Associative Memory

Soft Computing

Associative Memory (AM), topics : Description, content address-ability, working, classes of AM : auto and hetero, AM related terms - encoding or memorization, retrieval or recollection, errors and noise, performance measure - memory capacity and content-addressability. Associative memory models : network architectures - linear associator, Hopfield model and bi-directional model (BAM). Auto-associative memory (auto-correlators) : how to store patterns ? how to retrieve patterns? recognition of noisy patterns. Bi-directional hetero-associative memory (hetero-correlators) : BAM operations - retrieve the nearest pair, addition and deletion of pattern pairs, energy function for BAM - working of Kosko's BAM, incorrect recall of pattern, multiple training encoding strategy – augmentation matrix, generalized correlation matrix and algorithm.

Associative Memory

Soft Computing

Topics

(Lectures 21, 22, 23, 24 4 hours)

Associative Memory

What is Associative Memory ?

- An associative memory is a content-addressable structure that maps a set of input patterns to a set of output patterns.

- A content-addressable structure is a type of memory that allows the recall of data based on the degree of similarity between the input pattern and the patterns stored in memory.

- There are two types of associative memory : auto-associative and hetero-associative.

- An auto-associative memory retrieves a previously stored pattern that most closely resembles the current pattern.

- In a hetero-associative memory, the retrieved pattern is in general, different from the input pattern not only in content but possibly also in type and format.

- Neural networks are used to implement these associative memory models called NAM (Neural associative memory).

1. Associative Memory

An associative memory is a content-addressable structure that maps a set of input patterns to a set of output patterns. A content-addressable structure refers to a memory organization where the memory is accessed by its content as opposed to an explicit address in the traditional computer memory system. The associative memory are of two types : auto-associative and hetero-associative.

- An **auto-associative memory** retrieves a previously stored pattern that most closely resembles the current pattern.

- In **hetero-associative memory**, the retrieved pattern is in general different from the input pattern not only in content but possibly also in type and format.

1.1 Description of Associative Memory

An associative memory is a content-addressable structure that allows, the recall of data, based on the degree of similarity between the input pattern and the patterns stored in memory.

- **Example : Associative Memory**

The figure below shows a memory containing names of several people.

If the given memory is content-addressable,

Then using the erroneous string "Crhistpher Columbos" as key is sufficient to retrieve the correct name "Christopher Colombus."

In this sense, this type of memory is robust and fault-tolerant, because this type of memory exhibits some form of error-correction capability.

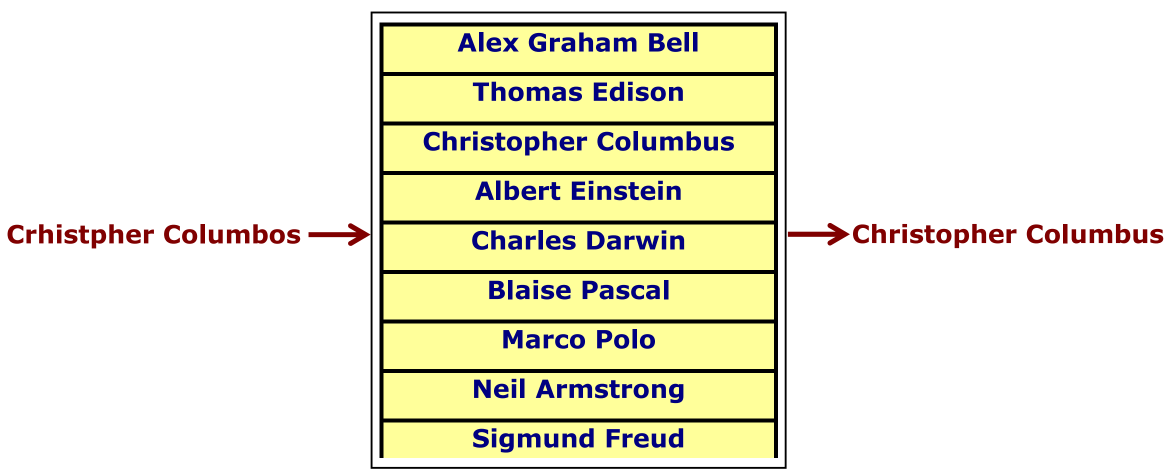


Fig. A content-addressable memory, Input and Output

Note : An associative memory is accessed by its content, opposed to an explicit address in the traditional computer memory system. The memory allows the recall of information based on partial knowledge of its contents.

[Continued in next slide]

[Continued from previous slide]

- Associative memory is a system that associates two patterns **(X, Y)** such that when one is encountered, the other can be recalled. The associative memory are of two types : auto-associative memory and hetero-associative memory.

  Auto-associative memory

  Consider, **y[1], y[2], y[3], ... y[M],** be the number of stored pattern vectors and let **y(m)** be the components of these vectors, representing features extracted from the patterns. The auto-associative memory will output a pattern vector **y(m)** when inputting a noisy or incomplete version of **y(m)**.

  Hetero-associative memory

  Here the memory function is more general. Consider, we have a number of key-response pairs **{c(1), y(1)}, {c(2), y(2)}, ... , {c(M), y(M)}**. The hetero-associative memory will output a pattern vector **y(m)** if a noisy or incomplete verson of the **c(m)** is given.

- Neural networks are used to implement associative memory models. The well-known neural associative memory models are :
  - **Linear associater** is the simplest artificial neural associative memory.
  - **Hopfield model** and **Bidirectional Associative Memory (BAM)** are the other popular ANN models used as associative memories.

  These models follow different neural network architectures to memorize information.

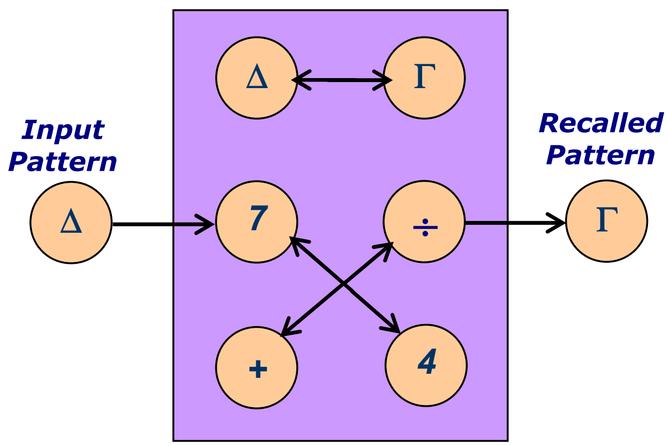
1.2 Working of Associative Memory

- **Example**

An associative memory is a storehouse of associated patterns which are encoded in some form.

- When the storehouse is triggered or excited with a pattern, then the associated pattern pair is recalled or appears at the output.
- The input could be an exact or distorted or partial representation of a stored pattern.

Fig below illustrates the working of an associated memory.



The associated pattern pairs

(Δ , Γ), (÷ , +), (7 , 4).

The association is represented by the symbol

The associated pattern pairs are stored the memory.

.

Fig. Working of an associated memory

When the memory is triggered with an input pattern say Δ then the associated pattern Γ is retrieved automatically.

1.3 Associative Memory - Classes

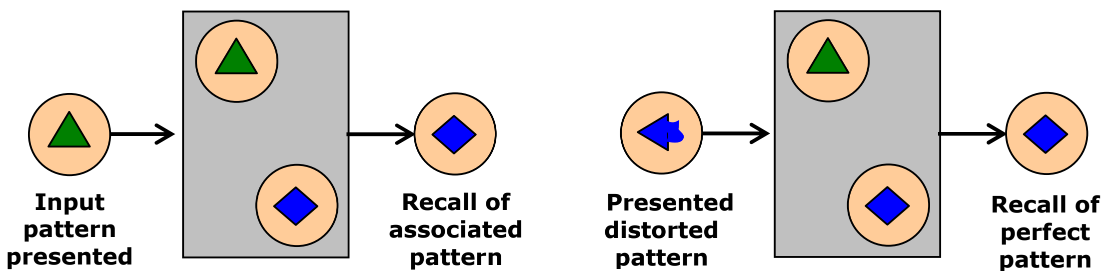
As stated before, there are two classes of associative memory:

- auto-associative and
- hetero-associative memory.

An auto-associative memory, also known as auto-associative correlator, is used to retrieve a previously stored pattern that most closely resembles the current pattern;

A hetero-associative memory, also known as hetero-associative correlator, is used to retrieve pattern in general, different from the input pattern not only in content but possibly also different in type and format.

Examples



Hetero-associative memory **Auto-associative memory**

Fig. Hetero and Auto Associative memory Correlators

1.4 Related Terms

Here explained : Encoding or memorization, Retrieval or recollection, Errors and Noise, Memory capacity and Content-addressability.

- **Encoding or memorization**

Building an associative memory means, constructing a connection weight matrix **W** such that when an input pattern is presented, and the stored pattern associated with the input pattern is retrieved.

This process of constructing the connection weight matrix is called **encoding.** During encoding, for an associated pattern pair (X_k, Y_k) , the weight values of the correlation matrix W_k are computed as

$(w_{ij})_k = (x_i)_k \ (y_j)_k$, where

$(x_i)_k$ represents the i^{th} component of pattern X_k , and

$(y_j)_k$ represents the j^{th} component of pattern Y_k

for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$.

Constructing of the connection weight matrix **W** is accomplished by summing up the individual correlation matrices W_k , i.e.,

$$W = \alpha \sum_{k=1}^{p} W_k \quad \text{where}$$

α is the proportionality or normalizing constant.

- **Retrieval or recollection**

After memorization, the process of retrieving a stored pattern, given an input pattern, is called **decoding**.

Given an input pattern **X**, the decoding or recollection is accomplished by:

first compute the net **input** to the output units using

$$\textbf{input}_j = \sum_{j=1}^{m} x_i \, w_{ij}$$

where $\textbf{input}_j$ is weighted sum of the input or activation value of node **j** , for **j = 1, 2, ..., n.**

then determine the units **output** using a bipolar output function:

$$Y_j = \begin{cases} +1 & \text{if } \textbf{input}_j \geq \theta_j \\ -1 & \text{other wise} \end{cases}$$

where θ_j is the threshold value of output neuron **j** .

- **Errors and noise**

  The input pattern may contain errors and noise, or may be an incomplete version of some previously encoded pattern.

  When a corrupted input pattern is presented, the network will retrieve the stored pattern that is closest to actual input pattern.

  The presence of noise or errors results only in a mere decrease rather than total degradation in the performance of the network.

  Thus, associative memories are robust and fault tolerant because of many processing elements performing highly parallel and distributed computations.

- **Performance Measures**

The memory capacity and content-addressability are the measures of associative memory performance for correct retrieval. These two performance measures are related to each other.

Memory capacity refers to the maximum number of associated pattern pairs that can be stored and correctly retrieved.

Content-addressability is the ability of the network to retrieve the correct stored pattern.

If input patterns are mutually orthogonal - perfect retrieval is possible.

If the stored input patterns are not mutually orthogonal - non-perfect retrieval can happen due to crosstalk among the patterns.

2. Associative Memory Models

An associative memory is a system which stores mappings of specific input representations to specific output representations.

- An associative memory "associates" two patterns such that when one is encountered, the other can be reliably recalled.
- Most associative memory implementations are realized as connectionist networks.

The simplest associative memory model is **Linear associator**, which is a feed-forward type of network. It has very low memory capacity and therefore not much used.

The popular models are **Hopfield Model** and **Bi-directional Associative Memory (BAM) model**.

The Network Architecture of these models are presented in this section.

2.1 Associative Memory Models

The simplest and among the first studied associative memory models is **Linear associator**. It is a feed-forward type of network where the output is produced in a single feed-forward computation. It can be used as an auto-associator as well as a hetero-associator, but it possesses a very low memory capacity and therefore not much used.

The popular associative memory models are **Hopfield Model** and **Bi-directional Associative Memory (BAM) model.**

- The Hopfield model is an auto-associative memory, proposed by John Hopfield in 1982. It is an ensemble of simple processing units that have a fairly complex collective computational abilities and behavior. The Hopfield model computes its output recursively in time until the system becomes stable. Hopfield networks are designed using bipolar units and a learning procedure.

- The Bi-directional associative memory (BAM) model is similar to linear associator, but the connections are bi-directional and therefore allows forward and backward flow of information between the layers. The BAM model can perform both auto-associative and hetero-associative recall of stored information.

The network architecture of these three models are described in the next few slides.

2.2 Network Architectures of AM Models

The neural associative memory models follow different neural network architectures to memorize information. The network architectures are either single layer or two layers .

- The **Linear associator model**, is a feed forward type network, consists, two layers of processing units, one serving as the input layer while the other as the output layer.

- The **Hopfield model**, is a single layer of processing elements where each unit is connected to every other unit in the network other than itself.

- The **Bi-directional associative memory (BAM) model** is similar to that of linear associator but the connections are bidirectional.

In this section, the neural network architectures of these models and the construction of the corresponding connection weight matrix **W** of the associative memory are illustrated.

- **Linear Associator Model** (two layers)

It is a feed-forward type network where the output is produced in a single feed-forward computation. The model consists of two layers of processing units, one serving as the input layer while the other as the output layer. The inputs are directly connected to the outputs, via a series of weights. The links carrying weights connect every input to every output. The sum of the products of the weights and the inputs is calculated in each neuron node. The network architecture of the linear associator is as shown below.

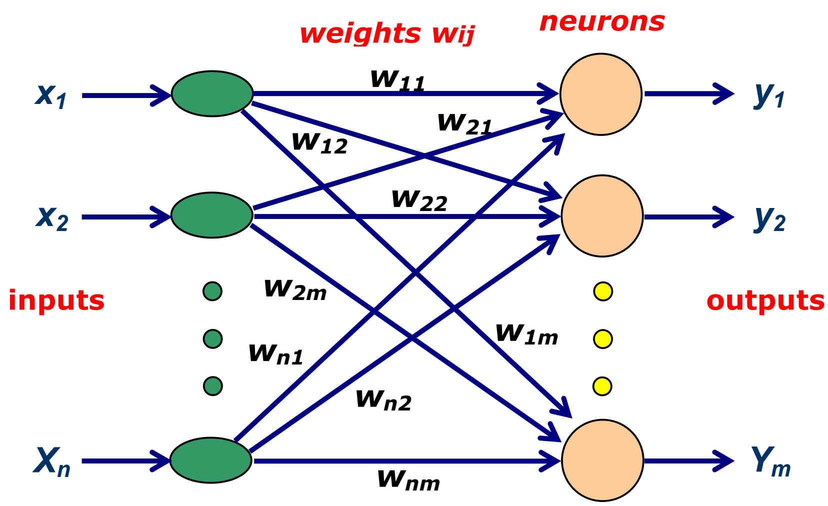


Fig. Linear associator model

- all **n** input units are connected to all **m** output units via connection weight matrix $\mathbf{W} = [w_{ij}]_{n \times m}$ where w_{ij} denotes the strength of the unidirectional connection from the **i** th input unit to the **j** th output unit.

- the connection weight matrix stores the **p** different associated pattern pairs $\{(X_k, Y_k) \mid k = 1, 2, \dots, p\}$.

- building an associative memory is constructing the connection weight matrix **W** such that when an input pattern is presented, then the stored pattern associated with the input pattern is retrieved.

[Continued in next slide]

[Continued from previous slide]

– **Encoding** : The process of constructing the connection weight matrix is called encoding. During encoding the weight values of correlation matrix W_k for an associated pattern pair (X_k, Y_k) are computed as:

$(w_{ij})_k = (x_i)_k \ (y_j)_k$ where

$(x_i)_k$ is the i^{th} component of pattern X_k for $i = 1, 2, ..., m$, and

$(y_j)_k$ is the j^{th} component of pattern Y_k for $j = 1, 2, ..., n$.

– **Weight matrix** : Construction of weight matrix W is accomplished by summing those individual correlation matrices W_k, ie, $W = \alpha \sum_{k=1}^{p} W_k$ where α is the constant of proportionality, for normalizing, usually set to $1/p$ to store p different associated pattern pairs.

– **Decoding** : After memorization, the network can be used for retrieval; the process of retrieving a stored pattern, is called decoding; given an input pattern X, the decoding or retrieving is accomplished by computing, first the net *Input* as $input_j = \sum_{j=1}^{m} x_i \ w_{ij}$ where $input_j$ stands for the weighted sum of the input or activation value of node j, for $j = 1, 2, . . , n.$ and x_i is the i^{th} component of pattern X_k, and then determine the units *Output* using a bipolar output function:

$$Y_j = \begin{cases} +1 & \text{if } input_j \geq \theta_j \\ -1 & \text{other wise} \end{cases}$$

where θ_j is the threshold value of output neuron j.

Note: The output units behave like linear threshold units; that compute a weighted sum of the input and produces a -1 or $+1$ depending whether the weighted sum is below or above a certain threshold value.

– **Performance** : The input pattern may contain errors and noise, or an incomplete version of some previously encoded pattern. When such corrupt input pattern is presented, the network will retrieve the stored pattern that is closest to actual input pattern. Therefore, the linear associator is robust and fault tolerant. The presence of noise or error results in a mere decrease rather than total degradation in the performance of the network.

- **Auto-associative Memory Model - Hopfield model** (single layer)

Auto-associative memory means patterns rather than associated pattern pairs, are stored in memory. Hopfield model is one-layer unidirectional auto-associative memory.

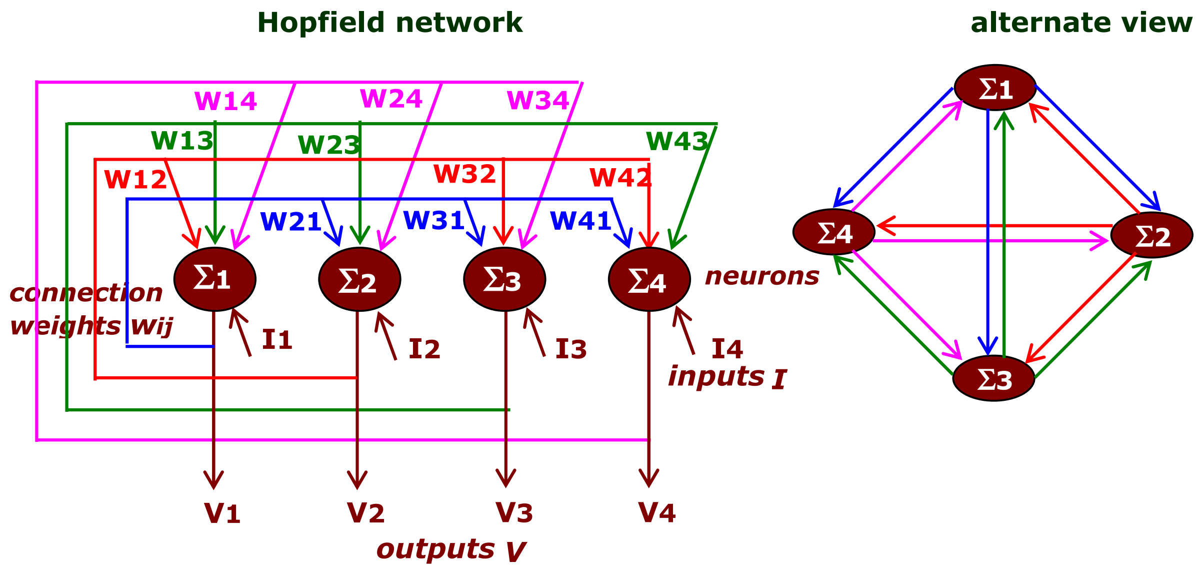


Fig. Hopfield model with four units

- the model consists, a single layer of processing elements where each unit is connected to every other unit in the network but not to itself.
- connection weight between or from neuron **j** to **i** is given by a number w_{ij}. The collection of all such numbers are represented by the weight matrix **W** which is square and symmetric, ie, $w_{ij} = w_{ji}$ for $i, j = 1, 2, \ldots, m$.
- each unit has an external input **I** which leads to a modification in the computation of the net input to the units as

$$\text{input}_j = \sum_{i=1}^{m} x_i w_{ij} + I_j \quad \text{for} \quad j = 1, 2, \ldots, m.$$

and x_i is the i^{th} component of pattern X_k

- each unit acts as both input and output unit. Like linear associator, a single associated pattern pair is stored by computing the weight matrix as $W_k = X_k^T Y_k$ where $X_K = Y_K$

[Continued in next slide]

[Continued from previous slide]

– **Weight Matrix** : Construction of weight matrix **W** is accomplished by summing those individual correlation matrices, ie, $\mathbf{W} = \alpha \sum_{k=1}^{p} \mathbf{W_k}$ where α is the constant of proportionality, for normalizing, usually set to **1/p** to store **p** different associated pattern pairs. Since the Hopfield model is an auto-associative memory model, it is the patterns rather than associated pattern pairs, are stored in memory.

– **Decoding** : After memorization, the network can be used for retrieval; the process of retrieving a stored pattern, is called decoding; given an input pattern **X**, the decoding or retrieving is accomplished by computing, first the net *Input* as $\text{input}_j = \sum_{j=1}^{m} x_i \; w_{ij}$ where input_j stands for the weighted sum of the input or activation value of node **j** , for **j = 1, 2, ..., n.** and x_i is the i^{th} component of pattern $\mathbf{X_k}$, and then determine the units *Output* using a bipolar output function:

$$Y_j = \begin{cases} +1 & \text{if input } j \geq \theta_j \\ -1 & \text{other wise} \end{cases}$$

where θ_j is the threshold value of output neuron **j** .

Note: The output units behave like linear threshold units; that compute a weighted sum of the input and produces a **-1** or **+1** depending whether the weighted sum is below or above a certain threshold value.

Decoding in the Hopfield model is achieved by a collective and recursive relaxation search for a stored pattern given an initial stimulus pattern. Given an input pattern **X**, decoding is accomplished by computing the net input to the units and determining the output of those units using the output function to produce the pattern **X'**. The pattern **X'** is then fed back to the units as an input pattern to produce the pattern **X''**. The pattern **X''** is again fed back to the units to produce the pattern **X'''**. The process is repeated until the network stabilizes on a stored pattern where further computations do not change the output of the units.

In the next section, the working of an auto-correlator : how to store patterns, recall a pattern from the stored patterns and how to recognize a noisy pattern are explained.

- **Bidirectional Associative Memory** (two-layer)

Kosko (1988) extended the Hopfield model, which is single layer, by incorporating an additional layer to perform recurrent auto-associations as well as hetero-associations on the stored memories. The network structure of the bidirectional associative memory model is similar to that of the linear associator but the connections are bidirectional; i.e.,

$w_{ij} = w_{ji}$, for $i = 1, 2, \dots, n$ and $j = 1, 2, \dots, m$.

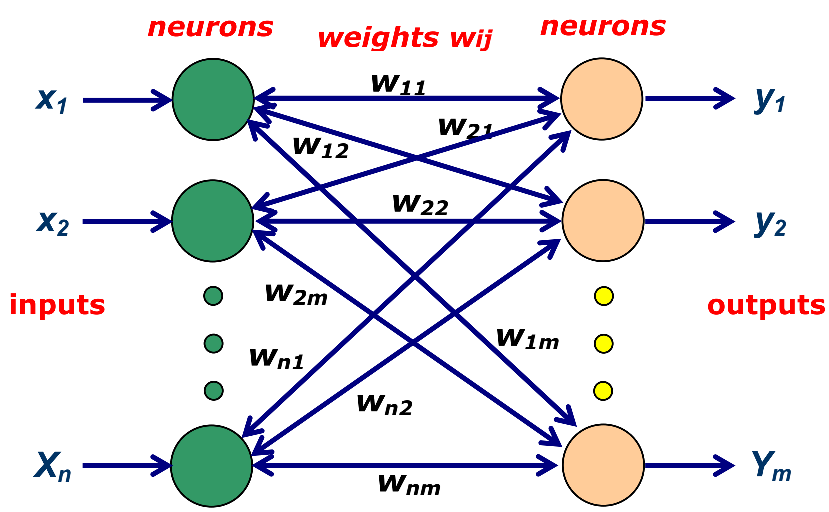


Fig. Bidirectional Associative Memory model

- In the bidirectional associative memory, a single associated pattern pair is stored by computing the weight matrix as $W_k = X_k^T Y_k$.
- the construction of the connection weight matrix W, to store p different associated pattern pairs simultaneously, is accomplished by summing up the individual correlation matrices W_k ,

  i.e., $W = \alpha \sum_{k=1}^{p} W_k$

  where α is the proportionality or normalizing constant.

3. Auto-associative Memory (auto-correlators)

In the previous section, the structure of the Hopfield model has been explained. It is an auto-associative memory model which means patterns, rather than associated pattern pairs, are stored in memory. In this section, the working of an auto-associative memory (auto-correlator) is illustrated using some examples.

Working of an auto-correlator :

- how to store the patterns,
- how to retrieve / recall a pattern from the stored patterns, and
- how to recognize a noisy pattern

- **How to Store Patterns : Example**

Consider the three bipolar patterns A_1 , A_2, A_3 to be stored as an auto-correlator.

$A_1 = (-1, 1, -1, 1)$

$A_2 = (1, 1, 1, -1)$

$A_3 = (-1, -1, -1, 1)$

Note that the outer product of two vectors **U** and **V** is

$$U \otimes V = U^T V = \begin{pmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{pmatrix} \begin{bmatrix} V_1 & V_2 & V_3 \end{bmatrix} = \begin{pmatrix} U_1V_1 & U_1V_2 & U_1V_3 \\ U_2V_1 & U_2V_2 & U_2V_3 \\ U_3V_1 & U_3V_2 & U_3V_3 \\ U_4V_1 & U_4V_2 & U_4V_3 \end{pmatrix}$$

Thus, the outer products of each of these three A_1 , A_2, A_3 bipolar patterns are

$$[A_1^T]_{4x1} \; [A_1]_{1x4} = \begin{array}{cc} & \xrightarrow{\qquad} j \\ \downarrow i & \begin{Bmatrix} 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \end{Bmatrix} \end{array}$$

$$[A_2^T]_{4x1} \; [A_2]_{1x4} = \begin{array}{cc} & \xrightarrow{\qquad} j \\ \downarrow i & \begin{Bmatrix} 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 \\ -1 & -1 & -1 & 1 \end{Bmatrix} \end{array}$$

$$[A_3^T]_{4x1} \; [A_3]_{1x4} = \begin{array}{cc} & \xrightarrow{\qquad} j \\ \downarrow i & \begin{Bmatrix} 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 \\ -1 & -1 & -1 & 1 \end{Bmatrix} \end{array}$$

Therefore the connection matrix is

$$T = [t_{ij}] = \sum_{i=1}^{3} [A_i^T]_{4x1} \; [A_i]_{1x4} \quad \begin{array}{cc} & \xrightarrow{\qquad} j \\ \downarrow i & \begin{Bmatrix} 3 & 1 & 3 & -3 \\ 1 & 3 & 1 & -1 \\ 3 & 1 & 3 & -3 \\ -3 & -1 & -3 & 3 \end{Bmatrix} \end{array}$$

This is how the patterns are stored .

- **Retrieve a Pattern from the Stored Patterns** (ref. previous slide)

The previous slide shows the connection matrix **T** of the three bipolar patterns A_1, A_2, A_3 stored as

$$T = [t_{ij}] = \sum_{i=1}^{3} [A_i]^T_{4x1} \; [A_i]_{1x4} = \begin{array}{c|c} & \xrightarrow{i} \\ j \downarrow & \begin{Bmatrix} 3 & 1 & 3 & -3 \\ 1 & 3 & 1 & -1 \\ 3 & 1 & 3 & -3 \\ -3 & -1 & -3 & 3 \end{Bmatrix} \end{array}$$

and one of the three stored pattern is $A_2 = (1, 1, 1, -1)$ ← a_i

- Retrieve or recall of this pattern A_2 from the three stored patterns.
- The recall equation is

$$a_j^{new} = f(a_i \; t_{ij}, \; a_j^{old}) \quad \text{for } \forall j = 1, 2, \ldots, p$$

Computation for the recall equation A_2 yields $\alpha = \sum a_i \, t_{ij}$ and then find β

$i =$ →	1		2		3		4	α	β
$\alpha = \sum a_i \; t_{i,j=1}$	1x3	+	1x1	+	1x3	+	-1x-3	= 10	1
$\alpha = \sum a_i \; t_{i,j=2}$	1x1	+	1x3	+	1x1	+	-1x-1	= 6	1
$\alpha = \sum a_i \; t_{i,j=3}$	1x3	+	1x1	+	1x3	+	-1x-3	= 10	1
$\alpha = \sum a_i \; t_{i,j=4}$	1x-3	+	1x-1	+	1x-3	+	-1x3	= -1	-1

Therefore $a_j^{new} = f(a_i \; t_{ij}, \; a_j^{old})$ for $\forall j = 1, 2, \ldots, p$ is $f(\alpha, \beta)$

$$a_1^{new} = f(10, 1)$$

$$a_2^{new} = f(6, 1)$$

$$a_3^{new} = f(10, 1)$$

$$a_4^{new} = f(-1, -1)$$

The values of β is the vector pattern $(1, 1, 1, -1)$ which is A_2.

This is how to retrieve or recall a pattern from the stored patterns.

Similarly, retrieval of vector pattern A_3 as

$$(a_1^{new}, \; a_2^{new}, \; a_3^{new}, \; a_4^{new}) = (-1, -1, -1, 1) = A_3$$

- **Recognition of Noisy Patterns** (ref. previous slide)

Consider a vector **A' = (1, 1 , 1 , 1)** which is a noisy presentation of one among the stored patterns.

– find the proximity of the noisy vector to the stored patterns using Hamming distance measure.

– note that the Hamming distance (HD) of a vector **X** from **Y**, where **X = (x1 , x2 , . . . , xn)** and **Y = (y1, y2 , . . . , yn)** is given by

$$HD(x, y) = \sum_{i=1}^{m} | (x_i - y_i) |$$

The HDs of **A'** from each of the stored patterns **A1 , A2, A3** are

$$HD(A', A_1) = \sum |(x_1 - y_1)|, |(x2 - y2)|, |(x_3 - y_3)|, |(x_4 - y_4)|$$
$$= \sum |(1 - (-1))|, |(1 - 1)|, |(1 - (-1))|, |(1 - 1)|$$
$$= 4$$
$$HD(A', A_2) = 2$$
$$HD(A', A_3) = 6$$

Therefore the vector **A'** is closest to **A2** and so resembles it.

In other words the vector **A'** is a noisy version of vector **A2**.

Computation of recall equation using vector **A'** yields :

$$\begin{array}{l|cccc|cc}
i = \longrightarrow & 1 & 2 & 3 & 4 & \alpha & \beta \\
\alpha = \sum ai\ t_{i,j=1} & 1x3 & + 1x1 & + 1x3 & + 1x\text{-}3 & = 4 & 1 \\
\alpha = \sum ai\ t_{i,j=2} & 1x1 & + 1x3 & + 1x1 & + 1x\text{-}1 & = 4 & 1 \\
\alpha = \sum ai\ t_{i,j=3} & 1x3 & + 1x1 & + 1x3 & + 1x\text{-}3 & = 4 & 1 \\
\alpha = \sum ai\ t_{i,j=4} & 1x\text{-}3 & + 1x\text{-}1 & + 1x\text{-}3 & + 1x3 & = \text{-}4 & \text{-}1
\end{array}$$

Therefore $a_j^{new} = f(a_i\ t_{ij}, a_j^{old})$ for $\forall j = 1, 2, \ldots, p$ is $f(\alpha, \beta)$

$$a_1^{new} = f(4, 1)$$
$$a_2^{new} = f(4, 1)$$
$$a_3^{new} = f(4, 1)$$
$$a_4^{new} = f(-4, -1)$$

The values of β is the vector pattern **(1, 1 , 1 , -1)** which is **A2 .**

Note : In presence of noise or in case of partial representation of vectors, an autocorrelator results in the refinement of the pattern or removal of noise to retrieve the closest matching stored pattern.

4. Bidirectional Hetero-associative Memory

The Hopfield one-layer unidirectional auto-associators have been discussed in previous section. Kosko (1987) extended this network to two-layer bidirectional structure called Bidirectional Associative Memory (BAM) which can achieve hetero-association. The important performance attributes of the BAM is its ability to recall stored pairs particularly in the presence of noise.

Definition : If the associated pattern pairs **(X, Y)** are different and if the model recalls a pattern **Y** given a pattern **X** or vice-versa, then it is termed as hetero-associative memory.

This section illustrates the bidirectional associative memory :

- Operations (retrieval, addition and deletion) ,
- Energy Function (Kosko's correlation matrix, incorrect recall of pattern),
- Multiple training encoding strategy (Wang's generalized correlation matrix).

4.1 Bidirectional Associative Memory (BAM) Operations

BAM is a two-layer nonlinear neural network.

Denote one layer as field **A** with elements A_i and the other layer as field **B** with elements B_i.

The basic coding procedure of the discrete BAM is as follows. Consider **N** training pairs $\{ (A_1 , B_1) , (A_2 , B_2), . . , (A_i , B_i), . . (A_N , B_N) \}$ where $A_i = (a_{i1} , a_{i2} , . . . , a_{in})$ and $B_i = (b_{i1} , b_{i2} , . . . , b_{ip})$ and a_{ij} , b_{ij} are either in **ON** or **OFF** state.

- in binary mode , **ON = 1** and **OFF = 0** and
  in bipolar mode, **ON = 1** and **OFF = -1**
- the original correlation matrix of the BAM is $M_0 = \sum_{i=1}^{N} [X_i^T] [Y_i]$
  where $X_i = (x_{i1} , x_{i2} , . . . , x_{in})$ and $Y_i = (y_{i1} , y_{i2} , . . . , y_{ip})$
  and $x_{ij}(y_{ij})$ is the bipolar form of $a_{ij}(b_{ij})$

The energy function **E** for the pair (α , β) and correlation matrix **M** is

$$E = - \alpha M \beta^T$$

With this background, the decoding processes, means the operations to **retrieve** nearest pattern pairs, and the **addition** and **deletion** of the pattern pairs are illustrated in the next few slides.

- **Retrieve the Nearest of a Pattern Pair, given any pair**

(ref : previous slide)

Example

Retrieve the nearest of (A_i , B_i) pattern pair, given any pair (α , β) .

The methods and the equations for retrieve are :

- start with an initial condition which is any given pattern pair (α , β),
- determine a finite sequence of pattern pairs (α' , β') , (α'' , β'') . . . . until an equilibrium point (α_f , β_f) is reached, where

$\beta' = \Phi(\alpha M)$ and $\alpha' = \Phi(\beta' M^T)$

$\beta'' = \Phi(\alpha' M)$ and $\alpha'' = \Phi(\beta'' M^T)$

$\Phi(F) = G = g_1 , g_2 , ... , g_r$,

$F = (f_1 , f_2 , ... , f_r)$

M is correlation matrix

$$g_i = \begin{cases} 1 & \text{if } f_i > 0 \\ \begin{cases} 0 \text{ (binary)} \\ -1 \text{ (bipolar)} \end{cases}, & f_i < 0 \\ \text{previous } g_i , & f_i = 0 \end{cases}$$

Kosko has proved that this process will converge for any correlation matrix **M**.

- **Addition and Deletion of Pattern Pairs**

Given a set of pattern pairs **(X_i , Y_i) , for $i = 1 , 2, . . . , n$** and a set of correlation matrix **M** :

- a new pair **(X' , Y')** can be added or
- an existing pair **(X_j , Y_j)** can be deleted from the memory model.

Addition : add a new pair **(X' , Y')** , to existing correlation matrix **M** , them the new correlation matrix **M_{new}** is given by

$$M_{new} = X_1^T \; Y_1 + X_1^T \; Y_1 + ... + X_n^T \; Y_n + X'^T \; Y'$$

Deletion : subtract the matrix corresponding to an existing pair **(X_j , Y_j)** from the correlation matrix **M** , them the new correlation matrix **M_{new}** is given by

$$M_{new} = M - (X_j^T \; Y_j)$$

Note : The addition and deletion of information is similar to the functioning of the system as a human memory exhibiting learning and forgetfulness.

4.2 Energy Function for BAM

Note : A system that changes with time is a dynamic system. There are two types of dynamics in a neural network. During training phase it iteratively update weights and during production phase it asymptotically converges to the solution patterns. State is a collection of qualitative and qualitative items that characterize the system e.g., weights, data flows. The Energy function (or Lyapunov function) is a bounded function of the system state that decreases with time and the system solution is the minimum energy.

Let a pair **(A , B)** defines the state of a BAM.

- to store a pattern, the value of the energy function for that pattern has to occupy a minimum point in the energy landscape.
- also adding a new patterns must not destroy the previously stored patterns.

The stability of a BAM can be proved by identifying the energy function **E** with each state **(A , B)** .

- For auto-associative memory : the energy function is

  $$E(A) = - AM\, A^T$$

- For bidirecional hetero associative memory : the energy function is

  $E(A, B) = - AM\, B^T$; for a particular case **A = B** , it corresponds to Hopfield auto-associative function.

We wish to retrieve the nearest of (A_i , B_i) pair, when any (α , β) pair is presented as initial condition to BAM. The neurons change their states until a bidirectional stable state (A_f , B_f) is reached. Kosko has shown that such stable state is reached for any matrix **M** when it corresponds to local minimum of the energy function. Each cycle of decoding lowers the energy **E** if the energy function for any point (α , β) is given by $E = \alpha\, M\, \beta^T$

If the energy $E = A_i\, M\, B_i^T$ evaluated using coordinates of the pair (A_i , B_i) does not constitute a local minimum, then the point cannot be recalled, even though one starts with $\alpha = A_i$. Thus Kosko's encoding method does not ensure that the stored pairs are at a local minimum.

- **Example : Kosko's BAM for Retrieval of Associated Pair**

The working of Kosko's BAM for retrieval of associated pair.
Start with X_3, and hope to retrieve the associated pair Y_3 .

Consider **N = 3** pattern pairs (A_1, B_1), (A_2, B_2), (A_3, B_3) given by

A_1 = (1 0 0 0 0 1) B_1 = (1 1 0 0 0)
A_2 = (0 1 1 0 0 0) B_2 = (1 0 1 0 0)
A_3 = (0 0 1 0 1 1) B_3 = (0 1 1 1 0)

Convert these three binary pattern to bipolar form replacing **0s** by **-1s**.

X_1 = (1 -1 -1 -1 -1 1) Y_1 = (1 1 -1 -1 -1)
X_2 = (-1 1 1 -1 -1 -1) Y_2 = (1 -1 1 -1 -1)
X_3 = (-1 -1 1 -1 1 1) Y_3 = (-1 1 1 1 -1)

The correlation matrix **M** is calculated as 6x5 matrix

$$M = X_1^T\ Y_1 + X_2^T\ Y_2 + X_3^T\ Y_3 = \begin{pmatrix} 1 & 1 & -3 & -1 & 1 \\ 1 & -3 & 1 & -1 & 1 \\ -1 & -1 & 3 & 1 & -1 \\ -1 & -1 & -1 & 1 & 3 \\ -3 & 1 & 1 & 3 & 1 \\ -1 & 3 & -1 & 1 & -1 \end{pmatrix}$$

Suppose we start with $\alpha = X_3$, and we hope to retrieve the associated pair Y_3 . The calculations for the retrieval of Y_3 yield :

$$\alpha M = (-1\ -1\ 1\ -1\ 1\ 1)\ (M) = (-6\ 6\ 6\ 6\ -6)$$
$$\Phi(\alpha M) = \beta' = (-1\ 1\ 1\ 1\ -1)$$
$$\beta' M^T = (-5\ -5\ 5\ -3\ 7\ 5)$$
$$\Phi(\beta' M^T) = (-1\ -1\ 1\ -1\ 1\ 1) = \alpha'$$
$$\alpha' M = (-1\ -1\ 1\ -1\ 1\ 1)\ M = (-6\ 6\ 6\ 6\ -6)$$
$$\Phi(\alpha' M) = \beta'' = (-1\ 1\ 1\ 1\ 1\ -1)$$
$$= \beta'$$

This retrieved patern β' is same as Y_3 .
Hence, $(\alpha_f , \beta_f) = (X_3 , Y_3)$ is correctly recalled, a desired result .

- **Example : Incorrect Recall by Kosko's BAM**

The Working of incorrect recall by Kosko's BAM.

Start with X_2, and hope to retrieve the associated pair Y_2 .

Consider **N = 3** pattern pairs (A_1, B_1), (A_2, B_2), (A_3, B_3) given by

$A_1 = (\ 1\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ )$ $\quad B_1 = (\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ )$

$A_2 = (\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 1\ )$ $\quad B_2 = (\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ )$

$A_3 = (\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ )$ $\quad B_3 = (\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ )$

Convert these three binary pattern to bipolar form replacing **0s** by **-1s**.

$X_1 = (\ 1\ -1\ -1\ 1\ 1\ 1\ -1\ -1\ -1\ )$ $\quad Y_1 = (\ 1\ 1\ 1\ -1\ -1\ -1\ -1\ 1\ -1\ )$

$X_2 = (\ -1\ 1\ 1\ 1\ -1\ -1\ 1\ 1\ 1\ )$ $\quad Y_2 = (\ 1\ -1\ -1\ -1\ -1\ -1\ -1\ -1\ 1\ )$

$X_3 = (\ 1\ -1\ 1\ -1\ 1\ 1\ -1\ 1\ 1\ )$ $\quad Y_3 = (\ -1\ 1\ -1\ 1\ -1\ -1\ 1\ 0\ 1\ )$

The correlation matrix **M** is calculated as 9 x 9 matrix

$$M = X_1^T\ Y_1 + X_2^T\ Y_2 + X_3^T\ Y_3$$

$$= \begin{pmatrix} -1 & 3 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \\ 1 & -3 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ -1 & -1 & -3 & 1 & -1 & -1 & 1 & -3 & 3 \\ 3 & -1 & 1 & -3 & -1 & -1 & -3 & 1 & -1 \\ -1 & 3 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \\ -1 & 3 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \\ 1 & -3 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ -1 & -1 & -3 & 1 & -1 & -1 & 1 & -3 & 3 \\ -1 & -1 & -3 & 1 & -1 & -1 & 1 & -3 & 3 \end{pmatrix}$$

(Continued in next slide)

[Continued from previous slide]

Let the pair (X_2 , Y_2) be recalled.

$X_2 =$ (-1 1 1 1 -1 -1 1 1 1) $Y_2 =$ (1 -1 -1 -1 -1 -1 -1 -1 1)

Start with $\alpha = X_2$, and hope to retrieve the associated pair Y_2 .

The calculations for the retrieval of Y_2 yield :

α M	=	(	5	-19	-13	-5	1	1	-5	-13	13	)		
Φ (α M)	=	(	1	-1	-1	-1	1	1	-1	-1	1	)	=	β'
β' M^T	=	(	-11	11	5	5	-11	-11	11	5	5	)		
Φ (β' M^T)	=	(	-1	1	1	1	-1	-1	1	1	1	)	=	α'
α' M	=	(	5	-19	-13	-5	1	1	-5	-13	13	)		
Φ (α' M)	=	(	1	-1	-1	-1	1	1	-1	-1	1	)	=	β''
	=	β'												

Here β'' = β' . Hence the cycle terminates with

α_F	=	α'	=	(	-1	1	1	1	-1	-1	1	1	1	)	= X_2
β_F	=	β'	=	(	1	-1	-1	-1	1	1	-1	-1	1	)	$\neq Y_2$

But β' is not Y_2 . Thus the vector pair (X2 , Y2) is not recalled correctly by Kosko's decoding process.

(Continued in next slide)

[Continued from previous slide]

Check with Energy Function : Compute the energy functions

for the coordinates of pair (X_2 , Y_2) , the energy $E_2 = - X_2 \, M \, Y_2^T = -71$

for the coordinates of pair (α_F , β_F) , the energy $E_F = - \alpha_F \, M \, \beta_F^T = -75$

However, the coordinates of pair (X_2 , Y_2) is not at its local minimum can be shown by evaluating the energy **E** at a point which is "one Hamming distance" way from Y_2 . To do this consider a point

$$Y_2' = (1 \quad -1 \quad -1 \quad -1 \quad 1 \quad -1 \quad -1 \quad -1 \quad 1)$$

where the fifth component **-1** of Y_2 has been changed to **1**. Now

$$E = - X_2 \, M \, {Y_2'}^T = - 73$$

which is lower than E_2 confirming the hypothesis that (X_2 , Y_2) is not at its local minimum of **E**.

Note : The correlation matrix **M** used by Kosko does not guarantee that the energy of a training pair is at its local minimum. Therefore , a pair **Pi** can be recalled if and only if this pair is at a local minimum of the energy surface**.**

4.3 Multiple Training Encoding Strategy

Note : (Ref. example in previous section). Kosko extended the unidirectional auto-associative to bidirectional associative processes, using correlation matrix $\mathbf{M} = \Sigma\ X_i^T\ Y_i$ computed from the pattern pairs. The system proceeds to retrieve the nearest pair given any pair $(\alpha\ ,\ \beta\ )$, with the help of recall equations. However, Kosko's encoding method does not ensure that the stored pairs are at local minimum and hence, results in incorrect recall.

Wang and other's, introduced **multiple training** encoding strategy which ensures the correct recall of pattern pairs. This encoding strategy is an enhancement / generalization of Kosko's encoding strategy. The Wang's generalized correlation matrix is $\mathbf{M} = \Sigma\ \mathbf{q_i}\ X_i^T\ Y_i$ where $\mathbf{q_i}$ is viewed as pair weight for $X_i^T\ Y_i$ as positive real numbers. It denotes the minimum number of times for using a pattern pair $(X_i\ ,\ Y_i)$ for training to guarantee recall of that pair.

To recover a pair $(A_i\ ,\ B_i)$ using multiple training of order $\mathbf{q}$, let us augment or supplement matrix $\mathbf{M}$ with a matrix $\mathbf{P}$ defined as

$\mathbf{P} = (q - 1)\ X_i^T\ Y_i$ where $(X_i\ ,\ Y_i)$ are the bipolar form of $(A_i\ ,\ B_i)$.

The augmentation implies adding $(q - 1)$ more pairs located at $(A_i\ ,\ B_i)$ to the existing correlation matrix. As a result the energy $\mathbf{E'}$ can reduced to an arbitrarily low value by a suitable choice of $\mathbf{q}$. This also ensures that the energy at $(A_i\ ,\ B_i)$ does not exceed at points which are one Hamming distance away from this location.

The new value of the energy function $\mathbf{E}$ evaluated at $(A_i\ ,\ B_i)$ then becomes

$$E'\ (A_i\ ,\ B_i)\ = -\ A_i\ M\ B_i^T\ -\ (q - 1)\ A_i\ X_i^T\ Y_i\ B_i^T$$

The next few slides explains the step-by-step implementation of Multiple training encoding strategy for the recall of three pattern pairs $(X_1\ ,\ Y_1\ )\ ,\ (X_1\ ,\ Y_1\ )\ ,\ (X_1\ ,\ Y_1\ )$ using one and same augmentation matrix $\mathbf{M}$. Also an algorithm to summarize the complete process of multiple training encoding is given.

- **Example : Multiple Training Encoding Strategy**

The working of multiple training encoding strategy which ensures the correct recall of pattern pairs.

Consider **N = 3** pattern pairs **(A_1, B_1), (A_2, B_2), (A_3, B_3)** given by

A_1 = (1 0 0 1 1 1 0 0 0) B_1 = (1 1 1 0 0 0 0 1 0)

A_2 = (0 1 1 1 0 0 1 1 1) B_2 = (1 0 0 0 0 0 0 0 1)

A_3 = (1 0 1 0 1 1 0 1 1) B_3 = (0 1 0 1 0 0 1 0 1)

Convert these three binary pattern to bipolar form replacing **0s** by **-1s**.

X_1 = (1 -1 -1 1 1 1 -1 -1 -1) Y_1 = (1 1 1 -1 -1 -1 -1 1 -1)

X_2 = (-1 1 1 1 -1 -1 1 1 1) Y_2 = (1 -1 -1 -1 -1 -1 -1 -1 1)

X_3 = (1 -1 1 -1 1 1 -1 1 1) Y_3 = (-1 1 -1 1 -1 -1 1 0 1)

Let the pair **(X_2 , Y_2)** be recalled.

X_2 = (-1 1 1 1 -1 -1 1 1 1) Y_2 = (1 -1 -1 -1 -1 -1 -1 -1 1)

Choose **q=2**, so that **P =** $X_2^T \; Y_2$ the augmented correlation matrix **M** becomes **M** $= X_1^T \; Y_1 \; + \; 2 \; X_2^T \; Y_2 + \; X_3^T \; Y_3$

$$
= \begin{pmatrix}
 & 4 & 2 & 2 & 0 & 0 & 2 & 2 & -2 \\
2 & -4 & -2 & -2 & 0 & 0 & -2 & -2 & 2 \\
0 & -2 & -4 & 0 & -2 & -2 & 0 & -4 & 4 \\
4 & -2 & 0 & -4 & -2 & -2 & -4 & 0 & 0 \\
-2 & 4 & 2 & 2 & 0 & 0 & 2 & 2 & -2 \\
-2 & 4 & 2 & 2 & 0 & 0 & 2 & 2 & -2 \\
2 & -4 & -2 & -2 & 0 & 0 & -2 & -2 & 2 \\
0 & -2 & -4 & 0 & -2 & -2 & 0 & -4 & 4 \\
0 & -2 & -4 & 0 & -2 & -2 & 0 & -4 & 4
\end{pmatrix}
$$

(Continued in next slide)

[Continued from previous slide]

Now give $\alpha = X_2$, and see that the corresponding pattern pair $\beta = Y_2$ is correctly recalled as shown below.

$$\alpha M = (\ 14\ \ -28\ \ -22\ \ -14\ \ -8\ \ -8\ \ -14\ \ -22\ \ 22\ )$$
$$\Phi(\alpha M) = (\ 1\ \ -1\ \ -1\ \ -1\ \ -1\ \ -1\ \ -1\ \ -1\ \ 1\ ) = \beta'$$
$$\beta' M^T = (\ -16\ \ 16\ \ 18\ \ 18\ \ -16\ \ -16\ \ 16\ \ 18\ \ 18\ )$$
$$\Phi(\beta' M^T) = (\ -1\ \ 1\ \ 1\ \ 1\ \ -1\ \ -1\ \ 1\ \ 1\ \ 1\ ) = \alpha'$$
$$\alpha' M = (\ 14\ \ -28\ \ -22\ \ -14\ \ -8\ \ -8\ \ -14\ \ -22\ \ 23\ )$$
$$\Phi(\alpha' M) = (\ 1\ \ -1\ \ -1\ \ -1\ \ 1\ \ 1\ \ -1\ \ -1\ \ 1\ ) = \beta''$$

Here $\beta'' = \beta'$. Hence the cycle terminates with

$$\alpha_F = \alpha' = (\ -1\ \ 1\ \ 1\ \ 1\ \ -1\ \ -1\ \ 1\ \ 1\ \ 1\ ) = X_2$$
$$\beta_F = \beta' = (\ 1\ \ -1\ \ -1\ \ -1\ \ 1\ \ 1\ \ -1\ \ -1\ \ 1\ ) = Y_2$$

Thus, $(X_2\ ,\ Y_2)$ is correctly recalled, using augmented correlation matrix M. But, it is not possible to recall $(X_1\ ,\ Y_1)$ using the same matrix M as shown in the next slide.

(Continued in next slide)

[Continued from previous slide]

Note : The previous slide showed that the pattern pair **(X_2 , Y_2)** is correctly recalled, using augmented correlation matrix

$$M = X_1^T \; Y_1 + 2 \; X_2^T \; Y_2 + X_3^T \; Y_3$$

but then the same matrix **M** can not recall correctly the other pattern pair **(X_1 , Y_1)** as shown below.

X_1 = (1 -1 -1 1 1 1 -1 -1 -1) Y_1 = (1 1 1 -1 -1 -1 -1 1 -1)

Let $\alpha = X_1$ and to retrieve the associated pair Y_1 the calculation shows

α M	= (	-6	24	22	6	4	4	6	22	-22	)		
Φ (α M)	= (	-1	1	1	1	1	1	1	1	-1	)	=	β'
β' M^T	= (	16	-16	-18	-18	16	16	-16	-18	-18	)		
Φ (β' M^T)	= (	1	-1	-1	-1	1	1	-1	-1	-1	)	=	α'
α' M	= (	-14	28	22	14	8	8	14	22	-22	)		
Φ (α' M)	= (	-1	1	1	1	1	1	1	1	-1	)	=	β''

Here $\beta'' = \beta'$. Hence the cycle terminates with

$\alpha_F = \alpha'$	= (	1	-1	-1	-1	1	1	-1	-1	-1	)	=	X_1
$\beta_F = \beta'$	= (	-1	1	1	1	1	1	1	1	-1	)	$\neq$	Y_1

Thus, the pattern pair **(X_1 , Y_1)** is not correctly recalled, using augmented correlation matrix **M**.

To tackle this problem, the correlation matrix **M** needs to be further augmented by **multiple training** of **(X_1 , Y_1)** as shown in the next slide.

(Continued in next slide)

[Continued from previous slide]

The previous slide shows that pattern pair (X_1 , Y_1) cannot be recalled under the same augmentation matrix **M** that is able to recall (X_2 , Y_2).

However, this problem can be solved by **multiple training** of (X_1 , Y_1) which yields a further change in **M** to values by defining

$$M = 2\, X_1^T\, Y_1 + 2\, X_2^T\, Y_2 + X_3^T\, Y_3$$

$$= \begin{pmatrix} -1 & 5 & 3 & 1 & -1 & -1 & 1 & 3 & -3 \\ 1 & -5 & -3 & -1 & 1 & 1 & -1 & -3 & 3 \\ -1 & -3 & -5 & 1 & -1 & -1 & 1 & -5 & 5 \\ 5 & -1 & 1 & -5 & -3 & -3 & -5 & 1 & -1 \\ -1 & 5 & 3 & 1 & -1 & -1 & 1 & 3 & -3 \\ -1 & 5 & 3 & 1 & -1 & -1 & 1 & 3 & -3 \\ 1 & -5 & -3 & -1 & 1 & 1 & -1 & -3 & 3 \\ -1 & -3 & -5 & 1 & -1 & -1 & 1 & -5 & 5 \\ -1 & -3 & -5 & 1 & -1 & -1 & 1 & -5 & 5 \end{pmatrix}$$

Now observe in the next slide that all three pairs can be correctly recalled.

(Continued in next slide)

[Continued from previous slide]

Recall of pattern pair (X1 , Y1)

$X_1 = (1\ -1\ -1\ 1\ 1\ 1\ -1\ -1\ -1)$ $\qquad$ $Y_1 = (1\ 1\ 1\ -1\ -1\ -1\ -1\ 1\ -1)$

Let $\alpha = X_1$ and to retrieve the associated pair Y_1 the calculation shows

$$
\begin{aligned}
\alpha M &= (3\ 33\ 31\ -3\ -5\ -5\ -3\ 31\ -31) \\
\Phi(\alpha M) &= (1\ 1\ 1\ -1\ -1\ -1\ -1\ 1\ -1) = \beta' \\
(\beta' M^T) &= (13\ -13\ -19\ 23\ 13\ 13\ -13\ -19\ -19) \\
\Phi(\beta' M^T) &= (1\ -1\ -1\ 1\ 1\ 1\ -1\ -1\ -1) = \alpha' \\
\alpha' M &= (3\ 33\ 31\ -3\ -5\ -5\ -3\ 31\ -31) \\
\Phi(\alpha' M) &= (1\ 1\ 1\ -1\ -1\ -1\ -1\ 1\ -1) = \beta''
\end{aligned}
$$

Here $\beta'' = \beta'$. Hence the cycle terminates with

$$
\begin{aligned}
\alpha_F &= \alpha' = (1\ -1\ -1\ 1\ 1\ 1\ -1\ -1\ -1) = X_1 \\
\beta_F &= \beta' = (1\ 1\ 1\ -1\ -1\ -1\ -1\ 1\ -1) = Y_1
\end{aligned}
$$

Thus, the pattern pair ***(X1 , Y1)*** is correctly recalled

Recall of pattern pair (X2 , Y2)

$X_2 = (-1\ 1\ 1\ 1\ -1\ -1\ 1\ 1\ 1)$ $\qquad$ $Y_2 = (1\ -1\ -1\ -1\ -1\ -1\ -1\ -1\ 1)$

Let $\alpha = X_2$ and to retrieve the associated pair Y_2 the calculation shows

$$
\begin{aligned}
\alpha M &= (7\ -35\ -29\ -7\ -1\ -1\ -7\ -29\ 29) \\
\Phi(\alpha M) &= (1\ -1\ -1\ -1\ -1\ -1\ -1\ -1\ 1) = \beta' \\
(\beta' M^T) &= (-15\ 15\ 17\ 19\ -15\ -15\ 15\ 17\ 17) \\
\Phi(\beta' M^T) &= (-1\ 1\ 1\ 1\ -1\ -1\ 1\ 1\ 1) = \alpha' \\
\alpha' M &= (7\ -35\ -29\ -7\ -1\ -1\ -7\ -29\ 29) \\
\Phi(\alpha' M) &= (1\ -1\ -1\ -1\ -1\ -1\ -1\ -1\ 1) = \beta''
\end{aligned}
$$

Here $\beta'' = \beta'$. Hence the cycle terminates with

$$
\begin{aligned}
\alpha_F &= \alpha' = (-1\ 1\ 1\ 1\ -1\ -1\ 1\ 1\ 1) = X_2 \\
\beta_F &= \beta' = (1\ -1\ -1\ -1\ -1\ -1\ -1\ -1\ 1) = Y_2
\end{aligned}
$$

Thus, the pattern pair ***(X2 , Y2)*** is correctly recalled

Recall of pattern pair (X3 , Y3)

$X_3 = (1\ -1\ 1\ -1\ 1\ 1\ -1\ 1\ 1)$ $\qquad$ $Y_3 = (-1\ 1\ -1\ 1\ -1\ -1\ 1\ 0\ 1)$

Let $\alpha = X_3$ and to retrieve the associated pair Y_3 the calculation shows

$$
\begin{aligned}
\alpha M &= (-13\ 17\ -1\ 13\ -5\ -5\ 13\ -1\ 1) \\
\Phi(\alpha M) &= (-1\ 1\ -1\ 1\ -1\ -1\ 1\ -1\ 1) = \beta' \\
(\beta' M^T) &= (11\ -11\ 27\ -63\ 11\ 11\ -11\ 27\ 27) \\
\Phi(\beta' M^T) &= (1\ -1\ 1\ -1\ 1\ 1\ -1\ 1\ 1) = \alpha' \\
\alpha' M &= (-13\ 17\ -1\ 13\ -5\ -5\ 13\ -1\ 1) \\
\Phi(\alpha' M) &= (-1\ 1\ -1\ 1\ -1\ -1\ 1\ -1\ 1) = \beta''
\end{aligned}
$$

Here $\beta'' = \beta'$. Hence the cycle terminates with

$$
\begin{aligned}
\alpha_F &= \alpha' = (1\ -1\ 1\ -1\ 1\ 1\ -1\ 1\ 1) = X_3 \\
\beta_F &= \beta' = (-1\ 1\ -1\ 1\ -1\ -1\ 1\ 0\ 1) = Y_3
\end{aligned}
$$

Thus, the pattern pair ***(X3 , Y3)*** is correctly recalled

(Continued in next slide)

[Continued from previous slide]

Thus, the multiple training encoding strategy ensures the correct recall of a pair for a suitable augmentation of **M** . The generalization of the correlation matrix, for the correct recall of all training pairs, is written as

$$\mathbf{M} = \sum_{i=1}^{N} q_i \, X_i^T \, Y_i$$ where q_i 's are **+ve** real numbers.

This modified correlation matrix is called generalized correlation matrix. Using one and same augmentation matrix **M**, it is possible to recall all the training pattern pairs .

- **Algorithm** (for the Multiple training encoding strategy)

To summarize the complete process of multiple training encoding an algorithm is given below.

Algorithm Mul_Tr_Encode (N , $\bar{X}_i$, $\bar{Y}_i$, $\bar{q}_i$) where

N : Number of stored patterns set

$\bar{X}_i$ $\bar{Y}_i$ the bipolar pattern pairs

$\bar{X}$: ($\bar{X}_1$, $\bar{X}_2$, ... , $\bar{X}_N$ where $\bar{X}_i = (x_{i\,1}, x_{i\,2}, \dots x_{i\,n})$

$\bar{Y}$: ($\bar{Y}_1$, $\bar{Y}_2$, ... , $\bar{Y}_N$ where $\bar{Y}_j = (x_{j\,1}, x_{j\,2}, \dots x_{j\,n})$

$\bar{q}$: is the weight vector $(q_1, q_2, \dots, q_N)$

Step 1 Initialize correlation matrix M to null matrix $M \leftarrow [0]$

Step 2 Compute the correlation matrix M as

For $i \leftarrow 1$ to N

$M \leftarrow M \oplus [\, qi * \text{Transpose}(\bar{X}_i) \otimes (\bar{X}_i)$ end

(symbols $\oplus$ matrix addition, $\otimes$ matrix multiplication, and $*$ scalar multiplication)

Step 3 Read input bipolar pattern $\bar{A}$

Step 4 Compute A_M where $A\_M \leftarrow \bar{A} \otimes M$

Step 5 Apply threshold function Φ to A_M to get $\bar{B}'$

ie $\bar{B}' \leftarrow \Phi(A\_M)$

where Φ is defined as $\Phi(F) = G = g_1, g_2, \dots, g_n$

Step 6 Output $\bar{B}'$ is the associated pattern pair

end

5. References : Textbooks

1. *"Neural Network, Fuzzy Logic, and Genetic Algorithms - Synthesis and Applications", by S. Rajasekaran and G.A. Vijayalaksmi Pai, (2005), Prentice Hall, Chapter 4, page 87-116.*

2. *"Elements of Artificial Neural Networks", by Kishan Mehrotra, Chilukuri K. Mohan and Sanjay Ranka, (1996), MIT Press, Chapter 6, page 217-263.*

3. *"Fundamentals of Neural Networks: Architecture, Algorithms and Applications", by Laurene V. Fausett, (1993), Prentice Hall, Chapter 3, page 101-152.*

4. *"Neural Network Design", by Martin T. Hagan, Howard B. Demuth and Mark Hudson Beale, (1996) , PWS Publ. Company, Chapter 13, page 13-1 to 13-37.*

5. *"An Introduction to Neural Networks", by James A. Anderson, (1997), MIT Press, Chapter 6-7, page 143-208.*

6. *Related documents from open source, mainly internet. An exhaustive list is being prepared for inclusion at a later date.*