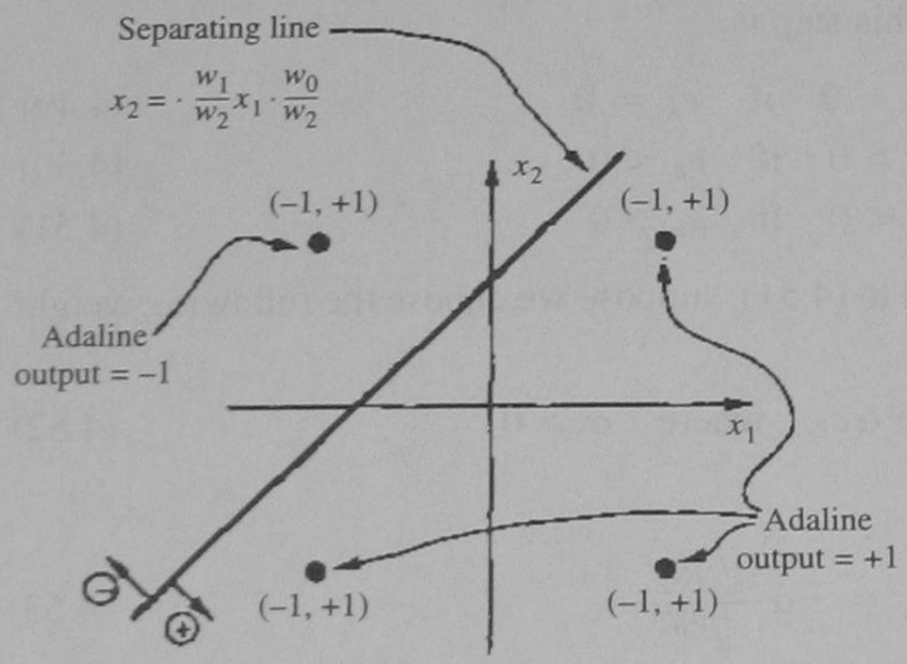


FIGURE 4.10
Adaline separating hyperplane (line) in input space [WL90].

the limited availability of suitable training algorithms. This shortcoming often reduces the applicability of the MLP to small, "hand-worked" solutions.

The WLIC structure of the Adaline unit allows for hyperplanar decision boundaries, as shown in Figure 4.10. The Adaline unit can provide only a single hyperplanar boundary in input space. Several modifications to allow application to nonlinearly separable problems are possible. One involves preprocessing inputs (and increasing the dimension of the input space) to achieve more complex boundaries. This is shown in Figure 4.11.

As shown in Figure 4.12, combinations of Adaline units yield the Madaline (*M*odified *Adaline*) structure, which may be used to form more complex decision regions (Figure 4.13). More generally, note that a two-layer Adaline/perceptron structure is capable of delineating *polyhedral regions in* R^d, and this, as we have shown by example, leads to more useful mappings in some applications.

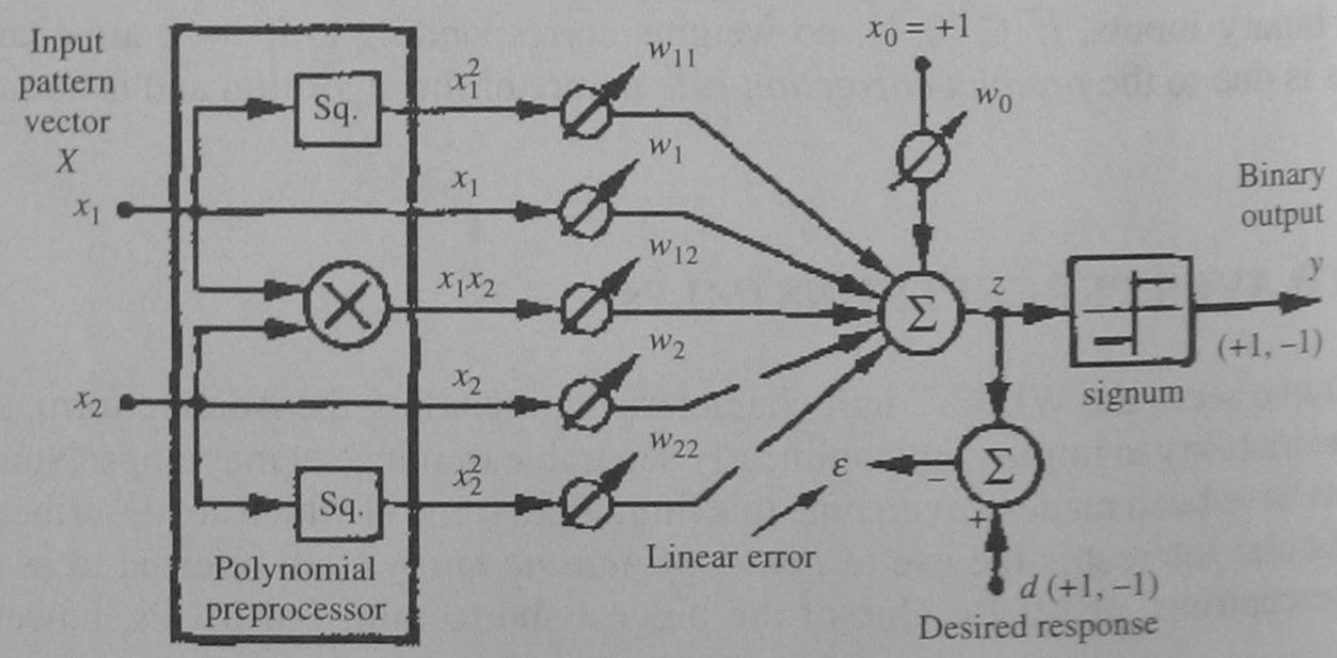


FIGURE 4.11
Using nonlinear mappings of inputs to achieve Adaline mappings [WL90].

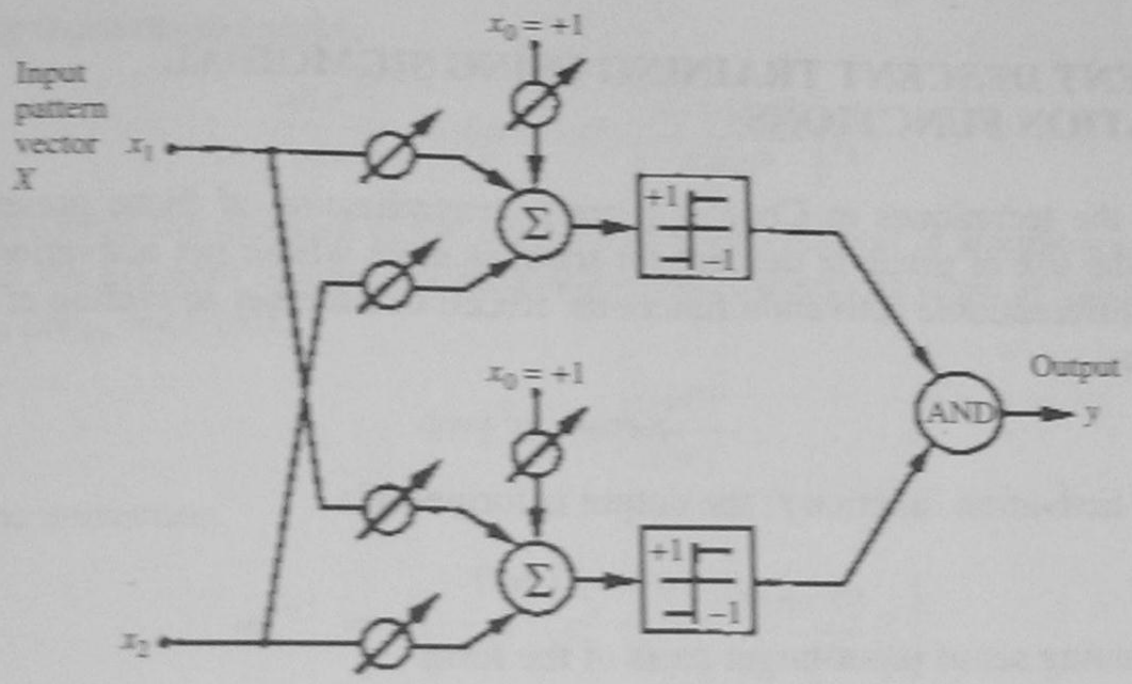


FIGURE 4.12
Modified Adaline (Madaline) [WL90].

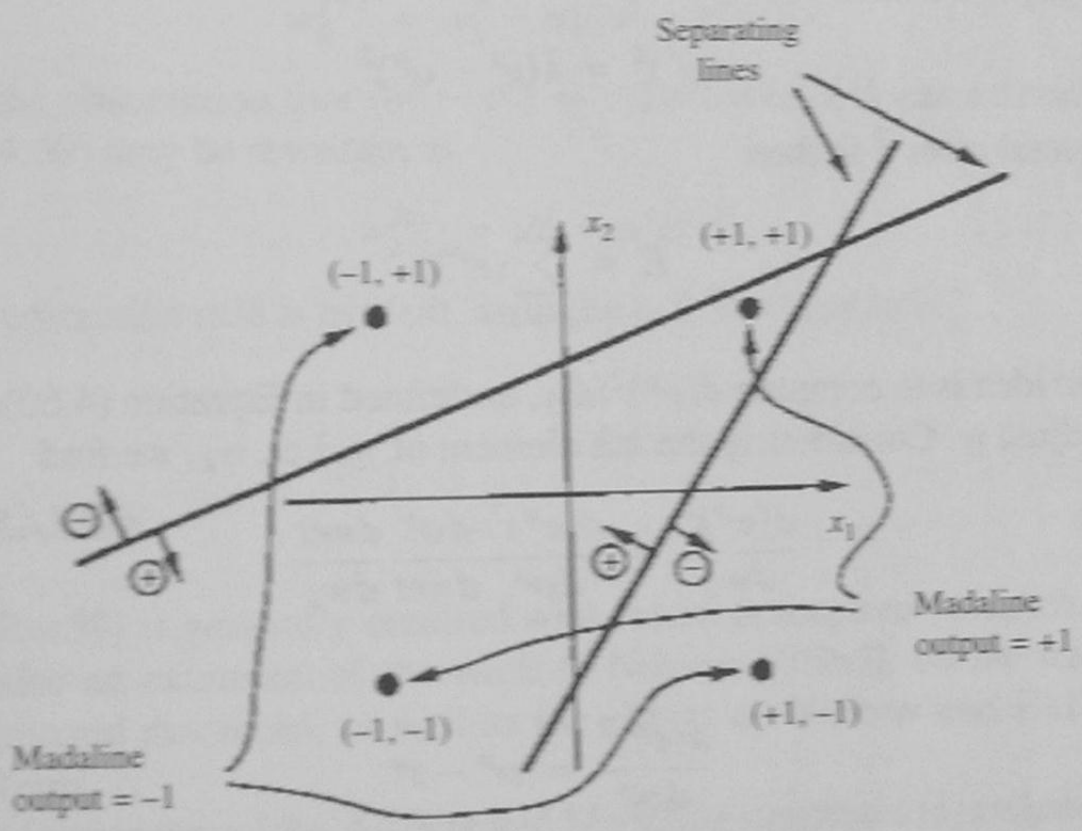


FIGURE 4.13
Sample Madaline decision boundary [WL90].

A *polyhedral subset* of R^d is defined to be the region defined by the intersection of some number of half spaces. Each of these half spaces may be delineated by a WLIC-T unit with appropriate parameters. In a two-layer configuration of WLIC-T units,[10] the first layer is capable of implementing q hyperplanes and thus dividing R^d into up to q half spaces. The second layer is used to map these half spaces into polyhedral regions or subsets by intersecting half spaces.

[10] Assume that it is composed of q units, each with d inputs and a bias.

4.6 GRADIENT DESCENT TRAINING USING SIGMOIDAL ACTIVATION FUNCTIONS

Although the techniques in Chapter 6 are a generalization of those presented here, we show the use of gradient descent for training units whose net activation is WLIC and with differentiable activation functions. Recall that the net activation of the unit is formed via

$$net = \underline{w}^T \underline{i} \tag{4.57}$$

and, given activation function f, the output is formed via

$$o = f(net) \tag{4.58}$$

Given a training set of input-target pairs of the form

$$H = \{(\underline{i}^1, t^1), (\underline{i}^2, t^2), \ldots, (\underline{i}^n, t^n)\} \tag{4.59}$$

and defining a single-pair error measure e^p based on $(t^p - o^p)$, where o^p is the output obtained from $\underline{i}^p$, we have[11]

$$(e^p)^2 = \tfrac{1}{2}(t^p - o^p)^2 \tag{4.60}$$

Over H, the total error[12] is then

$$E = \sum_{p=1}^{n} (e^p)^2 \tag{4.61}$$

The basic idea is to compute $d(e^p)^2/d\underline{w}$, as defined in Equation (4.60), and use this quantity to adjust $\underline{w}$. Considering the kth element of $\underline{w}$, i.e., w_k, we find

$$\frac{d(e^p)^2}{dw_k} = \frac{d(e^p)^2}{do^p} \frac{do^p}{dnet} \frac{dnet}{dw_k} \tag{4.62}$$

Therefore,

$$\frac{d(e^p)^2}{do^p} = o^p - t^p \tag{4.63}$$

$$\frac{do^p}{dnet^p} = \frac{d}{dnet^p} f(net^p) \tag{4.64}$$

where it is assumed that the quantity in Equation (4.64) exists. From Equation (4.57),

$$\frac{dnet^p}{dw_k} = i_k^p \tag{4.65}$$

---

[11] The reader should note that the factor of $\frac{1}{2}$ is for convenience in the derivation only. Minimizing any positive multiple of $(e^p)^2$ yields the same result.

[12] Or two times this quantity if the total sum of squares (TSS) error is desired.

Combining these results yields

$$\frac{d(e^p)^2}{dw_k} = (o^p - t^p)\left[\frac{df(net^p)}{dnet^p}\right] i_k^p \tag{4.66}$$

which therefore is the gradient of the pattern error with respect to weight w_k and forms the basis of the gradient descent training algorithm. Specifically, we assign the weight correction, Δw_k, such that

$$\Delta w_k = -\alpha \frac{d(e^p)^2}{dw_k} \tag{4.67}$$

yielding the correction

$$w_k^{j+1} = w_k^j - \alpha\left[(o^p - t^p)\frac{df(net^p)}{dnet^p} i_k^p\right] \tag{4.68}$$

Several points concerning Equation (4.68) are as follows:

- For a linear activation function ($o = net$), Equation (4.68) may be rewritten as

$$w_k^{j+1} = w_k^j - \alpha[(o^p - t^p) i_k^p] \tag{4.69}$$

- Based on the observation that $(o^p - t^p) = -2e^p$, with a linear activation function Equation (4.69) may be rewritten as

$$w_k^{j+1} = w_k^j + \alpha' e^p i_k^p \tag{4.70}$$

- A product correction rule is evident, as in the α-LMS approach.

4.7 BIBLIOGRAPHY

Rosenblatt [Ros59] is generally credited with initial perceptron research. The general structure is also an extension of the work of Nilsson [Nil65], on the transformations enabled by layered machines, as well as the efforts of Widrow and Hoff [WH60], in adaptive systems.

Up-to-date coverage of the Adaline and Madaline structures is presented in [WL90] and [WW88]. An extended analysis of linear separability and its relation to linear programming is found in [SRK94].

From a pattern recognition viewpoint, the computational advantages (in both implementation and training) and ease of visualization of linear classifiers account for their popularity. Seminal works include [Fis50], [HK65], and [FO70]. An extensive consideration of the development of linear discriminant functions for minimum error is found in [Fuk72]. A summary of linear classifiers in the pattern recognition context is found in [Sch92]. A good summary of the concept of linear separability as related to threshold logic and switching theory is in [GCO80].

An interesting extension of the hyperplanar-based classification strategy of the WLIC-T (Adaline) unit is the concept of units that implement *hypercone* decision regions. A training algorithm and sample results are provided in [Wan94].

In the next chapter we explore extensions of linear classifiers to layers of linear (and semilinear) classifiers that achieve complex decision boundaries, and we expand our capabilities to include several more complex types of neural networks.

REFERENCES

[AR88] J. A. Anderson and E. Rosenfeld, eds. *Neurocomputing: Foundation of Research.* MIT Press, Cambridge, MA, 1988.

[Cov65] T. M. Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers,* EC-14:326–334, June 1965.

[DH73] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, New York, 1973.

[Fis50] R. A. Fisher. The use of multiple measurements in taxonomic problems. In *Contributions to Mathematical Statistics*. John Wiley & Sons, New York, 1950. (A reprint of the original paper.)

[FO70] Fukunaga, K., and D. R. Olsen, "Piecewise Linear Descriminant Functions and Classification Errors for Multiclass Problems", *IEEE Transactions on Information Theory,* IT-16, pp. 99–100, 1970.

[Fuk72] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 1972.

[GCO80] R. M. Glorioso and F. C. Colon-Osorio. *Engineering Intelligent Systems: Concepts, Theory and Applications*. Digital Press, Maynard, MA, 1980.

[HK65] Y. C. Ho and R. L. Kayshap. An algorithm for linear inequalities and its application. *IEEE Transactions on Electronic Computers*, EC-14:683–688, October 1965.

[HS94] C. B. Herwig and R. J. Schalkoff. Morphological image processing using artificial neural networks. In C. T. Leondes, ed. *Control and Dynamic Systems*, vol. 67. Academic Press, New York, 1994.

[KS91] B. J. A Kröse and P. P. van der Smagt. *An Introduction to Neural Networks,* 4th ed. Technical report, University of Amsterdam, Faculty of Mathematics and Computer Science, September 1991.

[Man65] O. L. Mangasarian. Linear and nonlinear separation of patterns by linear programming. *Operations Research,* 13:444–452, 1965.

[MP69] M. L. Minsky and S. A. Papert. *Perceptrons: An Introduction to Computational Geometry.* MIT Press, Cambridge, MA, 1969.

[Nil65] N. Nilsson. *Learning Machines.* McGraw-Hill, New York, 1965.

[Ros59] F. Rosenblatt. *Principles of Neurodynamics.* Spartan Books, New York, 1959.

[Sch89] R. J. Schalkoff. *Digital Image Processing and Computer Vision*. John Wiley & Sons, New York, 1989.

[Sch92] R. J. Schalkoff. *Pattern Recognition: Statistical, Structural and Neural Approaches.* John Wiley & Sons, New York, 1992.

[SRK94] K. Y. Siu, V. Roychowdhury, and T. Kailath. *Discrete Neural Computation: A Theoretical Foundation*. Prentice Hall, Englewood Cliffs, NJ, 1994.

[Wan94] S. J. Wan. Cone algorithm: An extension of the perceptron algorithm. *IEEE Transactions on Systems, Man, and Cybernetics,* 24(10):1571–1576, October 1994.

[WH60] B. Widrow and M. E. Hoff. Adaptive switching circuits. In *1960 IRE WESCON Convection Record, Part 4*, August 1960, pp. 96–104, (Reprinted in [AR88]).

[WL90] B. Widrow and M. A. Lehr. 30 Years of adaptive neural networks: Perceptron, Madaline and backpropagation. *Proceedings of the IEEE*, 78(9):1415–1442, September 1990.

[WW88] B. Widrow and R. G. Winter. Neural nets for adaptive filtering and adaptive pattern recognition. *IEEE Computer*, 21:25–39, March 1988.

PROBLEMS

4.1. Compare the LMS algorithm with that obtained using gradient descent. Be specific; emphasize similarities and differences.

4.2. We know from Chapter 2 that a weighted-error pseudoinverse formulation is possible. We also know that nonequal presentation of pattern pairs in an iterative solution is possible. Both are useful when certain input-output mappings are more significant than others. Compare, as quantitatively as possible, the two approaches.

4.3. Suppose we try to use the pseudoinverse formulation

$$A^{\dagger} = (A^T A)^{-1} A^T \tag{4.71}$$

with a *single* multiple-input, one-output unit and a *single training pattern*. Show where this approach fails.

4.4. The desired characteristics for a D/A converter are shown in Table P4.4. For parts (*a*) and (*b*), *the unit is to be trained without a bias input* and with a linear activation function ($o = net$).

TABLE P4.4
Desired D/A converter characteristics

Inputs		Output
0	0	1/8
0	1	3/8
1	0	5/8
1	1	7/8

(*a*) (i) Formulate the solution and develop weights using the LMS algorithm.
(ii) Repeat (i) using the pseudoinverse formulation.
(*b*) Repeat part (*a*) but with the additional constraint that the last input-output training pair [$(1, 1) \rightarrow 7/8$] is much more critical and therefore deserves special emphasis.
(*c*) Repeat part (*a*) but allow a bias input.

4.5. (*a*) Design an inverter (in 0–1 logic, i.e., with outputs of 0 or 1) using a single unit with a sigmoidal net_i–o_i squashing function. Is a solution possible without a bias input?
(*b*) Design an inverter using a cascaded pair of single units with sigmoidal net_i–o_i squashing functions (our first two-layer net). Comment on the role of bias inputs in this case.

4.6. Consider a four-input, one-output *parity detector*. The output is 1 if the number of inputs is even; otherwise it is 0. Is this problem linearly separable? Justify your answer.

4.7. For the training set shown in the morphological signal processing example in Section 4.2.3, find several nonlinearly separable subsets.

4.8. This problem is somewhat whimsical, yet it illustrates a significant point. Use the OR training data of Table 4.3 in Section 4.2.4, together with the gradient descent–based training algorithm of Section 4.3.2 [Equation (4.33)], *but with a negative value of* α. What we are attempting, therefore, is *gradient ascent*. Intuition suggests that to maximize the error

measure, the algorithm of Equation (4.33) would try to put all the samples *on the wrong side of the hyperplane*. Does this happen?

4.9. Cite and discuss the relative advantages and disadvantages of training by sample versus training by epoch.

4.10. Show how the solution of Equation (4.41), which is equivalent to adding a scalar multiple of a misclassified vector to $\underline{w}^{(n)}$, leads to a solution vector and discuss why this makes intuitive sense.

4.11. Assume a problem of the form of Equation (4.14) is linearly separable.
(*a*) Show that the choice of the margin vector $\underline{b} = \underline{0}$ may lead to a "marginal" solution, in the sense that the resulting hyperplane may be very close to several samples and therefore ill-posed to handle errors or vectors that are perturbed versions of those in H.
(*b*) Show that an injudicious choice of $\underline{b}$ may lead to a formulation with no solution.

4.12. The network shown in Figure P4.12 uses sigmoidal units with $\lambda = 1$. For $o_1 = 0.28$ and $o_2 = 0.73$, find the input vector, $\underline{i}$, that has been applied to the network.

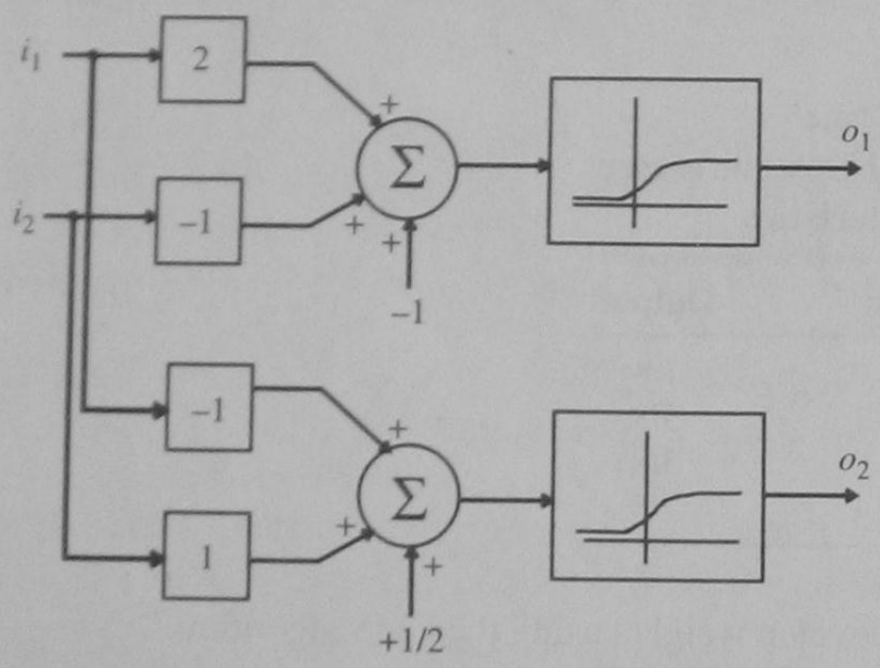


FIGURE P4.12
Finding ANN inputs (inversion example).

4.13. For the feedforward net shown in Figure P4.13 find the region(s) of i_1–i_2 space for which $o = 1$. Draw your answer.

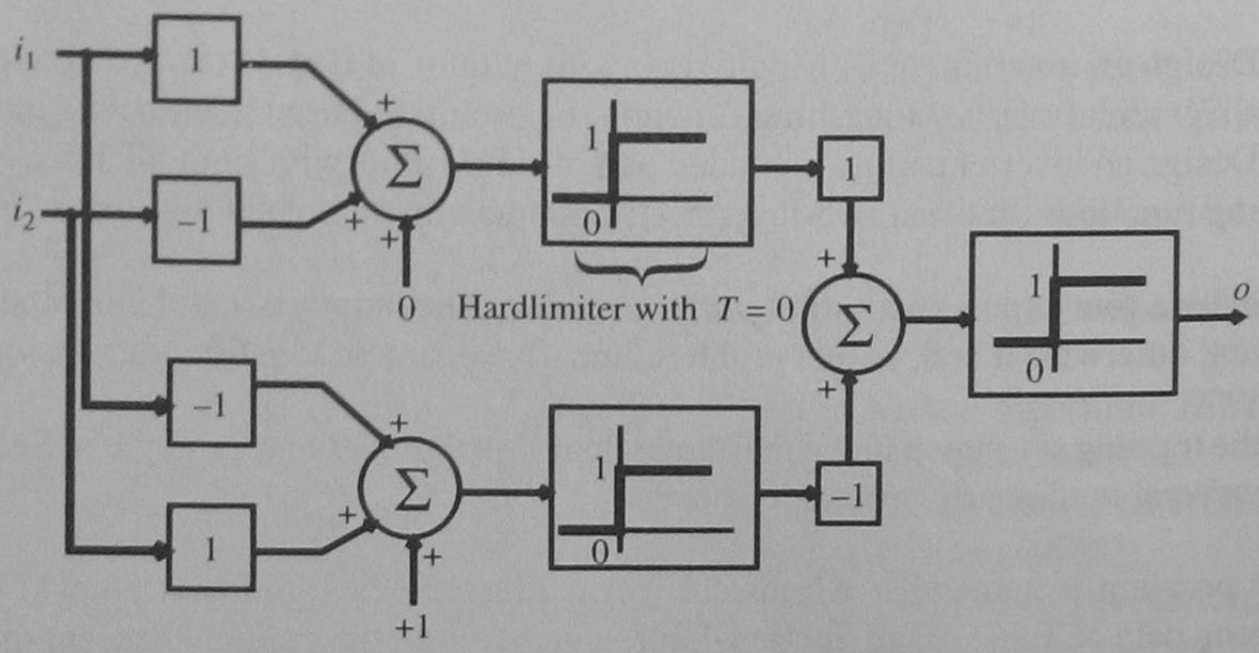


FIGURE P4.13
Three-unit ANN mapping network (classifier).

Problems 4.14, 4.15, and 4.16 illustrate some aspects of gradient descent procedures. Readers may wish to review Chapter 2 prior to attempting these problems.

4.14. In this chapter and Chapter 2, we considered the training of a single linear unit using gradient descent. The purpose of this problem is to verify and then extend these results. The problem is formulated as $P\underline{w} = \underline{t}$, with

$$P = \begin{pmatrix} 1 & -1 \\ 2 & 1 \end{pmatrix} \quad \underline{t} = \begin{pmatrix} -1 \\ 4 \end{pmatrix} \tag{4.72}$$

Using the *total mapping error over the training set* ($E = \underline{e}^T\underline{e} = e_1^2 + e_2^2 + \cdots + e_n^2$), or what is referred to as *epoch-based training*, the gradient descent solution is to form

$$\underline{w}^{n+1} = \underline{w}^n - \rho \frac{dE}{d\underline{w}}\bigg|_{\underline{w}=\underline{w}^n} \tag{4.73}$$

For the data given, implement the gradient descent solution, pick some values of $= \underline{w}(0)$ and ρ, and show the solution.

4.15. Extend Problem 4.14 by adding another element to H; i.e., make $n = 3$ with the resulting formulation

$$P = \begin{pmatrix} 1 & -1 \\ 2 & 1 \\ -1 & 3 \end{pmatrix} \qquad \underline{y} = \begin{pmatrix} -1 \\ 4 \\ 8 \end{pmatrix} \tag{4.74}$$

Plot the error surface as a function of w_1 and w_2. Repeat Problem 4.14 for this case and assess your results.

4.16. Instead of using E (or $dE/d\underline{w}$) at each iterative step (as in Problems 4.14 and 4.15), consider the implementation of *pattern-based training* where the error for each element of the training set is used individually; i.e., $E^p = e_p^2$ replaces E in Equation (4.73) and is used to correct $\underline{w}$ at each iteration. Revise the formulation in Problems 4.14 and 4.15 to incorporate this, and show sample results.

4.17. For the nonlinearly separable XOR shown (in R^2) in Table P4.17:

(*a*) Increase the dimension of the input space to achieve a linearly separable mapping in R^3.

(*b*) For your answer to part (*a*):

(i) Determine the parameters of the separating hyperplane.

(ii) Show how the additional input(s) is (are) formed from i_1 and i_2.

TABLE P4.17
XOR problem

i_2	i_1	$i_1 \oplus i_2$
0	0	0
0	1	1
1	1	0
1	0	1

4.18. Consider the formulation of a digital-to-analog (D/A) converter unit mapping shown in Table P4.18 For this mapping, use a single linear unit, with and without bias, and the

pseudoinverse training formulation to determine unit weights. Discuss the resulting design.

TABLE P4.18
Revised D/A design parameters

i_2	i_1	Output
0	0	0
0	1	0.75
1	0	0.50
1	1	0.25

4.19. (Generalization) Recall the desired characteristics for a D/A converter, shown in Table P4.4. The single-unit form of the sigmoid-based D/A converter to be designed is shown in Figure P4.19. *In each case, the unit is to be trained with and without a bias input.*

(*a*) Formulate the solution and develop weights using a sigmoidal activation function but with the restriction that one pattern (of the four possible) is left out of the training set. This yields four subproblems.

(*b*) Evaluate your network response (i.e., generalization) to the training pair that was omitted in each case.

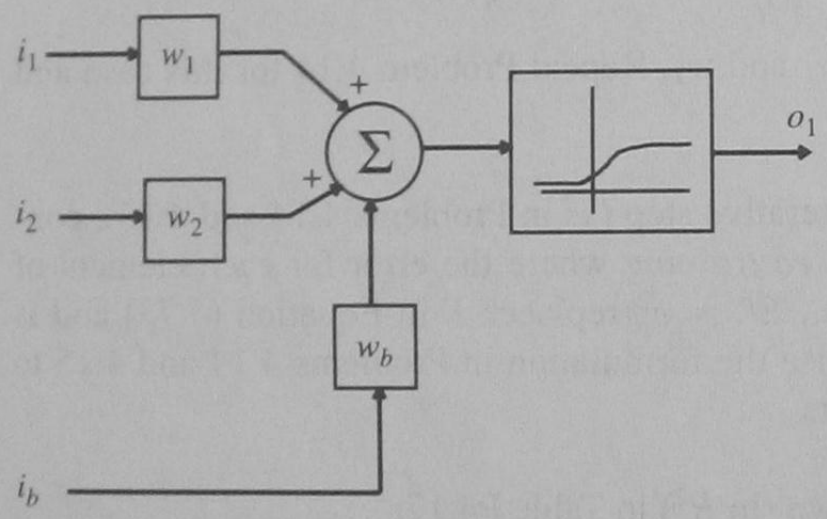


FIGURE P4.19
Single-unit sigmoid-based D/A converter structure (shown with bias input).

4.20. Verify the following statement: The weight correction strategy of Equation (4.55) causes the error at each iteration to be reduced by a factor of α.

4.21. Comment on the following statements:

1. The perceptron implementations of AND and OR logic functions differ only by the value of unit bias.
2. The perceptron implementations of AND and OR logic functions could be achieved without bias by changing the activation function threshold.

4.22. Repeat the analysis of Section 4.2.4 using the logical OR function.

4.23. Repeat the analysis of Section 4.2.4 using the training sets for *Opening* and *Closing* shown in Section 4.2.3, Table 4.1. Can you use Corollary 1 to identify elements in each case that contribute to the nonlinearly separable nature of H?

4.24. For the crosshatched decision region shown in Figure P4.24, design a two-layer configuration that uses WLIC-T units and implements the mapping shown.

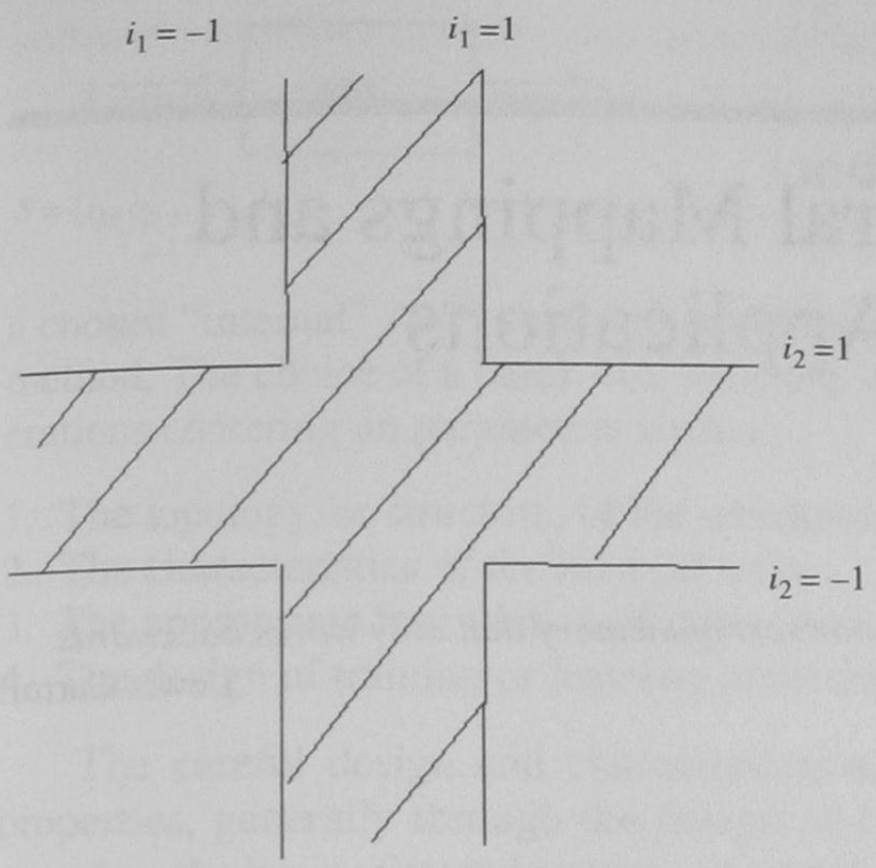


FIGURE P4.24
Regions to be mapped by MLP.

4.25. Each of the ANN designs in Figure P4.25 uses the McCulloch-Pitts neuron model. Assume that $\sum i_i \geq T$ for $o_i = 1$. Find the logic functions that are implemented by each design.

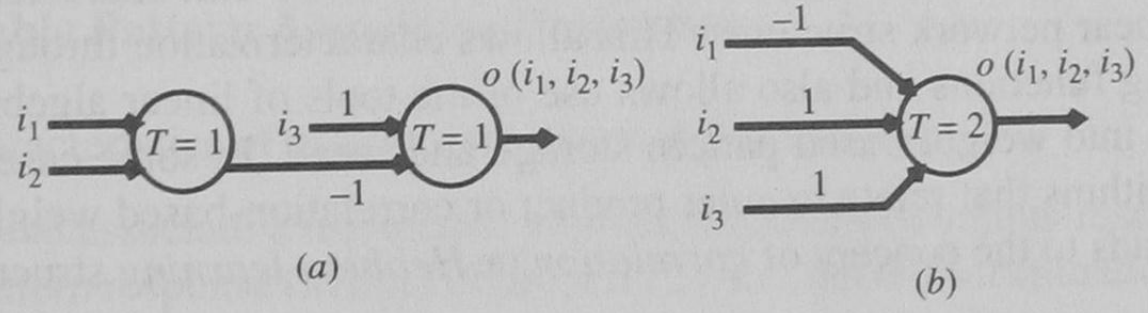


FIGURE P4.25
Logic functions using MP units.