CHAPTER 5

Introduction to Neural Mappings and Pattern Associator Applications

It's a poor sort of memory that only works backwards.
Lewis Carroll

In this chapter, approaches to developing *trainable mapping networks* are continued. The emphasis moves from the characterization of single units to groups of units.[1] The units are arranged in a nonrecurrent topology, forming *layers of units*. Initially, our interest is centered on single-layer architectures with linear unit characteristics, which give rise to linear network structures. This allows characterization through analysis of linear mapping functions and also allows use of the tools of linear algebra to develop some insights into weight-based pattern storage and recall. In some cases, training or learning algorithms that relate to outer product or correlation-based weight formations result. This leads to the concept of *correlation* or *Hebbian learning* structures.

Notational changes

In this chapter we relax our notational constraints somewhat. Whereas typical inputs and outputs are denoted $\underline{i}$ and $\underline{o}$, respectively, we augment the notation slightly. Inputs to the network are denoted as a vector $\underline{i}$ (or $\underline{s}$, for "stimulus," or simply $\underline{x}$) and the corresponding desired output is denoted $\underline{o}$ (or $\underline{r}$, for "response," or simply $\underline{y}$).

5.1 NEURAL NETWORK–BASED PATTERN ASSOCIATORS

5.1.1 Black-Box Pattern Associator Structure

The ANN mapping network from a black-box or I/O point of view is shown in Figure 5.1. Recall that the trainable black-box approach does not require detailed knowledge of underlying models for the data (e.g., statistical characterizations) but merely involves

[1]However, we show a few examples using single units, for simplicity.

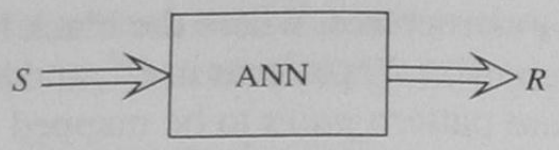


FIGURE 5.1
ANN pattern associator structure.

a chosen "internal" ANN structure (which may be invisible to the user) and a training method. The choice of a black-box structure still requires several ANN design considerations centering on parameters such as

1. The topology, or structure, of the internal network
2. The characteristics of the internal units
3. The appropriate formulation of inputs and outputs, as discussed in Chapter 1
4. The design of training or learning procedures

The careful design and characterization of desired network pattern association properties, generally through the design of H, is important. The structure should respond on the basis of stored patterns or stored pattern pairs. Given $\underline{x}_u$, *nearest neighbor (NN) recall* returns the stored pattern closest (using some measure of similarity) to $\underline{x}_u$. In *interpolative recall*, on the other hand, response is based on interpolation over all stored patterns.

5.1.2 Desirable Pattern Associator Properties

Desirable characteristics of pattern associators (PAs) include

1. The ability to associate a reasonable number of pattern (stimulus-response) pairs
2. Correct pattern response (in light of property 1), i.e., good discrimination ability with large stimulus-response set storage)
3. Trainability, given H (supervised learning possible)
4. Self-organization (given H_u, the network determines "natural" data clusters)
5. The ability to associate or generate the correct output(s) (response) when the input pattern is a distorted or incomplete version of that used in training

5.1.3 Autocorrelator versus Heterocorrelator Structures

The ANN stimulus is represented by a vector $\underline{x}$ and the desired response by $\underline{x}_d$. The desired ANN mapping is then formulated as

$$\underline{f}_D : \underline{x} \rightarrow \underline{x}_d \tag{5.1}$$

or

$$\underline{x}_d = \underline{f}_D(\underline{x}) \tag{5.2}$$

Specifying a set of states $\underline{x}_d$ or I/O pairs $(\underline{x}, \underline{x}_d)$ enables the design of $\underline{f}_D$, as well as the ANN. The ANN structure shown in Figure 5.1 may be visualized as implementing $\underline{f}_D$. The formulation of Equation (5.2) does not require $\underline{x}$ and $\underline{x}_d$ to be in the same vector space; rather, they may have different dimensions. Therefore, a distinction may

be made between *autocorrelator* (or *auto-associative*) structures, where the black box or content-addressable memory simply stores (or "memorizes") patterns in R^d, and *heterocorrelator* (or *hetero-associative*) structures, where pattern *pairs* to be mapped are stored in the network.

In autocorrelator structures, a set of d-dimensional patterns $\underline{x}_k$, $k = 1, 2, \ldots, n$, is encoded in the PA. Given an input, denoted $\underline{x}_u$, the PA is expected to return $\underline{x}_i$, the stored input closest to $\underline{x}_u$. Thus, auto-associative structures are commonly used for input recollection, input correction, and input completion.

In contrast, hetero-associative structures encode pattern pairs $(\underline{x}_k, \underline{y}_k)$, $k = 1, 2, \ldots, n$, where $\underline{y}_k$ is the desired response (also denoted $\underline{r}_k$) for input $\underline{x}_k$ (also denoted $\underline{s}_k$). Since the desired response may be a pattern class, hetero-associative structures are commonly used in pattern recognition for classification.

5.2 THE INFLUENCE OF PSYCHOLOGY ON PA DESIGN AND EVALUATION

Many of the concepts popularized by ANN research, especially those related to connectionist computing and associative behavior of networks, were proposed by noted psychologist William James in 1890 [Jam90]. James studied and formulated a number of hypotheses regarding the order of ideas in thought. Specifically, he was curious as to how the sequence of mental imagery in a temporal sample of thought could be explained. Clearly, this sequence contains transitions between concepts. Some are very smooth or continuous; others are quite abrupt. In many cases the thought path contains logical *links*. Many of James's assertions suggest a knowledge representation that is "connectionist" in nature. We explore a few highlights of his theory here; the reader should return to these ideas when exploring the recurrent and self-organizing structures of Chapters 8 and 9.

5.2.1 Discrimination, Association, and Principles of Connection

James postulated that the two fundamental operations involving thought are *discrimination* and *association* and that *principles of connection* explain the succession or coexistence of ideas in mental imagery. For example, a common subpath in thought might be expressed as

> Thinking about A leads to thinking about B, where A and B are concepts.

Concepts A and B are somehow connected. James explained this as a result of training or experience, which is learned, and postulated that

- There is no elementary causal law of association other than the *law of neural habit:*

  > When two elementary brain-processes have been active *together* or in *immediate succession* (emphasis added), one of them, on re-occurring, tends to propagate its excitement into the other.

- An extension of the elementary principle for *many processes* (each process excited by conjunction of many others) is the following corollary:

  The amount of activity at any given point in the brain-cortex is the sum of the tendencies of all other points to discharge into it, such tendencies being proportionate:

  1. to the number of times the excitement of each other point may have accompanied that of the point in question;
  2. to the intensity of such excitements; and
  3. to the absence of any rival point functionally disconnected with the first point, into which the discharges might be diverted.

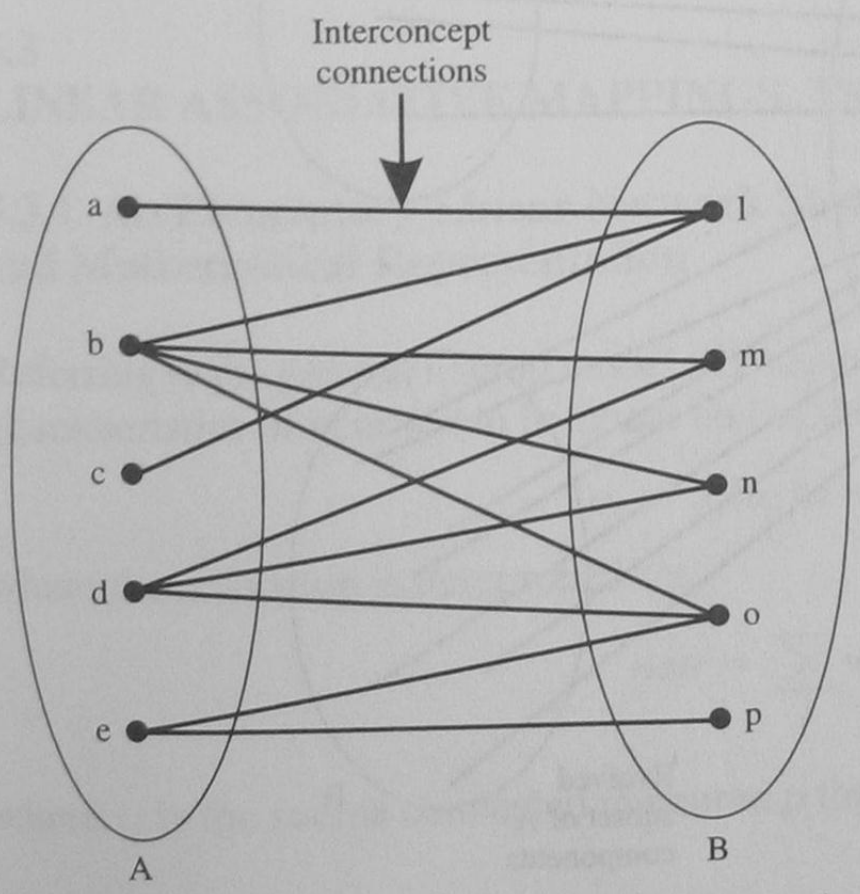
Items 1–3 are closely related to the notion of Hebbian or correlation-based learning, described in Sections 5.3.4 and 5.4.

5.2.2 Relevant Principles of Association

Conceptual (temporal) links

James postulated that the structure of memory processes involved inter- (activation) and intra- (resonance) concept connections. Sample interconcept connections and intraconcept "nerve tracts" are shown in Figure 5.2 . For example, notice that

- Concept B nerve tract l is excited by elementary nerve tracts a, b, and c in concept A.
- Nerve tract o in concept B is excited by concept A nerve tracts b, d, and e.



a, b, c, d, e: "Elementary nerve tracts" excited by concept A
l, m, n, o, p: "Nerve tracts" for concept B

FIGURE 5.2
Links between concepts A and B.

In addition, nerve tracts a, b, c, d, e and tracts l, m, n, o, p also "vibrate in unison" or resonate *within the concept*. This viewpoint will become significant in our exploration of BAM in Chapter 8.

Total recall

Total recall is a process where the mind is in a "perpetual treadmill" of reminiscences with perfect detail. All corresponding nerve tracts for an active concept are excited. The sequence of concepts that is mentally replayed is determined by the interconnections and initial resonances. Total recall is not a plausible model for thinking, except in cases of mental disorder [Jam90].

Partial recall

Partial recall is a powerful basis for learning/forgetting and for recall and association models in ANN structures, and leads to implementations involving self-adaptation. The most important characteristics of the concept of partial recall, are [Jam90]

> In no revival of a past experience are all the items of our thought equally operative in determining what the next thought shall be. Always some ingredient is prepotent over the rest.
>
> Prepotent items are those which appeal most to our *interest*.

These observations suggest that stored representations are time-varying. While one part of a process (concept A or B) may be fading, decaying, and becoming indistinct (losing

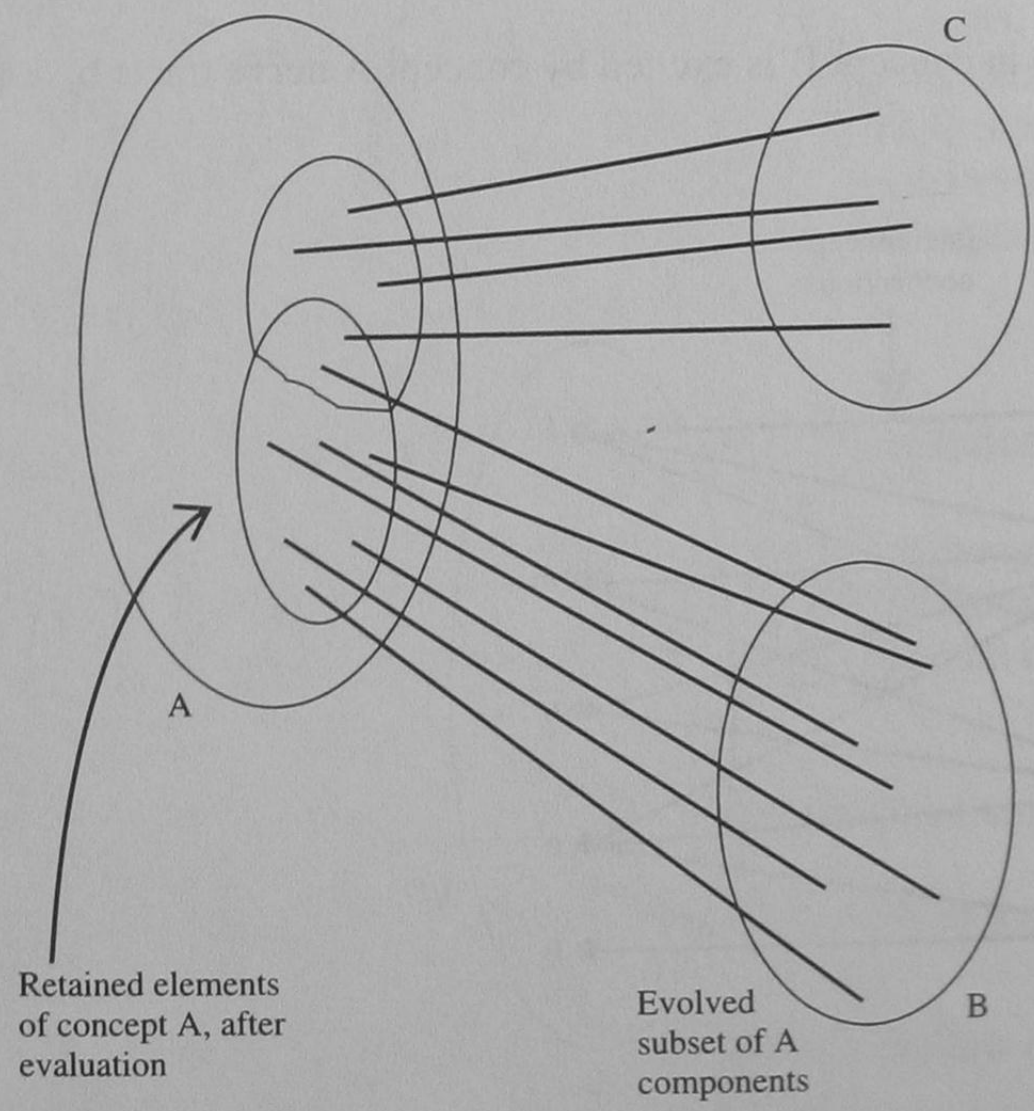


FIGURE 5.3
Partial recall.

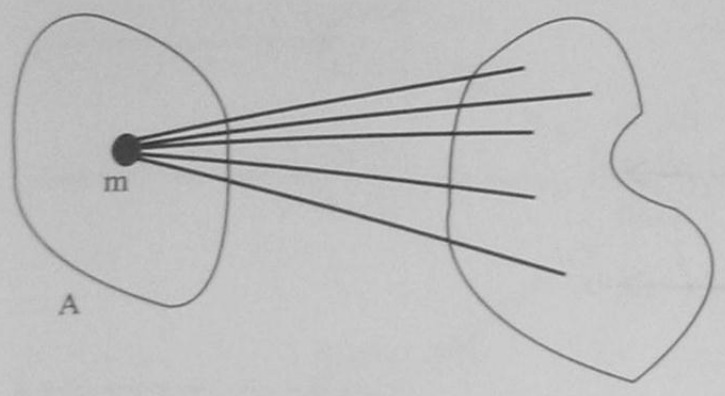


FIGURE 5.4
Focalized recall, where an individual component of concept A is used.

vividness), another part of the same process (that possesses a strong internal *interest*) resists this tendency, becoming relatively stronger. In other words, an internal representation *evolves* (Figure 5.3) and certain portions become *dominant*.

Focalized recall

Partial recall leads to *focalized recall* or *association by similarity*, as shown in Figure 5.4. In focalized recall, one entity (nerve tract) invokes an entire other concept; i.e., not all entities are needed to address or invoke a concept.

Summary and utility

Over time (experience) concepts are modified by evolution and interest. Some entities become dominant, where dominance of a concept is based on *reinforcement* (Hebbian learning). Most important, for our design of ANNs,

1. Figure 5.2 suggests a network architecture or structure.
2. Figures 5.3 and 5.4 suggest an approach for implementing recall.

5.3 LINEAR ASSOCIATIVE MAPPINGS, TRAINING, AND EXAMPLES

5.3.1 An Elementary Linear Network Structure and Mathematical Representation

Referring to the general neuron model of Chapter 3, suppose we choose the input/output characteristics of neuron i to be linear and of the form

$$o_i = f_i(net_i) = net_i \tag{5.3}$$

where the activation is determined by

$$net_i = \sum_j w_{ij} i_j \tag{5.4}$$

where i_j is the source connected to neuron p through weight w_{pj}.[2]

[2]In what follows we will be more specific about the origin of the layer input vector $\underline{i}$.

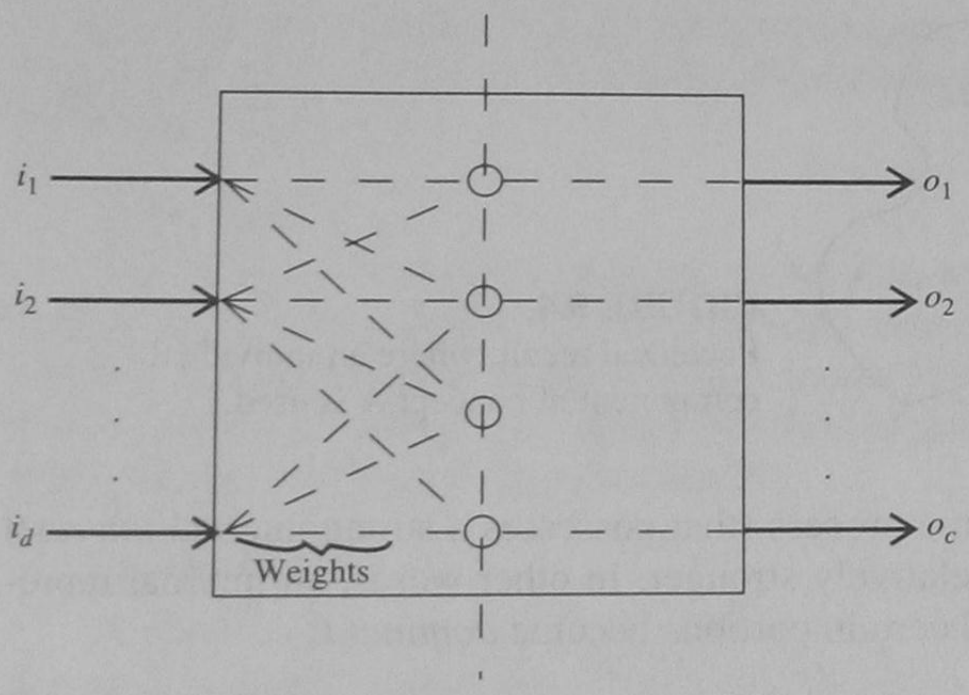


FIGURE 5.5
A layer of mapping or PA units.

Prior to further mathematical characterization, we impose the following conditions on a single neuron layer:

1. All units share the same inputs.
2. All units have independent weights (and possibly activation functions).

Thus, a layer of units composing an elementary black box is shown in Figure 5.5.

More specifically, for a network with c (output) units and d inputs, Equations (5.3) and (5.4) may be cast as

$$o_i = \langle \underline{w}_i, \underline{i} \rangle \qquad i = 1, 2, \ldots, c \tag{5.5}$$

where $\underline{w}_i$ is a $d \times 1$ column vector for the ith unit weights with respective elements w_{ij}, and $\underline{i}$ is a $d \times 1$ vector of inputs, the jth of which is denoted i_j. This is shown in Figure 5.6.

For c outputs, Equation (5.5) may be formulated as

$$\underline{o} = W\underline{i} \tag{5.6}$$

Using our alternative stimulus-response notation, Equation 5.6 may be written

$$\underline{r} = W\underline{s} \tag{5.7}$$

where $\underline{o}$ (or $\underline{r}$) is $c \times 1$, $\underline{i}$ (or $\underline{s}$) is $d \times 1$, and W is a $c \times d$ matrix of the form

$$W = [w_{ij}] = [\underline{w}_i^T] \tag{5.8}$$

i.e., the rows of W are the individual unit weight vectors.[3] This is shown in Figure 5.6.

A general formulation of Equation (5.7) using the n elements of H yields

$$W[\underline{s}^1 \ \underline{s}^2 \ \ldots \ \underline{s}^n] = [\underline{r}^1 \ \underline{r}^2 \ \ldots \ \underline{r}^n] \tag{5.9}$$

or

$$WS = R \tag{5.10}$$

[3]The reader should expect to see alternative formulations and definitions related to this; however, they are merely matrix-transposed versions.

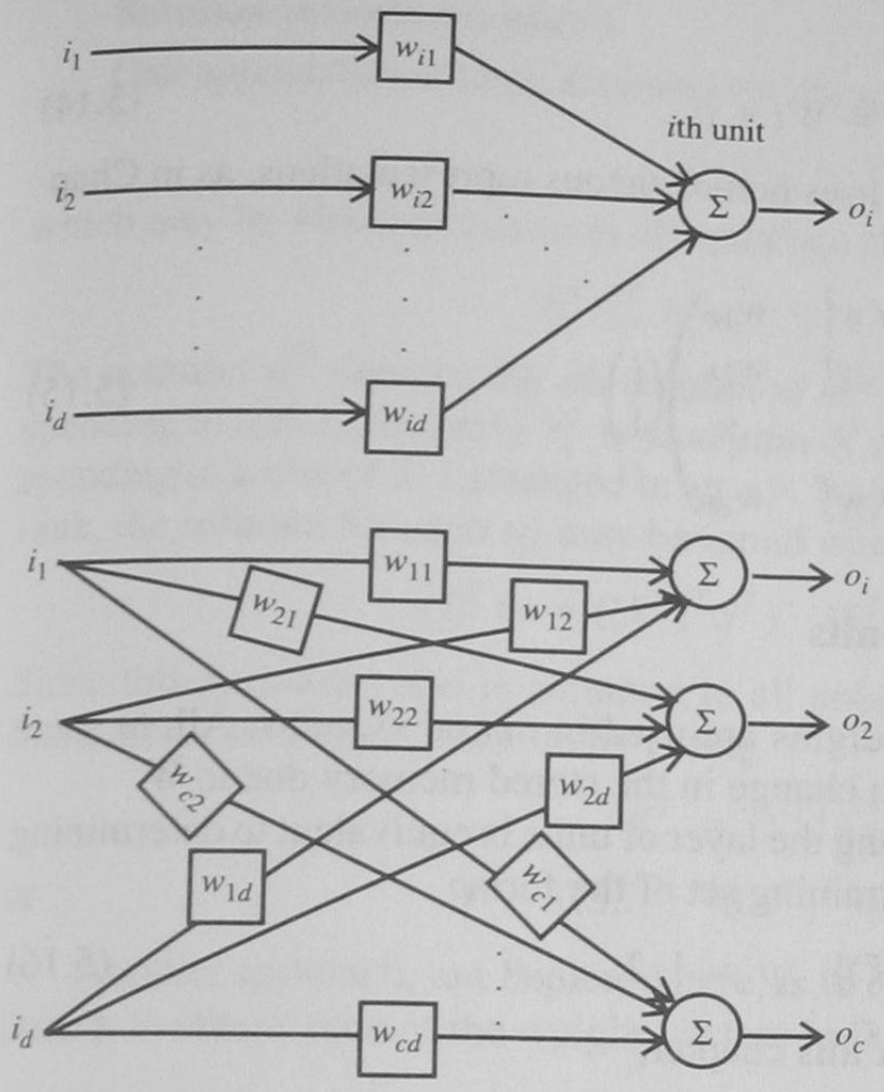


FIGURE 5.6
Linear model development: From individual units to a linear layer.

where W is a $c \times d$ matrix whose ith row is the respective set of weights for unit i, S is $d \times n$, and R is $c \times n$.

Keeping the unit output characteristic of Equation (5.3), but adding a bias to each unit yields

$$o_i = net_i = \sum_j w_{ij} i_j + b_i = \underline{w}_i^T \underline{i} + w_{ib} i_b \tag{5.11}$$

Expanding Equation (5.11) for a layer of c units yields

$$\underline{o} = \begin{pmatrix} \underline{w}_1^T \\ \underline{w}_2^T \\ \underline{w}_3^T \\ \vdots \\ \underline{w}_c^T \end{pmatrix} \underline{i} + \underline{b} \tag{5.12}$$

where

$$\underline{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_c \end{pmatrix} \tag{5.13}$$

This yields the matrix formulation

$$\underline{o} = W\underline{i} + \underline{b} \tag{5.14}$$

Another, equivalent approach that uses homogeneous representations, as in Chapters 3 and 4, is

$$\underline{o} = \begin{pmatrix} \underline{w}_1^T & w_{1b} \\ \underline{w}_2^T & w_{2b} \\ \vdots & \vdots \\ \underline{w}_c^T & w_{cb} \end{pmatrix} \begin{pmatrix} \underline{i} \\ 1 \end{pmatrix} \tag{5.15}$$

5.3.2 Training a Single Layer of Units

A number of learning or training paradigms are used in neural systems. All, to some extent, involve learning or training as a change in the stored memory due to W.

Referring to Equation (5.15), training the layer of units is equivalent to determining W and $\underline{b}$. Assume that we are given a training set of the form

$$H = \{(\underline{i}^p, \underline{t}^p)\} \quad p = 1, 2, \ldots, n \tag{5.16}$$

or, in the stimulus-response notation of this chapter,

$$H = \{(\underline{s}^p, \underline{r}^p)\} \quad p = 1, 2, \ldots, n \tag{5.17}$$

5.3.3 Multiple Layers (Linear Units)

Notice that Equation (5.6) may be used to define a multilayer network where, in the two-layer case, the second layer input $\underline{i}_2$ is the output of the first layer defined by Equation (5.6); i.e., $\underline{i}_2 = \underline{o}$, and the overall mapping is

$$\underline{o}_2 = W_2\underline{i}_2 = W_2W_1\underline{i} \tag{5.18}$$

Matrices W_1 and W_2 need not have the same dimension. In Chapter 6 multiple nonlinear layers that implement arbitrary mappings are considered.

5.3.4 Hetero-associators Directly from Generalized Inverses

Suppose the linear hetero-associative network structure is formulated as follows:

$$W\underline{s}^p = \underline{r}^p \qquad p = 1, 2, \ldots, n \tag{5.19}$$

where $\underline{r}^p$ is the $c \times 1$ response vector, $\underline{s}^p$ is a $d \times 1$ stimulus vector, and W is the $c \times d$ matrix that represents the linear network interconnection structure. Recall that the ith row of W represents the weights of unit i. The problem is determining W, especially when the training set $H = \{(\underline{s}^p, \underline{r}^p)\}$ is arbitrary. Using H, the formulation of Equation (5.10) yields:

$$WS = R \tag{5.20}$$

Solution procedure, part I

One approach to solving Equation (5.20) is to note that it may be rewritten as

$$S^T W^T = R^T \tag{5.21}$$

which may be visualized as a set of equations of the form

$$S^T \underline{w}_i^T = \underline{r}_i^T \quad i = 1, 2, \ldots, c \tag{5.22}$$

The notation $\underline{w}_i^T$ denotes the ith column of W^T, which consists of the weights corresponding to unit i. Similarly, $\underline{r}_i^T$ is a column of desired responses for unit i (each corresponding to a row of S^T) arranged in an $n \times 1$ vector. Assuming that S^T has full column rank, the solution for each $\underline{w}_i$ may be found using the pseudoinverse of S^T as

$$(S^T)^\dagger = ((S^T)^T S^T)^{-1}(S^T)^T = (SS^T)^{-1}S \tag{5.23}$$

Since this pseudoinverse is common to all unknown vectors $\underline{w}_i$ in Equation (5.22), a more direct solution is possible by noting

$$W^T = (SS^T)^{-1}SR^T \tag{5.24}$$

or

$$W = ((SS^T)^{-1}SR^T)^T = RS^T(SS^T)^{-1} \tag{5.25}$$

Another approach, not explored here, is to employ the gradient descent–based approach to obtain each of the weight vectors in Equation (5.22).

Solution procedure, part II

Assuming that S^T has full row rank,[4] Equation (5.20) is postmultiplied by the pseudoinverse of S, denoted $S^\dagger$, which yields

$$RS^\dagger = WSS^\dagger = W \tag{5.26}$$

or

$$W = RS^\dagger = RS^T(SS^T)^{-1} \tag{5.27}$$

Extensions and modifications to the basic procedure

When the $d \times n$ stimulus matrix S^T in Equation (5.10) or (5.20) does not have full column rank (i.e., S does not have full row rank), there are not n linearly independent columns in S^T, and the pseudoinverse procedure of Equation (5.27) breaks down. One way to overcome this problem is to use *dummy augmentation* [WCM90] on each of the $d \times 1$ stimulus vectors $\underline{s}^i$. Additional components ($d + 1, d + 2$, etc.) are added to each stimulus vector in S, which has the effect of adding columns to S^T.

Simplification for orthogonal S

Equation (5.27) is simplified considerably when S (the matrix whose columns are the stimulus vectors) is orthogonal.[5] In this case, it is easy to show

$$S^\dagger = S^{-1} = S^T \tag{5.28}$$

---

[4] The reader should verify this.

[5] Actually, it is considerably simplified when S is just invertible.

so the weight matrix is formed using

$$W = RS^T \tag{5.29}$$

Expanding Equation (5.29) yields

$$W = [\underline{r}^1 \ \underline{r}^2 \ \dots \ \underline{r}^n] \begin{pmatrix} (\underline{s}^1)^T \\ (\underline{s}^2)^T \\ \vdots \\ (\underline{s}^n)^T \end{pmatrix} \tag{5.30}$$

Equation (5.30) may be expanded to show

$$W = [\underline{r}^1 \underline{r}^2 \dots \underline{r}^n] \begin{pmatrix} (\underline{s}^1)^T \\ (\underline{s}^2)^T \\ \vdots \\ (\underline{s}^n)^T \end{pmatrix} = (\underline{c}_1 | \underline{c}_2 | \dots | \underline{c}_n) \tag{5.31}$$

where column $\underline{c}_j$ is formed from

$$\underline{c}_j = \sum_{i=1}^{n} \underline{r}^i s_{ij} \tag{5.32}$$

and s_{ij} is the jth element of row vector $(\underline{s}^i)^T$. For example, the first column, $\underline{c}_1$, of the resultant matrix in Equation (5.31) is

$$\underline{c}_1 = \underline{r}^1 s_{11} + \underline{r}^2 s_{21} + \cdots + \underline{r}^n s_{n1} \tag{5.33}$$

The general form of Equation (5.31) is therefore

$$[\underline{r}^1 \ \underline{r}^2 \ \dots \ \underline{r}^n] \begin{pmatrix} (\underline{s}^1)^T \\ (\underline{s}^2)^T \\ \vdots \\ (\underline{s}^n)^T \end{pmatrix} = (\textstyle\sum_{i=1}^{n} \underline{r}^i s_{i1} | \sum_{i=1}^{n} \underline{r}^i s_{i2} | \dots | \sum_{i=1}^{n} \underline{r}^i s_{in}) \tag{5.34}$$

Therefore, a special form for W exists in this case:

$$W = \sum_{i=1}^{n} \underline{r}^i (\underline{s}^i)^T \tag{5.35}$$

This reduces to an outer product formulation and should be compared with Equation (5.54), derived in a later section.

5.3.5 Extended Examples: Hetero-associative Memory Design

Example 1: Three-input, 1-of-3 output

Problem formulation. Consider the following $n = 3$ training set, consisting of $d = 3$ S-R pairs $(\underline{s}^i, \underline{r}^i)$ where

$$\underline{s}^1 = \begin{pmatrix} 4 \\ 1 \\ 1 \end{pmatrix} \qquad \underline{r}^1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \tag{5.36}$$

$$\underline{s}^2 = \begin{pmatrix} 4 \\ 1 \\ 0 \end{pmatrix} \qquad \underline{r}^2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \tag{5.37}$$

$$\underline{s}^3 = \begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix} \qquad \underline{r}^3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{5.38}$$

Clearly, the form of the desired output of the PA is a 1-of-3 selector, where in this $c = 3$ class example element $r_i = 1$ corresponds to the (binary) classification of stimulus $\underline{s}^k$ to class w_i. Equation (5.20) becomes

$$R = {}^{3\times 3}I \tag{5.39}$$

and

$$S = (\underline{s}^1 \ \underline{s}^2 \ \underline{s}^3) = \begin{pmatrix} 4 & 4 & 2 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \tag{5.40}$$

Since S has full column rank and is of dimension 3×3, S is invertible (the reader should verify this). Using Equation (5.28), W becomes

$$W = S^{-1} = \begin{pmatrix} -\frac{1}{2} & 2 & 1 \\ \frac{1}{2} & -1 & -1 \\ \frac{1}{2} & -2 & 0 \end{pmatrix} \tag{5.41}$$

Verification of recall properties. The reader should verify that, for the S-R pairs given in Equations (5.36) to (5.38), the trained interconnection matrix of Equation (5.41) yields

$$W\underline{s}^i = \underline{r}^i \quad i = 1, 2, 3 \tag{5.42}$$

It is interesting to consider the response of the network for both the training set and perturbed stimulus patterns. For example, given a perturbed version of $\underline{s}^1$, denoted $\underline{s}^p$, where

$$\underline{s}^p = \begin{pmatrix} 4 \\ 1 \\ 2 \end{pmatrix} \tag{5.43}$$

the response is

$$\underline{r}_p = W\underline{s}_p = \begin{pmatrix} 2 \\ -1 \\ 0 \end{pmatrix} \tag{5.44}$$

Clearly, the network displays an interpolative form of response. This example is continued in the exercises.

Example 2: Linear 2-of-3 detector and variations

Problem formulation. Consider the development, via linear units, of a 2-of-3 input active detector. A single-unit architecture is used. The input/output relationship for such a device is shown in Table 5.1. The reader should compare the characteristic of Table 5.1 with that of a parity detector or generator. We should also note that in this problem *the training set is exhaustive*; i.e., there are no other input combinations to consider, so that generalization is not an issue. We will consider several variations on this problem, including incorporation of a bias and modification of the desired output characteristic.

Linear separability concerns. The use of a linear unit to accomplish the specified mappings again raises the issue of linear separability. Using the techniques of Chapter 4, it is possible to show that H represents a nonlinearly separable data set. This is illustrated graphically in Figure 5.7.

MATLAB formulation (no bias). The desired mapping of Table 5.1 is rewritten in the form of Equation (5.20), or simply

TABLE 5.1
Characteristic for (exactly) 2-of-3 detector

Input (i_1 i_2 i_3)			Output (o_1)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

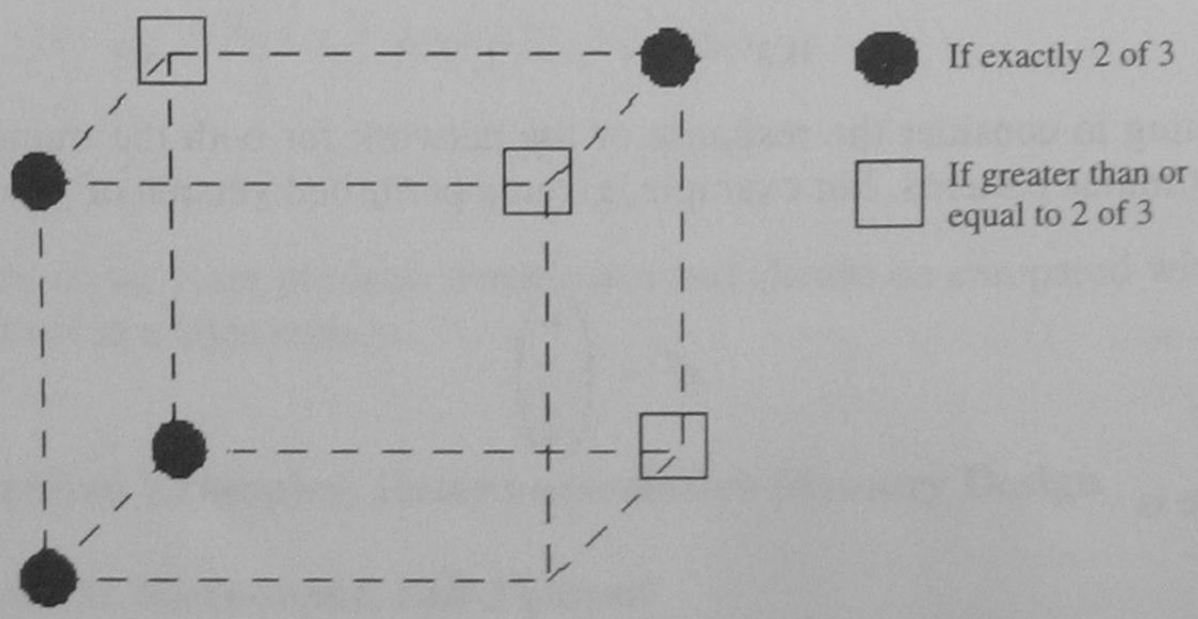


FIGURE 5.7
Mapping of 2-of-3 example.

$$\underline{w}^T S = \underline{r}^T \tag{5.45}$$

where each column of S corresponds to a row of Table 5.1. The MATLAB formulation and solution is shown below.

```
S =

     0     0     0     0     1     1     1     1
     0     0     1     1     0     0     1     1
     0     1     0     1     0     1     0     1

R =

     0     0     0     1     0     1     1     0

 W=R*pinv(S)

W =

    0.2500    0.2500    0.2500
```

To check the resultant mapping, an error vector is computed. First we determine the actual response to all elements of H.

```
 W*S

ans =

0    0.2500    0.2500    0.5000    0.2500    0.5000    0.5000    0.7500
```

Notice that this result is significantly different from that specified in Table 5.1; in fact, the mapping achieved with a single linear unit and without bias is quite poor. Quantitatively,

```
 error=R-W*S

error =

  0   -0.2500   -0.2500    0.5000   -0.2500    0.5000    0.5000   -0.7500

 e=errnorm2(error)

err2 =

    1.5000

e =

    1.5000
```

Using a unit bias. As an attempt at achieving a better mapping, we consider the addition of a bias, here implemented by augmenting each input vector with a $(d + 1)$th

input whose value is 1. The correspond $(d + 1)$th unit weight is therefore the unit bias. The revised MATLAB formulation is as follows.

```
S =

     0     0     0     0     1     1     1     1
     0     0     1     1     0     0     1     1
     0     1     0     1     0     1     0     1

 R

R =

     0     0     0     1     0     1     1     0


 B=[1 1 1 1 1 1 1 1]

B =

     1     1     1     1     1     1     1     1

 SB=[S;B]

SB =

     0     0     0     0     1     1     1     1
     0     0     1     1     0     0     1     1
     0     1     0     1     0     1     0     1
     1     1     1     1     1     1     1     1

 WB=R*pinv(SB)

WB =

    0.2500    0.2500    0.2500   -0.0000
```

It is left for the reader to evaluate the resulting mapping and compare it with that of the nonbiased formulation.

Example 3: Gradient descent solution to Example 2

We formulate the problem as

$$S^T \underline{w}^T = R^T \tag{5.46}$$

and solve for $\underline{w}$ using a gradient approach. After 15 iterations, with the gain value $\alpha = 0.1$ and $\underline{w}^{(0)} = (-1\ 0\ 1)^T$, the results are

```
gradient =

   -0.0078
   -0.0047
   -0.0016
```

```
w=

    0.3747
    0.3752
    0.3756
```

Notice that the values of the gradient are approaching the vector $\underline{0}$, indicating that a local minimum in the error function has been reached. The reader should verify the resulting solution. The success of the gradient solution, however, depends on the suitable choice of $\underline{w}^{(0)}$ and ρ.

Example 4: The A/D converter

Overview and training set, H. D/A converter implementation was addressed in Chapter 4. Here we consider the development of a mapping network to achieve the inverse of this mapping, namely, from an analog input to a binary output representation. The user is encouraged to review "classical" solutions[6] to this problem. The training set for a 2-bit A/D converter is shown in Table 5.2. It is left for the reader to consider alternative formulations, for example, cases where the output representation is centered about the midpoint of the input interval.

TABLE 5.2
Training data for desired A/D converter

Input (i_1)	Output (o_1 o_2)	
0.0	0	0
0.25	0	1
0.50	1	0
0.75	1	1

Two-unit A/D formulation without unit bias. A MATLAB solution follows:

```
S =

         0    0.2500    0.5000    0.7500

 R

R =

     0     0     1     1
     0     1     0     1

 W=R*pinv(S)
```

[6]For example, the design of successive approximation and flash converters.

```
W =

    1.4286
    1.1429

M=W*S

M =

         0    0.3571    0.7143    1.0714
         0    0.2857    0.5714    0.8571

e1=R-M

e1 =

         0   -0.3571    0.2857   -0.0714
         0    0.7143   -0.5714    0.1429
```

The results show that even with use of a nonlinear element (e.g., a threshold unit) to "clean up" the analog output, a perfect mapping is not possible. We explore the fundamental reason for this later.

A/D formulation using a unit bias. The MATLAB results are as follows:

```
SB =

         0    0.2500    0.5000    0.7500
    1.0000    1.0000    1.0000    1.0000

WB=R*pinv(SB)

WB =

    1.6000   -0.1000
    0.8000    0.2000


MB=WB*SB

MB =

   -0.1000    0.3000    0.7000    1.1000
    0.2000    0.4000    0.6000    0.8000

e2=MB-R

e2 =

   -0.1000    0.3000   -0.3000    0.1000
    0.2000   -0.6000    0.6000   -0.2000
```

A/D converter formulation using additional inputs. To see the fundamental problem with a single-layer implementation of the A/D, consider the linear separability of each of the desired output mappings. Clearly, the desired mapping for response r_1 is linearly separable, and that for r_2 is not. To solve the problem, suppose we add a third input to the network formed by thresholding output o_1. Two consequences are

- This yields an exact solution.
- This yields what may be visualized as a two-layer network, with each layer consisting of a single unit.

A MATLAB solution follows. First, the stimulus matrix is formed:

```
SBr2=[SB; 0 0 1 1]

SBr2 =

         0    0.2500    0.5000    0.7500
    1.0000    1.0000    1.0000    1.0000
         0         0    1.0000    1.0000
```

Note that the rows of **SBr2** correspond to the input, bias, and "added input," respectively. The solution yields

```
WBr2=R*pinv(SBr2)

WBr2 =

   -0.0000    0.0000    1.0000
    4.0000   -0.0000   -2.0000

WBr2*SBr2

ans =

    0.0000   -0.0000    1.0000    1.0000
   -0.0000    1.0000   -0.0000    1.0000
```

This result indicates that a perfect mapping is possible with this formulation.

5.3.6 Training Nonlinear Mappings Using Linear Solution Techniques

The presence of a nonlinear activation function, with WLIC formation of net_i, does not necessarily require a nonlinear solution formation. If the nonlinear activation function is invertible, a linear form of the input to net_i mapping is possible and may be used to determine W. For example, suppose the nonlinear activation function is the sigmoid, as shown in Figure 5.8. Specification of o_i allows determination of the corresponding net_i, and therefore the input to net_i mapping is linear. The overall structure for this approach is shown in Figure 5.9.

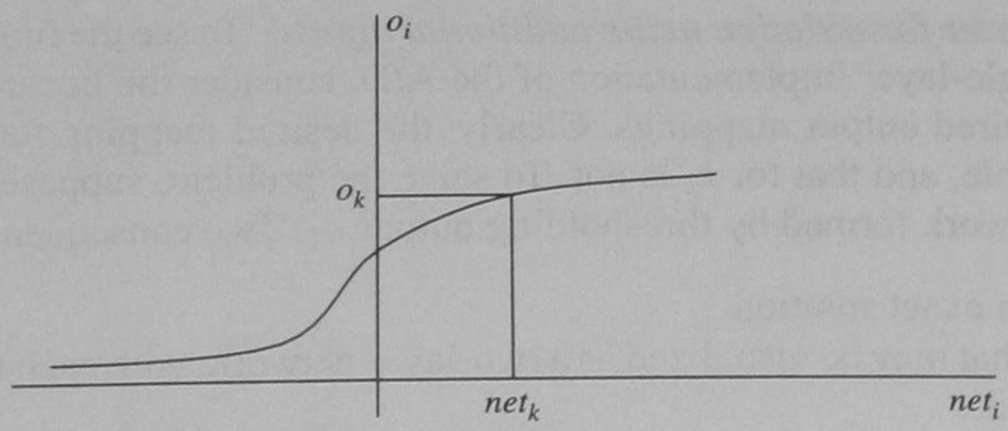


FIGURE 5.8
Inverting the sigmoid activation function to allow a linear solution.

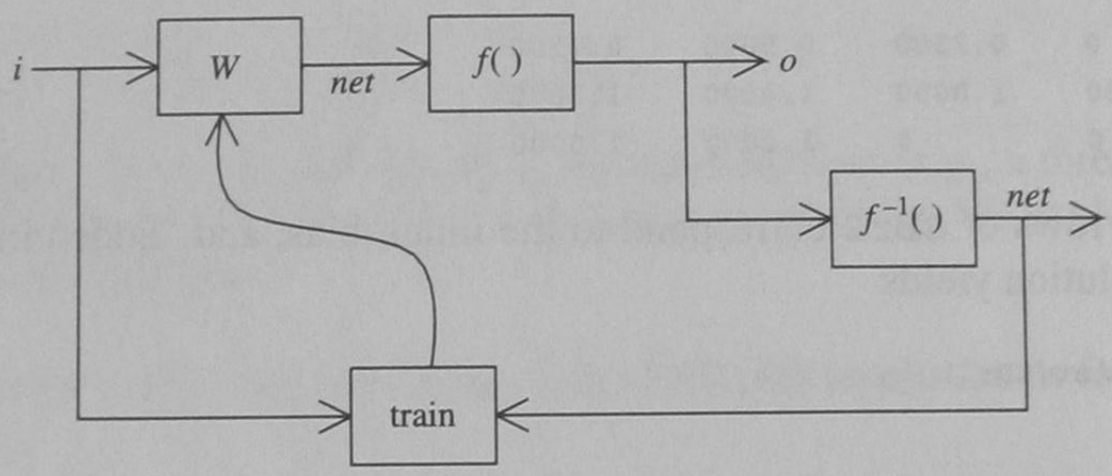


FIGURE 5.9
Overall solution strategy for using a linear training algorithm with a single layer of nonlinear (invertible) units.

5.4 HEBBIAN OR CORRELATION-BASED LEARNING

Several of the previous approaches have relied on outer product or correlation techniques for weight determination. For example, recall the results of Section 5.3.4, where special forms for S led to an outer product–based training algorithm. In this section we develop a procedure in which the values of s_i and r_j are used directly to form W.

5.4.1 A Storage Prescription for a Single Pattern

Consider a single stimulus-response pair $(\underline{s}, \underline{r})$ with the $d \times 1$ stimulus vector $\underline{s}$ normalized to unity length, i.e., $\|\underline{s}\| = \sqrt{\underline{s}^T \underline{s}} = 1$. The formation of a weight matrix using the following form is proposed:

$$W = \underline{r}\,\underline{s}^T \tag{5.47}$$

Recall that $\underline{r}$ is $c \times 1$; therefore, W is a $c \times d$ (rank 1) matrix formed via the outer-product operation of Equation (5.47). The ijth element of W, denoted w_{ij}, is a weight

value formed by the product $r_i s_j$, where

$$\underline{s} = \begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_d \end{pmatrix} \tag{5.48}$$

and

$$\underline{r} = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_c \end{pmatrix} \tag{5.49}$$

The sign of element w_{ij} depends on the signs of s_j and r_i. Like signs cause the interconnection strength to be positive, whereas opposite signs lead to a w_{ij} whose sign is negative. Another interpretation is that positive weights result from Equation (5.47) when the input-output values have a positive correlation, and negative interconnection strengths result from negatively correlated input-output values in H.[7]

5.4.2 Assessment of Storage Properties

For an arbitrary stimulus vector, Equation (5.47) is used to find the network response. In the case that $\underline{s}$ is the vector used to train the system in Equation (5.47) by forming the matrix product

$$W\underline{s} = \underline{r}\underline{s}^T\underline{s} = \|\underline{s}\|^2\underline{r} = \underline{r} \tag{5.50}$$

the correct response $\underline{r}$ is generated. If a stimulus vector $\underline{s}'$ is a distorted version of $\underline{s}$, the linear network response is

$$W\underline{s}' = \underline{r}\underline{s}^T\underline{s}' = (\underline{s}^T\underline{s}')\underline{r} \tag{5.51}$$

where the term $\underline{s}^T\underline{s}'$ represents a weight that causes the network output to be a scaled version of the trained response. For normalized $\underline{s}$ and $\underline{s}'$, the reader should verify from the cosine inequality that

$$\underline{s}^T\underline{s}' < \|\underline{s}\|^2 \tag{5.52}$$

and therefore the weight or scaling of the training set response $\underline{r}$ in Equation (5.50) to a distorted stimulus vector is predictable and given by

$$\underline{s}^T\underline{s}' < 1 \tag{5.53}$$

This indicates the generalization capability of this type of ANN mapping network and learning algorithm.

---

[7]This is explored further in Chapter 8.

It is desirable to be able to use Equations (5.47) and (5.50) to store several $(\underline{s}^i, \underline{r}^i)$ pairs, $i = 1, 2, \ldots, n$, from H. Consider forming W for the n-pair storage case by superposition of the single-pair case of Equation (5.47), i.e.,

$$W = \sum_{i=1}^{n} \underline{r}^i (\underline{s}^i)^T \tag{5.54}$$

5.4.3 Alternative Formulation for the Hebbian Prescription

An alternative to the form of Equation (5.54) is the form

$$W = RS^T \tag{5.55}$$

Given an input stimulus, denoted $\underline{s}^u$, the response is computed from

$$W\underline{s}^u = \sum_{i=1}^{n} \underline{r}^i (\underline{s}^i)^T \underline{s}^u = \sum_{i=1}^{n} [(\underline{s}^i)^T \underline{s}^u] \underline{r}^i \tag{5.56}$$

Notice that n terms contribute to the output in Equation (5.56). These may be rewritten in the form

$$W = \sum_{i=1}^{n} \underline{r}^i (\underline{s}^i)^T = \sum_{i=1}^{n} W_i \tag{5.57}$$

where W_i corresponds to the outer-product formulation matrix used to store training set pattern pair $(\underline{s}^i, \underline{r}^i)$. Suppose the desired response to input $\underline{s}^u$ is $\underline{r}^p$; i.e., $\underline{s}^u$ corresponds to $\underline{s}^p$ in "stored" association $(\underline{s}^p, \underline{r}^p)$. A desirable characteristic in Equation (5.56) is

$$\langle \underline{s}^i, \underline{s}^p \rangle = \delta_{ip} = \begin{cases} 1 & \text{for } i = p \\ 0 & \text{elsewhere} \end{cases} \tag{5.58}$$

where δ is the Kronecker delta. Equation (5.58) defines W as a set of n orthonormal vectors, which could be achieved using the Gram-Schmidt orthogonalization process. Unfortunately, since the $\underline{s}^i$ are input patterns or stimuli (and therefore not always controllable by the ANN system designer), this desirable orthogonality is difficult to obtain in a general problem.

5.4.4 Hebbian Example 1: Problem Formulation for Ternary-Valued Output (Single Pattern Pair)

Consider the following training set H, where a ternary-valued ($\{-1, 0, 1\}$ values) stimulus-response pair $(\underline{s}, \underline{r})$ is specified:

$$\underline{s} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} s_1 \\ s_2 \\ s_3 \end{pmatrix} \tag{5.59}$$

and

$$\underline{r} = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix} \tag{5.60}$$

Outer-product solution

For this pattern pair, the network weights, using Equation (5.47) or (5.30), are

$$W = [w_{ij}] = \underline{r}\,\underline{s}^T = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix} \tag{5.61}$$

Recalling from Section 5.4.1 how the w_{ij} are formed, note that W is also ternary valued. It is left for the reader to formulate and solve this problem using the pseudoinverse formulation and compare the results with those shown here using the outer-product or Hebbian approach.

Assessment of solution

In Equation (5.61), interconnection w_{11} is positive, indicating a (desired) positive correlation between s_1 and r_1. Conversely, w_{31} is negative, indicating a negative correlation or relationship between s_1 and r_3. From Equation (5.54), the overall network weight set is determined by summation of the individual weights. Thus, each weight component in the sum of Equation (5.57) of the form

$$\Delta w_{ij} = s_j \cdot r_i \tag{5.62}$$

where s_j and r_i are the jth and ith components of $\underline{s}$ and $\underline{r}$, respectively, represents an incremental portion of the overall network weight, w_{ij}. The overall weight is $w_{ij} = \sum_n \Delta w_{ij}$. Training set pattern pairs in which s_j and r_i are positive increase or reinforce w_{ij}. If many such patterns appear in H, then as w_{ij} is developed by sequentially processing H, the connection strength w_{ij} "builds up" or becomes large and positive. This is an example of *Hebbian learning*. A similar remark holds for a negative correlation between s_j and r_j.

5.4.5 Example 2: Three-Input, Three-Output PA Using Hebbian Training

This example uses the S-R pairs of Example 1 Section 5.3.5, where a pseudoinverse formulation was used. In addition, a problem with the scale of the W matrix obtained from the Hebbian prescription is introduced.

Using the data from Section 5.3.5, we form W using the following formulation. The reader should verify that this procedure is identical to that given in Equation (5.54).

```
R =

     1     2     3
     4     5     6
     7     8     9

S=[1 3 5; 5 3 1; 1 0 2]
```

```
W1=R *S'

W1 =

    22    14     7
    49    41    16
    76    68    25
```

The reader should compare a scaled version of this matrix with that of the pseudoinverse formulation.

One of the major concerns raised by the Hebbian formulation is shown in this example. Specifically, the size or scale of elements in H needs to be addressed. In the examples, the scale of W is obviously too large; some normalization is necessary. This is addressed in the problems.

5.5 BIBLIOGRAPHY

In this chapter, a number of elementary PA designs were considered. Pattern associator input and output representations have been shown to influence mapping and learning ability. A useful reference for general concerns in this area is [DB91].

The original works of William James are extensive; many libraries archive this set. *The Writing of William James* is an accessible version of James's work, published in 1977 by the University of Chicago Press (J. J. McDermottt, ed.). Dover reprinted [Jam90] in 1950.

An especially good modern introduction to the concept of a generalized inverse is [Str76]. A fast method of updating pseudoinverse-based associative memory solutions is shown in [TC94].

Hebbian or correlation-based learning seems to have originated with Hebb [Heb94]. A physiological mechanism that lends support to Hebb's neurophysiological postulate is described in depth in [Ste73]. Excellent sources for extensions and related work are [Koh72], [Koh84], and [BSA83].

A related strategy that achieves hetero-associative memory using a recurrent network of nonlinear (bipolar) elements is considered in Chapter 9, under the topic of bidirectional associative memory (BAM) [Kos87], [Kos88], [WCM90]. The basis of the BAM structure is Hebbian or correlation-based learning.

REFERENCES

[BSA83] A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics,* SMC-13:834–846, 1983.

[DB91] T. G. Dietterich and G. Bakiri. Error-correcting output codes: A general method for improving multiclass inductive learning programs. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91).* AAAI Press, Anaheim, CA, 1991.

[Heb49] D. Hebb. *Organization of Behavior.* John Wiley & Sons, New York, 1949.

[Jam90] William James. *Psychology (Briefer Course).* Holt, New York, 1890 (see Chapter XVI, "Association").

[Koh72] T. Kohonen. Correlation associative memories. *IEEE Transactions on Computers,* C-21(4):353–357, April 1972.

[Koh84] T. Kohonen. *Self-Organization and Associative Memory.* Springer-Verlag, Berlin, 1984.

[Kos87] B. Kosko. Adaptive bidirectional associative memories. *Applied Optics,* 26(23):4947–4960, December 1987.

[Kos88] B. Kosko. Bidirectional associative memories. *IEEE Transactions on Systems, Man, and Cybernetics,* SMC-18:42–60, 1988.

[Ste73] G. S. Stent. A physiological mechanism for Hebb's postulate of learning. *Proceedings of the National Academy Sciences,* 70(4):997–1001, April 1973.

[Str76] G. Strang. *Linear Algebra and Its Applications.* Academic Press, New York, 1976.

[TC94] B. A. Telfer and D. P. Casasent. Fast method for updating robust pseudoinverse and ho-kashyap associative processors. *IEEE Transactions on Systems, Man, and Cybernetics,* 24(9):1387–1390, September 1994.

[WCM90] Y. F. Wang, J. B. Cruz, and J. H. Mulligan. Two coding strategies for bidirectional associative memory. *IEEE Transactions on Neural Networks,* 1(1):81–92, March 1990.

PROBLEMS

5.1. Using Equation (5.56), determine the response of the pattern associator to a distorted version of $\underline{s}$. Consider normalized vectors and both orthogonal and nonorthogonal cases.

5.2. Verify that Equation (5.29) reduces to Equation (5.35).

5.3. Verify Equation (5.42).

5.4. Consider the response of the PA developed in Equations (5.39) to (5.41) to the following perturbed inputs:

$$\underline{s}_{1p} = \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} \quad \underline{s}^{2p} = \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} \quad \underline{s}^{3p} = \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix} \tag{5.63}$$

How does the perturbation of each of these vectors (compared with the training set) affect the response (again, with respect to the given $\underline{r}^i$)?

5.5. Consider an extension of a PA design example with the inputs specified in Equation (5.39) as follows:

$$\underline{r}^1 = \begin{pmatrix} 4 \\ 1 \\ 1 \end{pmatrix} \quad \underline{s}^1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \underline{r}^2 = \begin{pmatrix} 4 \\ 1 \\ 0 \end{pmatrix} \quad \underline{s}^2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \tag{5.64}$$

$$\underline{r}^3 = \begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix} \quad \underline{s}^3 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \quad \underline{r}^4 = \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix} \quad \underline{s}^4 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \tag{5.65}$$

(*a*) Formulate S and R.
(*b*) Consider a solution based on forming the pseudoinverse of S.
(*c*) Develop a solution for this case using the Hebbian procedure. Normalize your results, where necessary.

5.6. [This problem explores the type of recall or association provided by a pseudoinverse-based PA. The results of Equations (5.39) to (5.41) may be useful for verification.] Suppose a perturbed input pattern is a linear combination of the n stored patterns (denoted $\underline{s}^i$), i.e.,

$$\underline{s}^p = \sum_{i=1}^{n} k_i \underline{s}^i \tag{5.66}$$

Is the PA response to $\underline{s}^p$ denoted $\underline{r}^p$ formed in the following manner?

$$\underline{r}^p = \sum_{i=1}^{n} k_i \underline{r}^i \tag{5.67}$$

5.7. The design of a D/A converter was addressed in the Chapter 4 problems. As an alternative to the pseudoinverse-based solution, use the Hebbian approach with *bipolar* ($\{-1, 1\}$) inputs and both centered and noncentered output representation ranges.

5.8. How would you incorporate training of the bias inputs into the Hebbian approach?

5.9. In this problem we investigate the rationale behind and influence of *repeated pattern pairs* in H. For example, suppose H contains two or more occurrences of a redundant pattern pair ($\underline{s}^{red}$, $\underline{r}^{red}$).
(*a*) Suppose $\underline{r}^{red}$ is a classification decision for stimulus pattern $\underline{s}^{red}$. If H accurately reflects inputs that the PA is likely to encounter, does the frequency of occurrence of ($\underline{s}^{red}$, $\underline{r}^{red}$) suggest anything concerning the a priori probability $P(w_r)$, where the decision $\underline{r}^{red}$ corresponds to class w_r?
(*b*) What is the effect of this repeated pattern in developing weights using Equation (5.54)? (Relate to Hebbian learning.)
(*c*) What is the effect of this repeated pattern in attempting to use the pseudoinverse training procedure?
(*d*) Discuss the circumstances under which the redundant patterns should be retained versus eliminated from H.

5.10. We desire a PA using linear units. Given the following $n = 2$ training set consisting of $d = 2$ S-R pairs ($\underline{s}^i$, $\underline{r}^i$), where

$$\underline{r}^i = \begin{pmatrix} r_{i1} \\ r_{i2} \end{pmatrix} \tag{5.68}$$

$$\underline{s}^1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad \underline{r}^1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \tag{5.69}$$

$$\underline{s}_2 = \begin{pmatrix} -1 \\ -1 \end{pmatrix} \qquad \underline{r}_2 = \begin{pmatrix} -1 \\ 0 \end{pmatrix} \tag{5.70}$$

(*a*) Considering each of the desired unit outputs (r_{i1} and r_{i2}) separately, is the problem linearly separable?

(*b*) Using a single linear layer of units, use the Hebbian training approach to develop the weight matrix.
(*c*) Assess your results. Does the network exhibit associative recall? How well does it generalize? Into what does it map patterns that are radically different from those in the training set?
(*d*) Consider the possibility of improving the mapping in part (*a*) by adding a bias input to each unit, thereby making the input dimension $d = 3$. Repeat parts (*a*)–(*c*) and assess your results.

5.11. Consider a single layer of linear elements of the form

$$\underline{o} = W\underline{i} \tag{5.71}$$

Develop gradient descent equations to train this network using:
(*a*) A single $(\underline{o}^k, \underline{i}^k)$ pair. Use the error formulation

$$\underline{e}^k = \underline{o}^k - W\underline{i}^k \tag{5.72}$$

and minimize the norm squared of $\underline{e}^k$.
(*b*) All $(\underline{o}^k, \underline{i}^k)$ pairs collectively. Note that here you will first need a way to formulate the error *over the entire training set.*

5.12. An A/D converter implementation was shown in Example 4 of Section 5.3.5. This problem considers the case where the representation is centered about the midpoint of the input interval. The training set for the revised 2-bit A/D converter is shown in Table P5.12. Develop and assess a suitable ANN solution.

TABLE P5.12
Training data for "centered-output representation" A/D converter

Input (i_1)	Output (o_1 o_2)	
0.125	0	0
0.375	0	1
0.625	1	0
0.875	1	1

5.13. Formulate and solve the example problem of Section 5.4.4 using a pseudoinverse formulation, and compare the results with those shown in Section 5.4.4.