Correlation Matrix Memory

3.1 Introduction

In a neurobiological context, memory refers to the relatively enduring neural alterations induced by the interaction of an organism with its environment (Teyler, 1986). Without such a change, there can be no memory. Furthermore, for the memory to be useful, it must be accessible to the nervous system so as to influence future behavior. In the first place, however, an activity pattern must be stored in memory through a learning process. Indeed, memory and learning are intricately connected. When a particular activity pattern is learned, it is stored in the brain, from which it can be recalled later when required. Memory may be divided into short-term and long-term memory, depending on the retention time (Arbib, 1989). Short-term memory refers to a compilation of knowledge representing the “current” state of the environment. Any discrepancies between knowledge stored in short-term memory and a “new” state are used to update the short-term memory. Long-term memory, on the other hand, refers to knowledge stored for a long time or permanently.

In this chapter we study a brainlike distributed memory that operates by association, which is rather simple to understand and yet fundamental in its operation. Indeed, association has been known to be a prominent feature of human memory since Aristotle, and all models of cognition use association in one form or another as the basic operation (Anderson, 1985). For obvious reasons, this kind of memory is called associative memory.

A fundamental property of associative memory is that it maps an output pattern of neural activity onto an input pattern of neural activity. In particular, during the learning phase, a key pattern is presented as stimulus, and the memory transforms it into a memorized or stored pattern. The storage takes place through specific changes in the synaptic weights of the memory. During the retrieval or recall phase, the memory is presented with a stimulus that is a noisy version or an incomplete description of a key pattern originally associated with a stored pattern. Despite imperfections in the stimulus, the associative memory has the capability to recall the stored pattern correctly. Accordingly, associative memories are used in applications such as pattern recognition, to recover data when the available information is imprecise.

From this brief exposition of associative memory, we may identify the following characteristics of this kind of memory:

1. The memory is distributed.
2. Both the stimulus (key) pattern and the response (stored) pattern of an associative memory consist of data vectors.
3. Information is stored in memory by setting up a spatial pattern of neural activities across a large number of neurons.
4. Information contained in a stimulus not only determines its storage location in memory but also an address for its retrieval.

5. Despite the fact that the neurons do not represent reliable and low-noise computing cells, the memory exhibits a high degree of resistance to noise and damage of a diffusive kind.

6. There may be interactions between individual patterns stored in memory. (Otherwise, the memory would have to be exceptionally large for it to accommodate the storage of a large number of patterns in perfect isolation from each other.) There is therefore the distinct possibility of the memory making errors during the recall process.

**Autoassociation Versus Heteroassociation**

We may distinguish two basic types of association: autoassociation and heteroassociation. In an **autoassociative memory**, a key vector is associated with itself in memory—hence the name. Accordingly, the input and output signal (data) spaces have the same dimensionality. In a **heteroassociative memory**, on the other hand, arbitrary key vectors are associated (paired) with other arbitrary memorized vectors. The output space dimensionality may or may not equal the input space dimensionality. In both cases, however, a stored vector may be recalled (retrieved) from the memory by applying a stimulus that consists of a partial description (i.e., fraction) or noisy version of the key vector originally associated with a desired form of the stored vector. For example, the data stored in an autoassociative memory may represent the photograph of a person, but the stimulus may be composed from a noisy reproduction or a masked version of this photograph.

**Linearity Versus Nonlinearity**

An associative memory may also be classified as linear or nonlinear, depending on the model adopted for its neurons. In the **linear** case, the neurons act (to a first approximation) like a linear combiner. To be more specific, let the data vectors $\mathbf{a}$ and $\mathbf{b}$ denote the stimulus (input) and the response (output) of an associative memory, respectively. In a linear associative memory, the input–output relationship is described by

$$\mathbf{b} = \mathbf{M}\mathbf{a}$$

where $\mathbf{M}$ is called the *memory matrix*. The matrix $\mathbf{M}$ specifies the network connectivity of the associative memory. Figure 3.1 depicts a block-diagram representation of a linear associative memory. In a **nonlinear** associative memory, on the other hand, we have an input–output relationship of the form

$$\mathbf{b} = \varphi(\mathbf{M}; \mathbf{a})\mathbf{a}$$

where, in general, $\varphi(\cdot; \cdot)$ is a nonlinear function of the memory matrix and the input vector.

**Organization of the Chapter**

In this chapter we study the characterization of linear associative memory and methods for learning the storage matrix from pairs of associated patterns. The effects of noise

---

**FIGURE 3.1** Block diagram of associative memory.
Correlation Matrix Memory

The recall properties of this type of memory are also considered. Nonlinear associative memories are considered in subsequent chapters.

In Section 3.2 we discuss the idea of distributed memory mapping, which provides the basis for the study of linear associative memories. Then, in Section 3.3 we describe the outer product rule, which is a generalization of Hebb's postulate of learning, for the design of a special form of linear associative memory known as the correlation matrix memory. The recall properties of this type of memory are also discussed. Section 3.4 describes how an error-correction mechanism may be incorporated into the design of a correlation matrix memory, forcing it to associate perfectly. The chapter concludes with Section 3.5, presenting a comparison between the correlation matrix memory and another linear associative memory known as the pseudoinverse memory; the basic theory of this latter type of memory is presented in Appendix A at the end of the book.

3.2 Distributed Memory Mapping

In a distributed memory, the basic issue of interest is the simultaneous or near-simultaneous activities of many different neurons, which are the result of external or internal stimuli. The neural activities form a large spatial pattern inside the memory that contains information about the stimuli. The memory is therefore said to perform a distributed mapping that transforms an activity pattern in the input space into another activity pattern in the output space. We may illustrate some important properties of a distributed memory mapping by considering an idealized neural network that consists of two layers of neurons. Figure 3.2a illustrates the case of a network that may be regarded as a model component of a nervous system (Scofield and Cooper, 1985; Cooper, 1973). Each neuron in the input layer is connected to every one of the neurons in the output layer. The actual synaptic connections between the neurons are very complex and redundant. In the model of Fig. 3.2a, a single ideal junction is used to represent the integrated effect of all the synaptic contacts between the dendrites of a neuron in the input layer and the axon branches of a neuron in the output layer. In any event, the level of activity of a neuron in the input layer may affect the level of activity of every other neuron in the output layer.

The corresponding situation for an artificial neural network is depicted in Fig. 3.2b. Here we have an input layer of source nodes and an output layer of neurons acting as computation nodes. In this case, the synaptic weights of the network are included as integral parts of the neurons in the output layer. The connecting links between the two layers of the network are simply wires.

In the mathematical analysis to follow in this chapter, the neural networks in Figs. 3.2a and 3.2b are both assumed to be linear. The implication of this assumption is that each neuron acts as a linear combiner.

To proceed with the analysis, suppose that an activity pattern \( a_k \) occurs in the input layer of the network and that an activity pattern \( b_k \) occurs simultaneously in the output layer. The issue we wish to consider here is that of learning from the association between the patterns \( a_k \) and \( b_k \).

The patterns \( a_k \) and \( b_k \) are represented by vectors, written in their expanded forms as follows:

\[
 a_k = [a_{k1}, a_{k2}, \ldots, a_{kp}]^T \tag{3.3}
\]

and

\[
 b_k = [b_{k1}, b_{k2}, \ldots, b_{kp}]^T \tag{3.4}
\]

where the superscript \( T \) denotes transposition. For convenience of presentation, we have assumed that the input space dimensionality (i.e., the dimension of vector \( a_k \)) and the
output space dimensionality (i.e., the dimension of vector $b_k$) are the same, equal to $p$. From here on, we will refer to $p$ as network dimensionality or simply dimensionality. Note that $p$ equals the number of source nodes in the input layer or neurons in the output layer. For a neural network with a large number of neurons, which is typically the case, the dimensionality $p$ can therefore be large.

The elements of both $a_k$ and $b_k$ can assume positive and negative values. This is a valid proposition in an artificial neural network. It may also occur in a nervous system by considering the relevant physiological variable to be the difference between an actual activity level (e.g., firing rate of a neuron) and a nonzero spontaneous activity level.

For a specified dimensionality $p$, the neural network of Fig. 3.2a or 3.2b can associate a number of patterns—say, $q$. In general, the network can support a number of different associations up to a potential capacity of $p$. In reality, however, the capacity of the network to store different patterns (i.e., $q$) is less than the dimensionality $p$.

In any event, we may describe the different associations performed by the network by writing

$$a_k \rightarrow b_k, \quad k = 1, 2, \ldots, q$$

(3.5)

The activity pattern $a_k$ acts as a stimulus that not only determines the storage location of information in the stimulus $a_k$, but also holds the key for its retrieval. Accordingly, $a_k$ is referred to as a key pattern, and $b_k$ is referred to as a memorized pattern.

With the networks of Fig. 3.2 assumed to be linear, the association of a key vector $a_k$ with a memorized vector $b_k$ described in symbolic form in Eq. (3.5) may be recast in
matrix form as follows:

\[ b_k = W(k)a_k, \quad k = 1, 2, \ldots, q \]  \hspace{1cm} (3.6)

where \( W(k) \) is a weight matrix determined solely by the input–output pair \((a_k, b_k)\).

To develop a detailed description of the weight matrix \( W(k) \), consider Fig. 3.3, which shows a detailed arrangement of neuron \( i \) in the output layer. The output \( b_{ki} \) of neuron \( i \) due to the combined action of the elements of the key pattern \( a_k \) applied as stimulus to the input layer is given by

\[ b_{ki} = \sum_{j=1}^{p} w_{ij}(k)a_{kj}, \quad i = 1, 2, \ldots, q \]  \hspace{1cm} (3.7)

where the \( w_{ij}(k), j = 1, 2, \ldots, p, \) are the synaptic weights of neuron \( i \) corresponding to the \( k \)th pair of associated patterns. Using matrix notation, we may express \( b_{ki} \) in the equivalent form

\[ b_{ki} = [w_{i1}(k), w_{i2}(k), \ldots, w_{ip}(k)]\begin{bmatrix} a_{k1} \\ a_{k2} \\ \vdots \\ a_{kp} \end{bmatrix}, \quad i = 1, 2, \ldots, q \]  \hspace{1cm} (3.8)

The column vector on the right-hand side of Eq. (3.8) is recognized as the key vector \( a_k \). Hence, substituting Eq. (3.8) in the definition of the stored vector \( b_k \) given in Eq. (3.4), we get

\[
\begin{bmatrix}
  b_{k1} \\
  b_{k2} \\
  \vdots \\
  b_{kp}
\end{bmatrix} =
\begin{bmatrix}
  w_{11}(k) & w_{12}(k) & \cdots & w_{1p}(k) \\
  w_{21}(k) & w_{22}(k) & \cdots & w_{2p}(k) \\
  \vdots & \vdots & \ddots & \vdots \\
  w_{p1}(k) & w_{p2}(k) & \cdots & w_{pp}(k)
\end{bmatrix}
\begin{bmatrix}
  a_{k1} \\
  a_{k2} \\
  \vdots \\
  a_{kp}
\end{bmatrix}
\]  \hspace{1cm} (3.9)

Equation (3.9) is the expanded form of the matrix transformation or mapping described in Eq. (3.6). In particular, the \( p \)-by-\( p \) weight matrix \( W(k) \) is defined by

\[
W(k) =
\begin{bmatrix}
  w_{11}(k) & w_{12}(k) & \cdots & w_{1p}(k) \\
  w_{21}(k) & w_{22}(k) & \cdots & w_{2p}(k) \\
  \vdots & \vdots & \ddots & \vdots \\
  w_{p1}(k) & w_{p2}(k) & \cdots & w_{pp}(k)
\end{bmatrix}
\]  \hspace{1cm} (3.10)

The individual presentations of the \( q \) pairs of associated patterns described in Eq. (3.5) produce corresponding values of the individual matrix, namely, \( W(1), W(2), \ldots, W(q) \). Recognizing that the pattern association \( a_k \rightarrow b_k \) is represented by the weight matrix \( W(k) \), we may define \( p \)-by-\( p \) memory matrix that describes the summation of the weight matrices for the entire set of pattern associations as follows:

\[
M = \sum_{k=1}^{q} W(k)
\]  \hspace{1cm} (3.11)
The memory matrix $M$ defines the overall connectivity between the input and output layers of the associative memory. In effect, it represents the total experience gained by the memory as a result of the presentations of $q$ input–output patterns. Stated in another way, the memory matrix $M$ contains a piece of every input–output pair of activity patterns presented to the memory.

The definition of the memory matrix given in Eq. (3.11) may be restructured in the form of a recursion, as shown by

$$M_k = M_{k-1} + W(k), \quad k = 1, 2, \ldots, q$$

(3.12)

where it is noted that the initial value $M_0$ is zero (i.e., the synaptic weights in the memory are all initially zero), and the final value $M_q$ is identically equal to $M$ as defined in Eq. (3.11). According to the recursive formula of Eq. (3.12), the term $M_{k-1}$ is the old value of the memory matrix resulting from $(k - 1)$ pattern associations, and $M_k$ is the updated value in light of the increment $W(k)$ produced by the $k$th association. Note, however, that when $W(k)$ is added to $M_{k-1}$, the increment $W(k)$ loses its distinct identity among the mixture of contributions that form $M_k$. But, in spite of the synaptic mixing of different associations, information about the stimuli may not have been lost, as demonstrated in the next section. Note also that as the number $q$ of stored patterns increases, the influence of a new pattern on the memory as a whole is progressively reduced.

### 3.3 Correlation Matrix Memory

Suppose that the associative memory of Fig. 3.2b has learned the memory matrix $M$ through the associations of key and memorized patterns described by $a_k \rightarrow b_k$, where $k = 1, 2, \ldots, q$. We may postulate $\tilde{M}$, denoting an estimate of the memory matrix $M$ in terms of these patterns as follows (Anderson, 1972, 1983; Cooper, 1973):

$$\tilde{M} = \sum_{k=1}^{q} b_k a_k^T$$

(3.13)

The term $b_k a_k^T$ represents the outer product of the key pattern $a_k$ and the memorized pattern $b_k$. This outer product is an "estimate" of the weight matrix $W(k)$ that maps the output pattern $b_k$ onto the input pattern $a_k$. Since the patterns $a_k$ and $b_k$ are both $p$-by-1 vectors by assumption, it follows that their outer product $b_k a_k^T$ and therefore the estimate $\tilde{M}$ is a $p$-by-$p$ matrix. This dimensionality is in perfect agreement with that of the memory matrix $M$ defined in Eq. (3.11). Note also that the format of the summation of the estimate $\tilde{M}$ bears a direct relation to that of the memory matrix defined in Eq. (3.11).

A typical term of the outer product $b_k a_k^T$ is written as $b_k a_k^T$, where $a_{kj}$ is the output of source node $j$ in the input layer, and $b_k$ is the output of neuron $i$ in the output layer. In the context of synaptic weight $w_{ij}(k)$ for the $k$th association, source node $j$ is recognized...
as a presynaptic node and neuron i in the output layer is recognized as a postsynaptic node. Hence, the “local” learning process described in Eq. (3.13) may be viewed as a generalization of Hebb’s postulate of learning. It is also referred to as the outer product rule, in recognition of the matrix operation used to construct the memory matrix \( \hat{M} \). Correspondingly, an associative memory so designed is called a correlation matrix memory.

Correlation, in one form or another, is indeed the basis of learning, association, pattern recognition, and memory recall (Eggermont, 1990).

Equation (3.13) may be reformulated in the equivalent form

\[
\hat{M} = [b_1, b_2, \ldots, b_q] \begin{bmatrix} a_1^T \\ a_2^T \\ \vdots \\ a_i^T \end{bmatrix}
\]

\[
= BA^T
\]

where

\[
A = [a_1, a_2, \ldots, a_q]
\]

and

\[
B = [b_1, b_2, \ldots, b_q]
\]

The matrix \( A \) is a p-by-q matrix composed of the entire set of key patterns used in the learning process; it is called the key matrix. The matrix \( B \) is a p-by-q matrix composed of the corresponding set of memorized patterns; it is called the memorized matrix.

Equation (3.13) may also be restructured in the form of a recursion as follows:

\[
\hat{M}_k = \hat{M}_{k-1} + b_k a_i^T, \quad k = 1, 2, \ldots, q
\]

A signal-flow graph representation of this recursion is depicted in Fig. 3.4. According to this signal-flow graph and the recursive formula of Eq. (3.17), the matrix \( \hat{M}_{k-1} \) represents an old estimate of the memory matrix; and \( \hat{M}_k \) represents its updated value in the light of a new association performed by the memory on the patterns \( a_i \) and \( b_k \). Comparing the recursion of Eq. (3.17) with that of Eq. (3.12), we see that the outer product \( b_k a_i^T \) represents an estimate of the weight matrix \( \hat{W}(k) \) corresponding to the kth association of key and memorized patterns, \( a_i \) and \( b_k \). Moreover, the recursion of Eq. (3.17) has an initial value \( \hat{M}_0 \) equal to zero, and it yields a final value \( \hat{M}_q \) identically equal to \( \hat{M} \) as defined in Eq. (3.13).

**FIGURE 3.4** Signal-flow graph representation of Eq. (3.17).
Recall

The fundamental problem posed by the use of an associative memory is the address and recall of patterns stored in memory. To explain one aspect of this problem, let $\mathbf{M}$ denote the memory matrix of an associative memory, which has been completely learned through its exposure to $q$ pattern associations in accordance with Eq. (3.13). Let a key pattern $\mathbf{a}_j$ be picked at random and reapplied as stimulus to the memory, yielding the response

$$\mathbf{b} = \mathbf{Ma}_j$$

(3.18)

Substituting Eq. (3.13) in (3.18), we get

$$\mathbf{b} = \sum_{k=1}^{q} \mathbf{b}_k \mathbf{a}_j^T \mathbf{a}_j$$

$$= \sum_{k=1}^{q} (\mathbf{a}_j^T \mathbf{a}_j) \mathbf{b}_k$$

(3.19)

where, in the second line, it is recognized that $\mathbf{a}_j^T \mathbf{a}_j$ is a scalar equal to the *inner product* of the key vectors $\mathbf{a}_k$ and $\mathbf{a}_j$. Moreover, we may rewrite Eq. (3.19) as

$$\mathbf{b} = (\mathbf{a}_j^T \mathbf{a}_j) \mathbf{b}_j + \sum_{k=1}^{q} (\mathbf{a}_j^T \mathbf{a}_j) \mathbf{b}_k$$

(3.20)

Let each of the key patterns $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_q$ be normalized to have unit energy; that is,

$$E_k = \sum_{l=1}^{q} a_{jl}^2$$

$$= \mathbf{a}_j^T \mathbf{a}_k$$

$$= 1, \quad k = 1, 2, \ldots, q$$

(3.21)

Accordingly, we may simplify the response of the memory to the stimulus (key pattern) $\mathbf{a}_j$ as

$$\mathbf{b} = \mathbf{b}_j + \mathbf{v}_j$$

(3.22)

where

$$\mathbf{v}_j = \sum_{k=1}^{q} (\mathbf{a}_j^T \mathbf{a}_j) \mathbf{b}_k$$

(3.23)

The first term on the right-hand side of Eq. (3.22) represents the “desired” response $\mathbf{b}_j$; it may therefore be viewed as the “signal” component of the actual response $\mathbf{b}$. The second term $\mathbf{v}_j$ is a “noise vector” that arises because of the *crosstalk* between the key vector $\mathbf{a}_j$ and all the other key vectors stored in memory. Indeed, if the individual patterns are statistically independent, then from the central limit theorem of probability theory, we may conclude that the noise vector $\mathbf{v}_j$ is Gaussian-distributed. We thus see that Eq. (3.22) represents a classic *signal in Gaussian noise detection problem* (Van Trees, 1968).

In a general setting, these noise considerations may severely limit the number of patterns that can be reliably stored in an associative memory.

In the context of a linear signal space, we may define the *cosine of the angle* between a pair of vectors $\mathbf{a}_k$ and $\mathbf{a}_j$ as the inner product of $\mathbf{a}_k$ and $\mathbf{a}_j$ divided by the product of their individual Euclidean *norms* or *lengths*, as shown by

$$\cos(\mathbf{a}_k, \mathbf{a}_j) = \frac{\mathbf{a}_j^T \mathbf{a}_k}{\|\mathbf{a}_k\| \|\mathbf{a}_j\|}$$

(3.24)
The symbol $\|a_k\|$ signifies the norm of vector $a_k$, defined as the square root of the energy of $a_k$, as shown by

$$
\|a_k\| = (a_k'^*a_k)^{1/2} = E_k^{1/2}
$$

(3.25)

Returning to the situation at hand, we note that the key vectors are normalized to have unit energy in accordance with Eq. (3.21). We may therefore simplify the definition of Eq. (3.24) as

$$
\cos(a_k, a_j) = a_k'^*a_j
$$

(3.26)

Correspondingly, we may redefine the noise vector of Eq. (3.23) as

$$
v_j = \sum_{k \neq j}^N \cos(a_k, a_j)b_k
$$

(3.27)

We now see that if the key vectors are orthogonal (i.e., perpendicular to each other in a Euclidean sense), then

$$
\cos(a_k, a_j) = 0, \quad k \neq j
$$

(3.28)

and therefore the noise vector $v_j$ is identically zero. In such a case, the response $b$ equals $b_j$. Accordingly, we may state that the memory associates perfectly if the key vectors form an orthonormal set; that is, they satisfy the following pair of conditions:

$$
a_k'^*a_j = \begin{cases} 
1, & k = j \\
0, & k \neq j 
\end{cases}
$$

(3.29)

Suppose now that the key vectors do form an orthonormal set, as prescribed in Eq. (3.29). What is then the limit on the storage capacity of the associative memory? Stated another way, what is the largest number of patterns that can be reliably stored? The answer to this fundamental question lies in the rank of the memory matrix $M$. The rank of a matrix is defined as the number of independent columns (rows) of the matrix. The memory matrix $M$ is a $p$-by-$p$ matrix, where $p$ is the dimensionality of the input space. Hence the rank of the memory matrix $M$ is limited by the dimensionality $p$. We may thus formally state that the number of patterns that can be reliably stored can never exceed the input space dimensionality.

**Practical Considerations**

Given a set of key vectors that are linearly independent but nonorthonormal, we may use a preprocessor to transform them into an orthonormal set; the preprocessor is designed to perform a Gram–Schmidt orthogonalization on the key vectors prior to association. This form of transformation is linear, maintaining a one-to-one correspondence between the input (key) vectors $a_1, a_2, \ldots, a_r$ and the resulting orthonormal vectors $c_1, c_2, \ldots, c_r$, as indicated here:

$$
\{a_1, a_2, \ldots, a_r\} \leftrightarrow \{c_1, c_2, \ldots, c_r\}
$$

where $c_1 = a_1$, and the remaining $c_k$ are defined by (Strang, 1980)

$$
c_k = a_k - \sum_{i=1}^{k-1} \left( \frac{c_i'^*a_k}{c_i'^*c_i} \right) c_i, \quad k = 2, 3, \ldots, q
$$

(3.30)

If $r$ is the rank of a rectangular matrix of dimensions $m$ by $n$, we then obviously have $r \leq \min(m, n)$. 

1 If $r$ is the rank of a rectangular matrix of dimensions $m$ by $n$, we then obviously have $r \leq \min(m, n)$. 


The associations are then performed on the pairs \((c_k, b_k)\), \(k = 1, 2, \ldots, q\). The block diagram of Fig. 3.5 highlights the order in which the preprocessing and association are performed.

The orthogonality of key vectors may also be approximated using statistical considerations. Specifically, if the input space dimensionality \(p\) is large and the key vectors have statistically independent elements, then they will be close to orthogonality with respect to each other.²

### Logic

In a real-life situation, we often find that the key patterns presented to an associative memory are not orthogonal nor are they highly separated from each other. Consequently, a correlation matrix memory characterized by the memory matrix of Eq. (3.13) may sometimes get confused and make errors. That is, the memory occasionally recognizes and associates patterns never seen or associated before.

To illustrate this property of an associative memory, consider a set of key patterns,

\[
\{a_{\text{key}}\}: a_1, a_2, \ldots, a_q
\]  

(3.31)

and a corresponding set of memorized patterns,

\[
\{b_{\text{mem}}\}: b_1, b_2, \ldots, b_q
\]  

(3.32)

To express the closeness of the key patterns in a linear signal space, we introduce the concept of community. In particular, we define the community of the set of patterns \(\{a_{\text{key}}\}\) as the lower bound of the inner products \(a_i^T a_k\) of any two patterns \(a_i\) and \(a_k\) in the set. Let \(\mathbf{M}\) denote the memory matrix resulting from the training of the associative memory on a set of key patterns represented by \(\{a_{\text{key}}\}\) and a corresponding set of memorized patterns \(\{b_{\text{mem}}\}\) in accordance with Eq. (3.13). The response of the memory, \(\mathbf{b}\), to a stimulus \(\mathbf{a}\), selected from the set \(\{a_{\text{key}}\}\), is given by Eq. (3.20), where it is assumed that each pattern in the set \(\{a_{\text{key}}\}\) is a unit vector (i.e., a vector with unit energy). Let it be further assumed that

\[
a_i^T a_j \geq \gamma \quad \text{for } k \neq j
\]  

(3.33)

If the lower bound \(\gamma\) is large enough, the memory may fail to distinguish the response \(\mathbf{b}\) from that of any other key pattern contained in the set \(\{a_{\text{key}}\}\). Indeed, if the key patterns in this set have the form

\[
a_j = a_0 + \mathbf{v}
\]  

(3.34)

where \(\mathbf{v}\) is a stochastic vector, then there is a real likelihood that the memory will recognize \(a_0\) and associate with it a vector \(b_0\) rather than any of the actual pattern pairs used to train it in the first place; \(a_0\) and \(b_0\) denote a pair of patterns never seen before. This phenomenon may be termed animal logic, which is not logic at all (Cooper, 1973).

² We say two random variables \(x\) and \(y\) are statistically independent if their joint probability density function satisfies the condition \(f(x, y) = f(x)f(y)\), where \(f(x)\) and \(f(y)\) are the probability density functions of \(x\) and \(y\), respectively.
The correlation matrix memory characterized by the memory matrix of Eq. (3.13) is simple to design. However, a major limitation of such a design is that the memory may commit too many errors, and the memory has no mechanism to correct for them. Specifically, given the memory matrix $\mathbf{M}$ that has been learned from the associations $a_k \rightarrow b_k$ for $k = 1, 2, \ldots, q$ in accordance with Eq. (3.13), the actual response $b$ produced when the key pattern $a$ is presented to the memory may not be close enough (in a Euclidean sense) to the desired response $b$ for the memory to associate perfectly. This shortcoming of the correlation matrix memory is inherited from the use of Hebb’s postulate of learning that has no provision for feedback from the output to the input. As a remedy for it, we may incorporate an error-correction mechanism into the design of the memory, forcing it to associate perfectly (Anderson and Murphy, 1986; Anderson, 1983).

Suppose that we wish to construct a memory matrix $\mathbf{M}$ that describes the synaptic weights of a neural network with two layers, as shown in Fig. 3.2a or 3.2b. We have two fundamental objectives in mind. First, the memory matrix $\mathbf{M}$ learns the information represented by the associations of Eq. (3.5), reproduced here for convenience:

$$a_k \rightarrow b_k, \quad k = 1, 2, \ldots, q \tag{3.35}$$

Second, the memory accurately reconstructs each one of these associations.

To proceed then with the development of the error-correction procedure, let $\mathbf{M}(n)$ denote the memory matrix learned at iteration $n$. A key vector $a_k$, selected at random, is applied to the memory at this time, yielding the “actual” response $\mathbf{M}(n)a_k$. Accordingly, we may define the error vector:

$$e_k(n) = b_k - \mathbf{M}(n)a_k \tag{3.36}$$

where $b_k$ is the activity pattern to be associated with $a_k$. We may view $b_k$ as the “desired” response. The error vector $e_k(n)$ is, in turn, used to compute an adjustment to the memory matrix at time $n$, which is constructed in accordance with the rule:

$$\text{(Adjustment)} = \begin{pmatrix} \text{learning-rate parameter} \\ \text{(error)} \cdot \text{(input)} \end{pmatrix} \tag{3.37}$$

For the problem at hand, we may therefore write

$$\Delta \mathbf{M}(n) = \eta e_k(n)a_k^\top$$

$$= \eta[b_k - \mathbf{M}(n)a_k]a_k^\top \tag{3.38}$$

where $\eta$ is a learning-rate parameter, the difference $b_k - \mathbf{M}(n)a_k$ is the error, and $a_k^\top$ is the input. The adjustment $\Delta \mathbf{M}(n)$ is used to increment the old value of the memory matrix, resulting in the updated value

$$\mathbf{M}(n + 1) = \mathbf{M}(n) + \Delta \mathbf{M}(n)$$

$$= \mathbf{M}(n) + \eta[b_k - \mathbf{M}(n)a_k]a_k^\top \tag{3.39}$$

Ordinarily, a positive constant value is assigned to the learning-rate parameter $\eta$. This has the effect of building a short-term memory into the operation of the algorithm, because recent changes are recalled more accurately than changes in the remote past. Sometimes, however, the learning-rate parameter is tapered with respect to time, so as to approach zero when many associations are learned by the memory (Anderson and Murphy, 1986). To initialize the algorithm described in Eq. (3.39), we may set $\mathbf{M}(0)$ equal to zero.
Figure 3.6 shows a signal-flow graph representation of the error-correction algorithm described in Eq. (3.39). The outer-feedback loop of this graph emphasizes the error-correction capability of the algorithm. However, the presence of this feedback loop also means that care has to be exercised in the selection of the learning-rate parameter $\eta$, so as to ensure stability of the feedback system described in Fig. 3.6, that is, convergence of the error-correction algorithm.

The iterative adjustment to the memory matrix described in Eq. (3.39) is continued until the error vector $e_k(n)$ is negligibly small; that is, the actual response $\hat{M}(n)a_k$ approaches the desired response $b_k$.

The supervised learning procedure based on the use of error correction, as described here, is repeated for each of the $q$ associations in Eq. (3.35), with each association being picked at random. This procedure is called the least-mean-square (LMS) rule or delta rule (Widrow and Hoff, 1960). The latter terminology is in recognition of the fact that the algorithm described in Eq. (3.39) learns from the difference between the desired and actual responses. We shall have more to say on the LMS algorithm in Chapter 5.

**Autoassociation**

In an autoassociative memory, each key pattern is associated with itself in memory; that is,

$$b_k = a_k, \quad k = 1, 2, \ldots, q$$ \hspace{1cm} (3.40)

In such a case, Eq. (3.39) takes on the form

$$\hat{M}(n + 1) = \hat{M}(n) + \eta[a_k - \hat{M}(n)a_k]a_k^T$$ \hspace{1cm} (3.41)

Ideally, as the number of iterations, $n$, approaches infinity, the error vector $e_k(n)$ approaches zero, and the memory characterized by the memory matrix $\hat{M}$ autoassociates perfectly, as shown by

$$\hat{M}(\infty)a_k = a_k \quad \text{for} \ k = 1, 2, \ldots, q$$ \hspace{1cm} (3.42)
Digressing for a moment, the *eigenvalue problem* involving a matrix $A$ is described by the relation (Strang, 1980)

$$Ax = \lambda x$$  \hspace{1cm} (3.43)

where $\lambda$ is called an *eigenvalue* of matrix $A$, and $x$ is called the associated *eigenvector*. Transposing the term $\lambda x$ to the left-hand side, we have

$$[A - \lambda I]x = 0$$  \hspace{1cm} (3.44)

where $I$ is the identity matrix. This equation can be satisfied if and only if

$$\text{det}[A - \lambda I] = 0$$  \hspace{1cm} (3.45)

where $\text{det}(\cdot)$ denotes the determinant of the matrix enclosed within. Equation (3.45) is called the *characteristic equation*, whose roots define the eigenvalues of the matrix $A$.

Returning to the autoassociation problem described in Eq. (3.42), we now see that it describes an eigenvalue problem involving the square matrix $\hat{M}(\infty)$ of dimension $p$, with two properties:

1. Each of the key vectors is an eigenvector of the memory matrix $\hat{M}(\infty)$, defined by

$$\hat{M}(\infty) = \sum_{k=1}^{q} \lambda_k a_k a_k^T$$  \hspace{1cm} (3.46)

2. The eigenvalues of the matrix $\hat{M}(\infty)$ are all equal to unity; that is, the characteristic equation

$$\text{det}[\hat{M}(\infty) - \lambda I] = 0$$  \hspace{1cm} (3.47)

has a root $\lambda = 1$ that is of multiplicity $q$.

Since the eigenvectors of a square matrix form an orthogonal set, it follows from the discussion presented in Section 3.3 that a network having the memory matrix $\hat{M}(\infty)$ autoassociates perfectly, which reconfirms the statement we made previously on perfect autoassociation.

In practice, however, the condition described in Eq. (3.42) is never satisfied exactly. Rather, as the number of iterations approaches infinity, the error vector for each association becomes small but remains finite. Indeed, the error vector $e(n)$ is a stochastic vector, whose final value fluctuates around a mean value of zero. Correspondingly, the actual activity pattern stored in memory (i.e., actual response), defined by

$$\hat{a}_k(n) = \hat{M}(n)a_k, \quad k = 1, 2, \ldots, q$$  \hspace{1cm} (3.48)

is also a stochastic vector whose final value fluctuates around a mean value equal to an eigenvector of $\hat{M}(\infty)$.

### 3.5 Discussion

The correlation matrix memory represents one form of a linear associative memory. There is another type of linear associative memory known as the pseudoinverse or generalized-inverse memory. Unlike the correlation matrix memory, the pseudoinverse memory has no neurobiological basis. Nevertheless, it provides an alternative method for the design of a linear associative memory.

The memory constructions of correlation matrix memory and pseudoinverse memory are different. Given a key matrix $A$ and a memorized matrix $B$, the estimate of the memory matrix constructed by the *correlation matrix memory* is given by Eq. (3.14), reproduced
here for convenience of presentation,

\[ \hat{M}_{\text{cnn}} = BA^T \]  

(3.49)

where the superscript \( T \) denotes transposition. On the other hand, the estimate of the memory matrix constructed by the *pseudoinverse memory* is given by

\[ \hat{M}_{\text{cnn}} = BA^+ \]  

(3.50)

where \( A^+ \) is the *pseudoinverse matrix* of the key matrix \( A \); see Appendix A at the end of the book for a description of the pseudoinverse memory and the derivation of Eq. (3.50).

Cherkassky et al. (1991) present a detailed comparative analysis of the correlation matrix memory and the pseudoinverse memory in the presence of additive noise. The training set vectors are assumed to be error free, and these vectors are used for memory construction. In the recall phase, it is assumed that the unknown stimulus vector \( a \) is described by

\[ a = a_j + \nu \]  

(3.51)

where \( a_j \) is one of the known key vectors, and \( \nu \) is an additive input noise vector. Components of the noise vector \( \nu \) are assumed to be *independent and identically distributed* (i.i.d.) random variables with mean \( \mu_i \) and variance \( \sigma_i^2 \). The resulting response of the associative memory with memory matrix \( \hat{M} \) is

\[ b = \hat{M}a = \hat{M}(a_j + \nu) \]  

(3.52)

In light of Eq. (3.22), we may express the matrix product \( \hat{M}a_j \) as the vector \( b_j \) plus a noise term that accounts for crosstalk between the key vector \( a_j \) and all the other key vectors stored in memory. Let this crosstalk term be denoted by \( \nu_j \). Accordingly, we may rewrite Eq. (3.52) as follows:

\[ b = b_j + \nu_j + \hat{M}\nu \]  

(3.53)

For performance measure, the output signal-to-noise ratio, \((\text{SNR})_o\), is divided by the input signal-to-noise ratio, \((\text{SNR})_i\), as shown by

\[ \frac{(\text{SNR})_o}{(\text{SNR})_i} = \frac{s_j^2/(\sigma_j^2 + \sigma_c^2)}{s_j^2/\sigma_i^2} \]

\[ = \frac{s_j^2\sigma_i^2}{s_j^2(\sigma_j^2 + \sigma_c^2)} \]  

(3.54)

where \( \sigma_j^2 \) is the input noise variance due to \( \nu \), \( \sigma_c^2 \) is the output noise variance due to \( \hat{M}\nu \), \( \sigma_c^2 \) is the variance of the crosstalk term \( \nu_j \), \( s_j^2 \) is the variance of elements in the response vectors, and \( s_j^2 \) is the variance of all elements in the key vectors. Equation (3.54) is applicable to autoassociative as well as heteroassociative memory.

The results obtained by Cherkassky et al. (1991) may be summarized as follows:

1. In the case of an autoassociative memory, the pseudoinverse memory provides better noise performance than the correlation matrix memory; this result is in agreement with earlier findings (Kohonen, 1988b).
2. In the case of a many-to-one classifier, in which a number of key vectors are mapped into one class and which includes the unary classifier as a special case, the correlation matrix memory provides a better noise performance than the pseudoinverse memory.
These results are based on certain assumptions about the key vectors. First, it is assumed that the key vectors are of approximately the same Euclidean length; this can be achieved by simple preprocessing (normalization) of the input data. Second, it is assumed that the key vectors have statistically independent components.

**PROBLEMS**

3.1 The theory of the correlation matrix memory developed in Section 3.3 assumed that the input space dimensionality and output space dimensionality are the same. How is this theory modified by assuming different values, $L$ and $M$, say, for these two dimensionalities?

3.2 Consider the following orthonormal sets of key patterns, applied to a correlation matrix memory:

\[
\begin{align*}
a_1 &= [1, 0, 0, 0]^T \\
a_2 &= [0, 1, 0, 0]^T \\
a_3 &= [0, 0, 1, 0]^T
\end{align*}
\]

The respective stored patterns are

\[
\begin{align*}
b_1 &= [5, 1, 0]^T \\
b_2 &= [-2, 1, 6]^T \\
b_3 &= [-2, 4, 3]^T
\end{align*}
\]

(a) Calculate the memory matrix $M$.

(b) Show that the memory associates perfectly.

3.3 Consider again the correlation matrix memory of Problem 3.2. The stimulus applied to the memory is a noisy version of the key pattern $a_1$, as shown by

\[
a = [0.8, -0.15, 0.15, -0.20]^T
\]

(a) Calculate the memory response $b$.

(b) Show that the response $b$ is closest to the stored pattern $b_1$ in a Euclidean sense.

3.4 An autoassociative memory is trained on the following key vectors:

\[
\begin{align*}
a_1 &= [2, -2, -3, \sqrt{3}]^T \\
a_2 &= [2, -2, -\sqrt{8}]^T \\
a_3 &= [3, -1, \sqrt{6}]^T
\end{align*}
\]

(a) Calculate the angles between these vectors. How close are they to orthogonality with respect to each other?

(b) Using the generalization of Hebb’s rule (i.e., the outer product rule), calculate the memory matrix of the network. Hence, investigate how close to perfect the memory autoassociates.

(c) A masked version of the key vector $a_1$, namely,

\[
a = [0, -3, \sqrt{3}]^T
\]

is applied to the memory. Calculate the response of the memory, and compare your result with the desired response $a_1$. 

3.5 The autoassociative memory of Problem 3.4 is modified to include an error-correction mechanism. The learning-rate parameter $\eta$ equals 0.1.

(a) Calculate the modified version of the memory matrix of the memory. Using this new result, investigate how close to perfect the memory autoassociates, and compare your finding with that of Problem 3.4, part (b).

(b) Repeat part (c) of Problem 3.4.

3.6 Consider an orthonormal set of vectors $a_1, a_2, \ldots, a_n$, the dimension of each of which is $p$. Using the outer product rule, determine the memory matrix $\hat{M}$ of an autoassociative memory based on this set of vectors.

(a) How are the eigenvectors of the memory matrix $\hat{M}$ related to the vectors $a_1, a_2, \ldots, a_n$? Assuming that $q < p$, what are the eigenvalues of $M$?

(b) What is the maximum value of vectors that can be stored in the autoassociative memory for reliable recall to be possible?