NEURAL NETWORKS
MACHINE LEARNING

ANN ee543

# CHAPTER III

# *Neural Networks as Associative Memory*

# CHAPTER III : *Neural Networks as Associative Memory*

## *Introduction*

- One of the primary functions of the brain is associative memory. We associate the faces with names, letters with sounds, or we can recognize the people even if they have sunglasses or if they are somehow elder now.

- In this chapter, first the **basic definitions** about associative memory will be given and then it will be explained how neural networks can be made **linear associators** so as to perform as **interpolative memory**.

- Next it will be explained how the **Hopfield network** can be used as **autoassociative** memory.

- Then **Bipolar Associative Memory** network that is designed to operate as **hetero-associative** memory will be introduced.

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.1. Associative Memory*

- In an **associative memory** we store a set of patterns $\mu^k$, $k=1...K$, so that the network responds by producing whichever of the stored patterns **most closely resembles** the one presented to the network

- Suppose that the stored patterns, which are called **exemplars** or memory elements, are in the form of pairs of associations, $\boldsymbol{\mu}^k=(\mathbf{u}^k,\mathbf{y}^k)$ where $\mathbf{u}^k \in R^N$, $\mathbf{y}^k \in R^M$, $k=1..K$.

- According to the mapping $\varphi: R^N \rightarrow R^M$ implemented, we distinguish the following types of associative memories:

  - **Interpolative associative** memory
  - **Accretive associative** memory

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.1. Associative Memory: Interpolative Memory*

- In **interpolative associative memory**,

  When $\mathbf{u}=\mathbf{u}^r$ is presented to the memory it responds by producing $\mathbf{y}^r$ of the stored association.

  However if $\mathbf{u}$ differs from $\mathbf{u}^r$ by an amount of $\boldsymbol{\varepsilon}$, that is if $\mathbf{u}=\mathbf{u}^r+\boldsymbol{\varepsilon}$ is presented to the memory, then the response differs from $\mathbf{y}^r$ by some amount $\boldsymbol{\varepsilon}^r$.

  Therefore in interpolative associative memory we have

$$\varphi(\mathbf{u}^r+\varepsilon)=\mathbf{y}^r+\varepsilon^r \quad such \quad that \quad \varepsilon=\mathbf{0}\Rightarrow\varepsilon^r=\mathbf{0}, \quad r=1..K \tag{3.1.1}$$

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.1. Associative Memory: Accretive Memory*

- In **accretive associative memory**,

  when $\mathbf{u}$ is presented to the memory it responds by producing $\mathbf{y}^r$ of the stored association such that $\mathbf{u}^r$ is the one **closest** to $\mathbf{u}$ among $\mathbf{u}^k$, $k=1..K$, that is,

$$\varphi(\mathbf{u}) = \mathbf{y}^r \quad \textit{such that} \quad \mathbf{u}^r = \min_{\mathbf{u}^k} \left\| \mathbf{u}^k - \mathbf{u} \right\|, \quad k = 1...K \tag{3.1.2}$$

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.1. Associative Memory: Heteroassociative-Autoassociative*

- The accretive associative memory in the form given above, that is $\mathbf{u}^k$ and $\mathbf{y}^k$ are different, is called **heteroassociative memory**.

- However if the stored exemplars are in a special form such that the desired patterns and the input patterns are the same, that is $\mathbf{y}^k=\mathbf{u}^k$ for $k=1..K$, then it is called **autoassociative memory**.

- In autoassociative memory, whenever $\mathbf{u}$ is presented to the memory it responds by $\mathbf{u}^r$ which is the closest one to $\mathbf{u}$ among $\mathbf{u}^k$, $k=1..K$, that is,

$$\varphi(\mathbf{u}) = \mathbf{u}^r \quad \textit{such that} \quad \mathbf{u}^r = \min_{\mathbf{u}^k} \left\| \mathbf{u}^k - \mathbf{u} \right\|, \quad k = 1...K \tag{3.1.3}$$

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.1. Associative Memory*

- While **interpolative memories** can be implemented by using **feed-forward** neural networks, it is more appropriate to use **recurrent** networks as **accretive** memories.

- The advantage of using recurrent networks as associative memory is their convergence to one of a finite number of stable states when started at some initial state. The basic goals are:

  - to be able to store as many exemplars as we need, each corresponding to a different stable state of the network,

  - to have no other stable state

  - to have the stable state that the network converges to be the one closest to the applied pattern .

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.1. Associative Memory*

- The problems that we are faced with being:
  - the capacity of the network is restricted
  - depending on the number and properties of the patterns to be stored, some of the exemplars may not be among the stable states
  - some spurious stable states different than the exemplars may arise by themselves
  - the converged stable state may be other than the one closest to the applied pattern

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.1. Associative Memory*

- One way of using recurrent neural networks as associative memory is to fix the external input of the network and present the input pattern $\mathbf{u}^r$ to the system by setting $\mathbf{x}(0)=\mathbf{u}^r$.
- If we relax such a network, then it will converge to the attractor $\mathbf{x}^*$ for which $\mathbf{x}(0)$ is within the basin attraction as explained in Chapter 2.
- If we are able to place each $\boldsymbol{\mu}^k$ as an attractor of the network by proper choice of the connection weights, then we expect the network to relax to the attractor $\mathbf{x}^*= \boldsymbol{\mu}^r$ that is related to the initial state $\mathbf{x}(0)=\mathbf{u}^r$.
- For a good performance of the network, we need the network only to converge to one of the stored patterns $\boldsymbol{\mu}^k$, $k=1...K$.

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.1. Associative Memory*

- Unfortunately, some initial states may converge to ***spurious states***, which are the undesired attractors of the network representing none of the stored patterns.
- Spurious states may arise by themselves depending on the model used and the patterns stored.
- The capacity of the neural associative memories is restricted by the size of the networks. If we increment the number of stored patterns for a fixed size neural network, spurious states arise inevitably.
- Sometimes, the network may converge not to a spurious state, but to a memory pattern that is not so close to the pattern presented.

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.1. Associative Memory*

- What we expect for a **feasible operation** is that, at least for the memory patterns themselves, if any of the stored pattern is presented to the network by setting $\mathbf{x}(0)=\boldsymbol{\mu}^r$ , then the network should stay converged to $\mathbf{x}^*=\boldsymbol{\mu}^r$ (Figure 3.1).

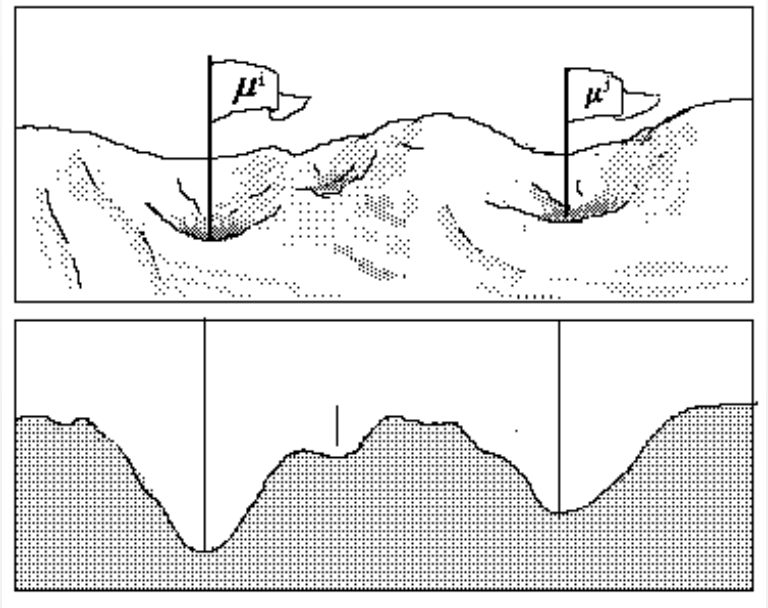



Figure 3.1. In associative memory each memory element is assigned to an attractor

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.1. Associative Memory*

- A second way to use recurrent networks as associative memory, is to present the input pattern $\mathbf{u}^r$ to the system as an external input.
- This can be done by setting $\theta=\mathbf{u}^r$, where $\theta$ is the threshold vector whose $i^{\text{th}}$ component is corresponding to the threshold of neuron $i$.
- After setting $\mathbf{x}(0)$ to some fixed value, we relax the network and then wait until it converges to an attractor $\mathbf{x}^*$.
- For a good performance of the network, we desire the network to have a single attractor such that $\mathbf{x}^*=\boldsymbol{\mu}^k$ for each stored input pattern $\mathbf{u}^k$, therefore the network will converge to this attractor independent of the initial state of the network.
- Another solution to the problem is to have predetermined initial values, so that these initial values lie within the basin attraction of $\boldsymbol{\mu}^k$ whenever $\mathbf{u}^k$ is applied.
- We will consider this kind of networks in Chapter 7 in more detail, where we will examine how these recurrent networks are trained.

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.2. Linear Associators : Orthonormal patterns*

- It is quite easy to implement interpolative associative memory when the set of input memory elements $\{\mathbf{u}^k\}$ constitutes an othonormal set of vectors, that is

$$\mathbf{u}^{i\,\mathrm{T}}\mathbf{u}^{j} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \qquad (3.2.1)$$

where T denotes transpose.

- By using kronecker delta, we write simply

$$\mathbf{u}^{i\,\mathrm{T}}\mathbf{u}^{j} = \delta_{ij} \qquad (3.2.2)$$

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.2. Linear Associators : Orthonormal patterns*

- The mapping function $\varphi(\mathbf{u})$ defined below may be used to establish an interpolative associative memory:

$$\varphi(\mathbf{u}) = \mathbf{W}^{\mathrm{T}}\mathbf{u} \tag{3.2.3}$$

where

$$\mathbf{W} = \sum_{k} \mathbf{u}^{k} \times \mathbf{y}^{k} \tag{3.2.4}$$

- Here the x symbol is used to denote outer product of vectors $\mathbf{x} \in \mathrm{R}^{N}$ and $\mathbf{y} \in \mathrm{R}^{M}$, which is defined as

$$\mathbf{u}^{k} \times \mathbf{y}^{k} = \mathbf{u}^{k}\mathbf{y}^{k^{\mathrm{T}}} = (\mathbf{y}^{k}\mathbf{u}^{k^{\mathrm{T}}})^{\mathrm{T}} \tag{3.2.5}$$

resulting in a matrix of size $N$ by $M$.

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.2. Linear Associators : Orthonormal patterns*

- By defining matrices [Haykin 94]:

$$\mathbf{U}=[\mathbf{u}^1\ \mathbf{u}^2..\ \mathbf{u}^k..\ \mathbf{u}^K] \tag{3.2.6}$$

and

$$\mathbf{Y}=[\mathbf{y}^1\ \mathbf{y}^2..\ \mathbf{y}^k..\ \mathbf{y}^K] \tag{3.2.7}$$

the weight matrix can be formulated as

$$\mathbf{W}^T = \mathbf{Y}\mathbf{U}^T \tag{3.2.8}$$

- If the network is going to be used as autoassociative memory we have $\mathbf{Y}=\mathbf{U}$ so,

$$\mathbf{W}^T = \mathbf{U}\mathbf{U}^T \tag{3.2.9}$$

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.2. Linear Associators : Orthonormal patterns*

- For a function $\varphi(\mathbf{u})$ to constitute an interpolative associative memory, it should satisfy the condition

$$\varphi(\mathbf{u}^r)=\mathbf{y}^r \quad r=1..K \tag{3.2.10}$$

- We can check it simply as

$$\varphi(\mathbf{u}^r) = \mathbf{W}^{\mathsf{T}}\mathbf{u}^r \tag{3.2.11}$$

- which is

$$\mathbf{W}^{\mathsf{T}}\mathbf{u}^r = \mathbf{Y}\mathbf{U}^{\mathsf{T}}\mathbf{u}^r \tag{3.2.12}$$

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.2. Linear Associators : Orthonormal patterns*

- Since the set $\{\mathbf{u}^k\}$ is orthonormal, we have

$$\mathbf{Y}\mathbf{U}^{\mathsf{T}}\mathbf{u}^r = \sum_k \delta_{kr}\mathbf{y}^k = \mathbf{y}^r \qquad (3.2.13)$$

- which results in

$$\varphi(\mathbf{u}^r) = \mathbf{Y}\mathbf{U}^{\mathsf{T}}\mathbf{u}^r = \mathbf{y}^r \qquad (3.2.14)$$

as we desired.

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.2. Linear Associators : Orthonormal patterns*

Remember:

$$\mathbf{W}^{\mathsf{T}}\mathbf{u}^r = \mathbf{Y}\mathbf{U}^{\mathsf{T}}\mathbf{u}^r \qquad (3.2.12)$$

$$\mathbf{Y}\mathbf{U}^{\mathsf{T}}\mathbf{u}^r = \sum_k \delta_{kr}\mathbf{y}^k = \mathbf{y}^r \qquad (3.2.13)$$

- Furthermore, if an input pattern $\mathbf{u}=\mathbf{u}^r+\varepsilon$ different than the stored patterns is applied as input to the network, we obtain

$$\varphi(\mathbf{u}) = \mathbf{W}^{\mathsf{T}}(\mathbf{u}^r + \varepsilon) = \mathbf{W}^{\mathsf{T}}\mathbf{u}^r + \mathbf{W}^{\mathsf{T}}\varepsilon \qquad (3.2.15)$$

- Using equation (3.2.12) and (3.2.13) results in

$$\varphi(\mathbf{u}) = \mathbf{y}^r + \mathbf{W}^{\mathsf{T}}\varepsilon \qquad (3.2.16)$$

- Therefore, we have

$$\varphi(\mathbf{u}) = \mathbf{y}^r + \varepsilon^r \qquad (3.2.17)$$

in the required form, where

$$\varepsilon^r = \mathbf{W}^{\mathsf{T}}\varepsilon \qquad (3.2.18)$$

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.2. Linear Associators : Orthonormal patterns*

- Such a memory can be implemented by using *M* neurons each having *N* inputs as shown in Figure 3.2.

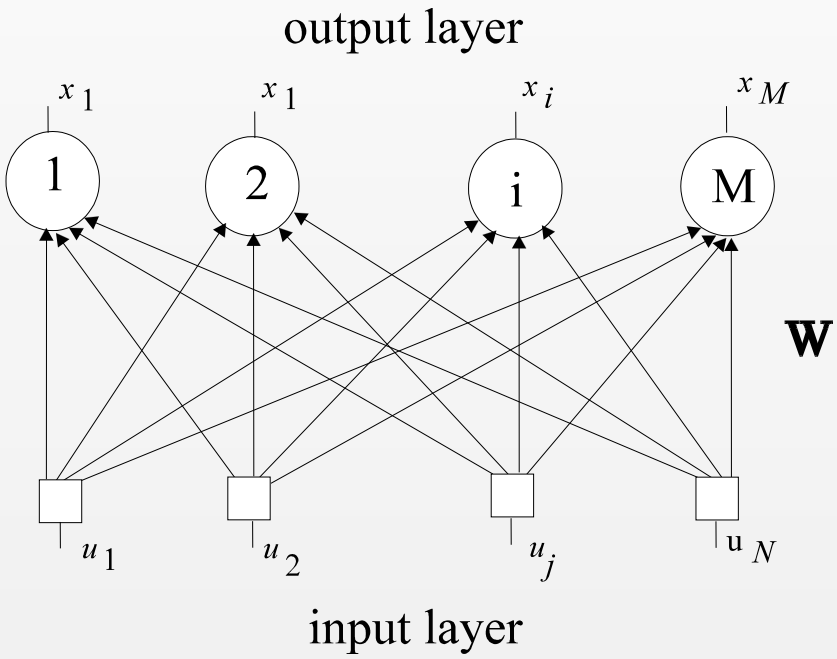
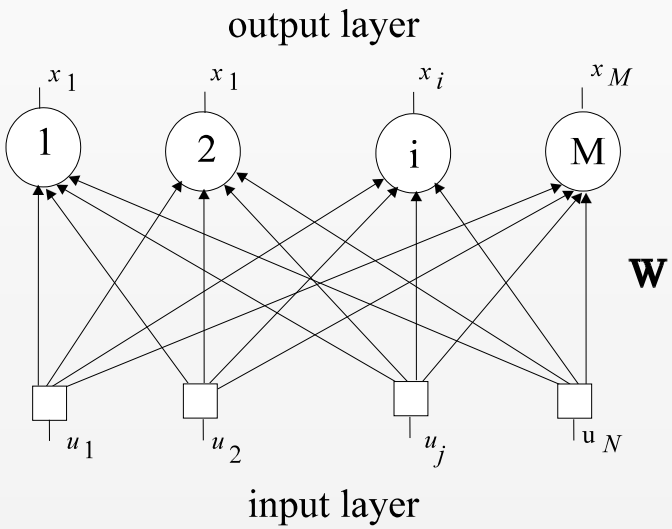



Figure 3.2 Linear Associator

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.2. Linear Associators : Orthonormal patterns*




- The connection weights of neuron *i* is assigned value $\mathbf{W}_i$, which is the $i^{th}$ column vector of matrix $\mathbf{W}$.
- Here each neuron has a linear output transfer function $f(a)=a$.
- When a stored pattern $\mathbf{u}^k$ is applied as input to the network, the desired value $\mathbf{y}^k$ is observed at the output of the network as:

$$\mathbf{x}^k = \mathbf{W}^T \mathbf{u}^k \qquad (3.2.19)$$

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.2. Linear Associators : General case*

- Until now, we have investigated the use of linear mapping $\mathbf{Y}\mathbf{U}^{\mathrm{T}}$ as associative memory, which works well when the input patterns are orthonormal.

- In the case the input patterns are not orthonormal, the linear associator cannot map some input patterns to desired output patterns without error. In the following we will investigate the conditions necessary to minimize the output error for the exemplar patterns.

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.2. Linear Associators : General case*

Remember:

$$\mathbf{U}=[\mathbf{u}^1\ \mathbf{u}^2..\ \mathbf{u}^k..\ \mathbf{u}^K] \qquad (3.2.6)$$

$$\mathbf{Y}=[\mathbf{y}^1\ \mathbf{y}^2..\ \mathbf{y}^k..\ \mathbf{y}^K] \qquad (3.2.7)$$

- Therefore, for a given set of exemplars $\mu^k=(\mathbf{u}^k,\mathbf{y}^k)$, $\mathbf{u}^k\in R^N$, $\in \mathbf{y}^k\in R^M$, $k=1..\ K$, our purpose is to find a linear mapping $\mathbf{A}^*$ among $\mathbf{A}$: $R^N \rightarrow R^M$ such that:

$$\mathbf{A}^* = \min_{\mathbf{A}} \sum_k \left\| \mathbf{y}^k - \mathbf{A}\mathbf{u}^k \right\| \qquad (3.2.20)$$

where $\|\cdot\|$ is chosen as Euclidean norm.

- The problem may be reformulated by using the matrices $\mathbf{U}$ and $\mathbf{Y}$ [Haykin 94]:

$$\mathbf{A}^* = \min_{\mathbf{A}} \left\| \mathbf{Y} - \mathbf{A}\mathbf{U} \right\| \qquad (3.2.21)$$

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.2. Linear Associators : General case*

- The pseudo inverse method [Kohonen 76] based on least squares estimation provides a solution for the problem in which $\mathbf{A}^*$ is determined as:

$$\mathbf{A}^* = \mathbf{Y}\mathbf{U}^+ \tag{3.2.22}$$

where $\mathbf{U}^+$ is pseudo inverse of $\mathbf{U}$.

- The pseudoinverse $\mathrm{U}^+$ is a matrix satisfying the condition:

$$\mathbf{U}^+\mathbf{U} = \mathbf{1} \tag{3.2.23}$$

where $\mathbf{1}$ is the identity matrix.

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.2. Linear Associators : General case*

- A perfect match is obtained by using

$$\mathbf{A}^* = \mathbf{Y}\mathbf{U}^+$$

since

$$\mathbf{A}^*\mathbf{U} = \mathbf{Y}\mathbf{U}^+\mathbf{U} = \mathbf{Y} \qquad (3.2.24)$$

resulting in no error due to the fact

$$\mathbf{Y} - \mathbf{A}^*\mathbf{U} = \mathbf{0} \qquad (3.2.25)$$

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.2. Linear Associators : Linearly Independent Patterns*

Remember

$$\mathbf{U}^{+}\mathbf{U} = \mathbf{1} \tag{3.2.23}$$

---

- In the case the input patterns are linearly independent, that is none of them can be obtained as a linear combinations of the others, then a matrix $\mathbf{U}^{+}$ satisfying Eq. (3.2.23) can be obtained by applying the formula [Golub and Van Loan 89, Haykin 94]

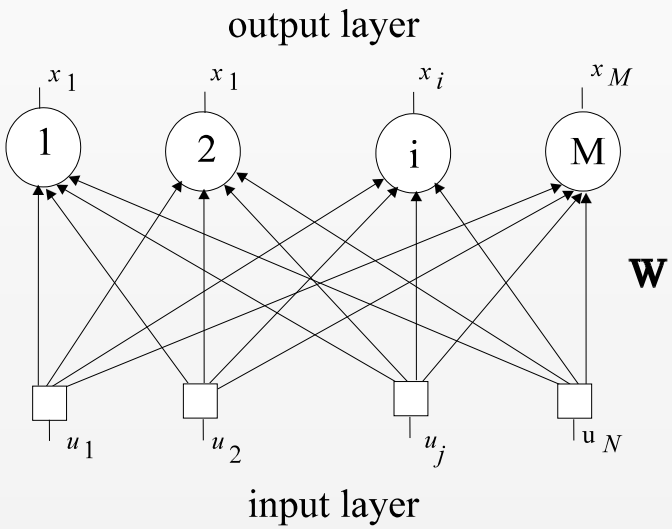
$$\mathbf{U}^{+} = (\mathbf{U}^{\mathsf{T}}\mathbf{U})^{-1}\mathbf{U}^{\mathsf{T}} \tag{3.2.26}$$

- Notice that for the input patterns, which are the columns of the matrix $\mathbf{U}$, to be linearly independent, the number of columns should not be more than the number of rows, that is $K \leq N$, otherwise $\mathbf{U}^{\mathrm{T}}\mathbf{U}$ will be singular and no inverse will exist.

- The condition $K \leq N$ means that the number of entries constituting the patterns restricts the capacity of the memory. At most $N$ patterns can be stored in such a memory.

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.2. Linear Associators : Linearly Independent Patterns*




- This memory can be implemented by a neural network for which $\mathbf{W}^{T}=\mathbf{Y}\mathbf{U}^{+}$.
- The desired value $\mathbf{y}^k$ appears at the output of the network as $\mathbf{x}^k$ when $\mathbf{u}^k$ is applied as input to the network:

$$\mathbf{x}^k = \mathbf{W}^{T}\mathbf{u}^k \qquad (3.2.27)$$

as explained previously.

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.2. Linear Associators : Linearly Independent Patterns*

Remember:

$$\mathbf{U}^{+} = (\mathbf{U}^{T}\mathbf{U})^{-1}\mathbf{U}^{T} \tag{3.2.26}$$

- Notice that for the special case of orthonormal patterns that we examined previously in this section, we have

$$\mathbf{U}^{T}\mathbf{U} = \mathbf{1} \tag{3.2.28}$$

that results in the pseudoinverse, which is in the form

$$\mathbf{U}^{+} = \mathbf{U}^{T} \tag{3.2.29}$$

and therefore

$$\mathbf{W}^{T} = \mathbf{Y}\mathbf{U}^{+} = \mathbf{Y}\mathbf{U}^{T} \tag{3.2.30}$$

as we have derived previously.

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.3. Hopfield Autoassociative Memory*

- In this section we will investigate how Hopfield network can be used as auto-associative memory. For this purpose some modifications are done on continuous Hopfield network so that it works in **discrete state space** and **discrete time**.

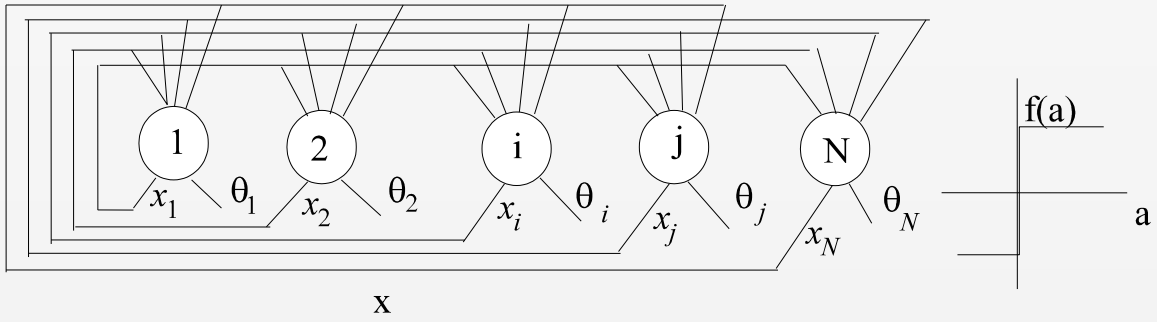



Figure 3.3 Hopfield Associative Memory

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.3. Hopfield Autoassociative Memory*

- Note that whenever the patterns to be stored in Hopfield network are from *N* dimensional bipolar space constituting a hypercube, that is $\mathbf{u}^k \in \{-1,1\}^N$, $k=1..K$, then it is convenient to have any stable state of the network on the corners of the hypercube.

- If we let the output transfer function of the neurons in the network to have very high gain, in the extreme case

$$f_i(a) = \lim_{\kappa \to \infty} \tanh(\kappa a) \tag{3.3.1}$$

we obtain

$$f_i(a) = \text{sign}(a) = \begin{cases} 1 & for \quad a > 0 \\ 0 & for \quad a = 0 \\ -1 & for \quad a < 0 \end{cases} \tag{3.3.2}$$

## *3.3. Hopfield Autoassociative Memory*

- Furthermore note that the second term of the energy function (which was given in Chapter 2 previously)

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ji}x_j x_i + \sum_{i=1}^{N}\frac{1}{R_i}\int_0^{x_i} f^{-1}(x)dx - \sum_{i=1}^{N}\theta_i x_i \qquad (3.3.3)$$

approaches to zero.

- Therefore the stable states of the network corresponds to the local minima of the function:

$$E = -\frac{1}{2}\sum_{i}\sum_{j} w_{ji}x_j x_i - \sum_{i}\theta_i x_i \qquad (3.3.4)$$

so that they lie on the corners of the hypercube as explained previously.

## *3.3. Hopfield Autoassociative Memory*

- Discrete time state excitation [Hopfield 82] of the network, is provided in the following:

$$x_i(k+1) = f(a_i(k)) = \begin{cases} 1 & a_i(k) > 0 \\ x(k) & a_i(k) = 0 \\ -1 & a_i(k) < 0 \end{cases} \qquad (3.3.5)$$

where $a_i(k)$ is defined as we used to, that is,

$$a_i(k) = \sum_j w_{ji} x_j(k) + \theta_i \qquad (3.3.6)$$

- The processing elements of the network are updated one at a time, such that all of the processing elements must be updated at the same average rate (called asynchronous update or sequential update)

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.3. Hopfield Autoassociative Memory*

Remember:

$$\mathbf{U}=[\mathbf{u}^1\ \mathbf{u}^2..\ \mathbf{u}^k..\ \mathbf{u}^K] \quad (3.2.6)$$

$$\mathbf{Y}=[\mathbf{y}^1\ \mathbf{y}^2..\ \mathbf{y}^k..\ \mathbf{y}^K] \quad (3.2.7)$$

- For stability of the bipolar discrete Hopfield network, it is further required to have $w_{ii}=0$ in addition to the constraint $w_{ij}=w_{ji}$
- In order to use discrete Hopfield network as auto-associative memory, first its weights are fixed to

$$\mathbf{W}^T = \mathbf{U}\mathbf{U}^T \quad (3.3.8)$$

  where $\mathbf{U}$ is the input pattern matrix as defined in Eq. (3.2.6), and then $w_{ii}$ are set to 0

- Remember that in auto-associative memory we have $\mathbf{Y}=\mathbf{U}$, where $\mathbf{Y}$ is the matrix of desired output patterns as defined in Eq (3.2.7).

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.3. Hopfield Autoassociative Memory*

- If all the states of the network are to be updated at once (called synchronous update or parallel update), then the next state of the system may be represented in the form

$$\mathbf{x}(k+1)=\mathbf{f}(\mathbf{W}^{\mathrm{T}}\mathbf{x}(k)) \qquad (3.3.9)$$

- For the special case if the exemplars are orthonormal, we have

$$\mathbf{f}(\mathbf{W}^{\mathrm{T}}\mathbf{u}^{\mathrm{r}})=\mathbf{f}(\mathbf{u}^{\mathrm{r}})=\mathbf{u}^{\mathrm{r}} \qquad (3.3.10)$$

that means each exemplar is a stable state of the network

.

  - Whenever the initial state is set to one of the exemplar, the system remains there.
  - However, if the initial state is set to some arbitrary input, then the network converges to one of the stored exemplars, depending on the basin of attraction in which $\mathbf{x}(0)$ lies.

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.3. Hopfield Autoassociative Memory*

- However in general the input patterns are not orthonormal, so there is no guarantee that each exemplar is corresponding to a stable state. Therefore the problems that we mentioned in Section 3.1 arise.

- The capacity of the Hopfield net is less than 0.138*$N$ patterns, where $N$ is the number of units in the network [Lippmann 89].

- It is shown in the lecture notes that the energy function always decreases as the state of the processing elements are changed one by one (asynchronous update).

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.4. Bi-directional Associative Memory*

- The Bi-directional Associative Memory (BAM) introduced in [Kosko 88] is a recurrent network (Figure 3.4) designed to work as hetero-associative memory [Nielsen 90].

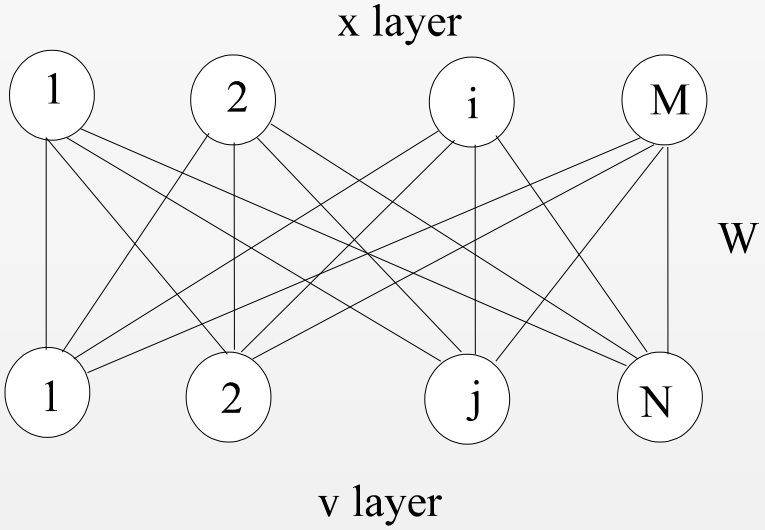
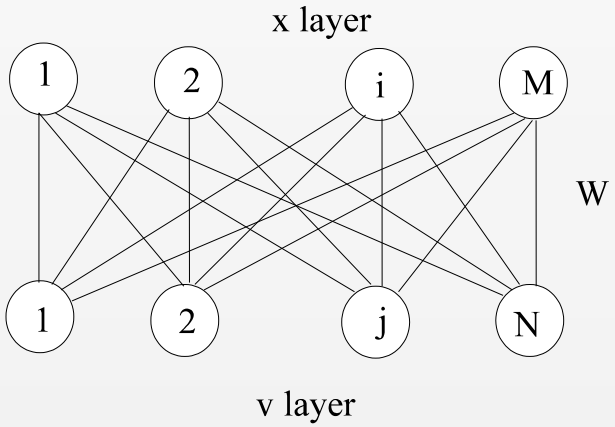



Figure 3.4: Bi-directional Associative Memory

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.4. Bi-directional Associative Memory*

- BAM network consists of two sets of neurons whose outputs are represented by vectors $\mathbf{x} \in R^N$ and $\mathbf{v} \in R^M$ respectively, having activation defined by the pair of equations:




$$\frac{da_{x_i}}{dt} = -\alpha_i a_{x_i} + \sum_{j=1}^{N} w_{ji} f(a_{v_j}) + \theta_i \; i = 1..M \qquad (3.4.1)$$

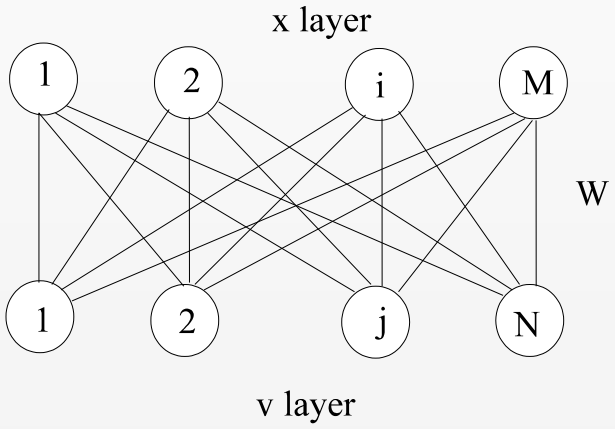
$$\frac{da_{v_j}}{dt} = -\beta_j a_{y_j} + \sum_{i=1}^{M} w_{ij} f(a_{x_i}) + \phi_j \; j = 1..N \qquad (3.4.2)$$

where

$\alpha_i, \beta_j, \theta_i, \phi_j$ are positive constants for $i=1..M, j=1..N$,
$f$ is tanh function and
$\mathbf{W}=[w_{ij}]$ is any $NxM$ real matrix.

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.4. Bi-directional Associative Memory*



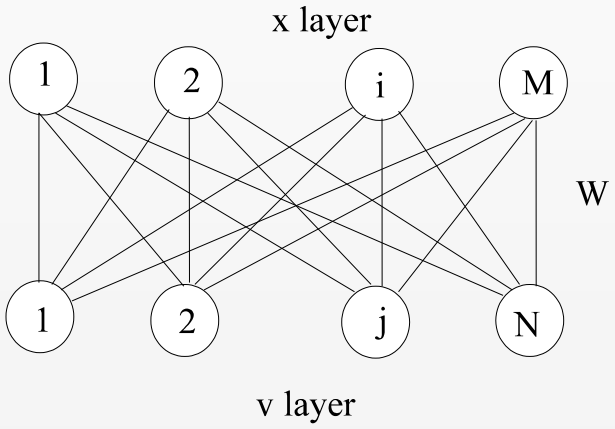

- The stability of the BAM network can be proved easily by applying Cohen-Grossberg theorem by defining a state vector $\mathbf{z} \in R^{N+M}$, such that

$$z_i = \begin{cases} x_i & i \le M \\ v_j & j = i - M, \quad M < i \le M + N \end{cases} \qquad (3.4.3)$$

that is $\mathbf{z}$ obtained through concatenation $\mathbf{x}$ and $\mathbf{v}$.

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.4. Bi-directional Associative Memory*




- Since BAM is a special case of the network defined by Cohen-Grossberg theorem, it has a Lyapunov Energy function as it is provided in the following:

$$E(\mathbf{x},\mathbf{v}) = -\sum_{i=1}^{M}\sum_{j=1}^{N} w_{ij} f(a_{x_i}) f(a_{v_j})$$

$$+ \sum_{i=1}^{M} \alpha_i \int_0^{a_{x_i}} f'(a)a\,da + \sum_{j=1}^{N} \beta_j \int_0^{a_{vj}} f'(b)b db$$

$$- \sum_{i=1}^{M} f(x_i)\theta_i - \sum_{j=1}^{N} f(v_j)\phi_j$$

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.4. Bi-directional Associative Memory*

- The discrete BAM model is defined in a similar manner to discrete Hopfield network.
- The output functions are chosen to be $f(a)=\text{sign}(a)$ and states are excited as:

$$x_i(k+1) = f(a_{x_i}(k)) = \begin{cases} 1 & for\ a_{x_i}(k) > 0 \\ x(k) & for\ a_{x_i}(k) = 0 \\ -1 & for\ a_{x_i}(k) < 0 \end{cases}$$

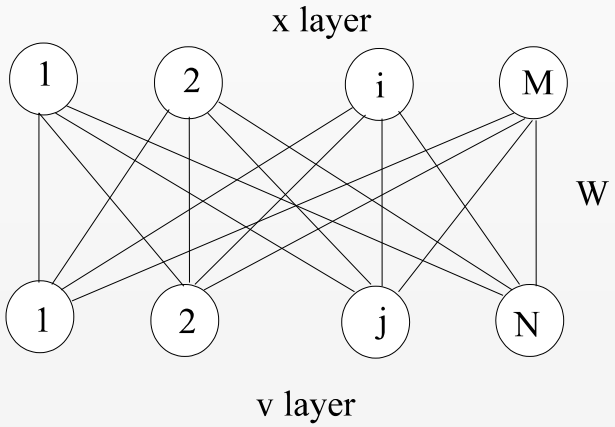
where $a_{x_i} = \sum_{j=1}^{m} w_{ji} f(a_{v_j}) + \theta_i \quad i = 1..M$

$$v_j(k+1) = f(a_{v_j}(k)) = \begin{cases} 1 & for\ a_{v_j}(k) > 0 \\ v(k) & for\ a_{v_j}(k) = 0 \\ -1 & for\ a_{v_j}(k) < 0 \end{cases}$$

where $a_{v_j} = \sum_{i=1}^{n} w_{ij} f(a_{x_i}) + \phi_j \quad j = 1..N$

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.4. Bi-directional Associative Memory*




- In compact matrix notation it is shortly

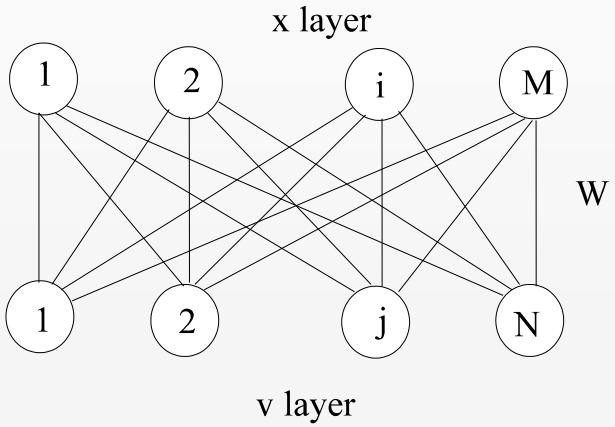
$$\mathbf{x}(k+1)=\mathbf{f}\,(\mathbf{W}^{\mathrm{T}}\mathbf{v}(k)) \qquad (3.4.9)$$

and

$$\mathbf{v}(k+1)=\mathbf{f}(\mathbf{W}\mathbf{x}(k+1)). \qquad (3.4.10)$$

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.4. Bi-directional Associative Memory*




- In the discrete BAM, the energy function becomes

$$E(\mathbf{x},\mathbf{y}) = -\sum_{i=1}^{M}\sum_{j=1}^{N} w_{ij} f(a_{x_i}) f(a_{v_j}) - \sum_{i=1}^{M} f(a_{x_i})\theta_i - \sum_{j=1}^{N} f(a_{v_j})\phi_j \qquad (3.4.11)$$

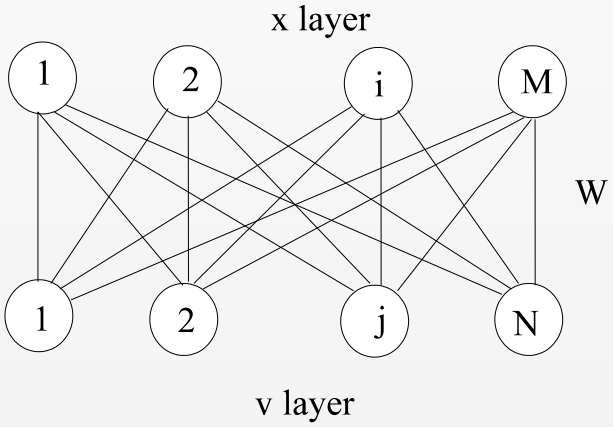
satisfying the condition

$$\Delta E \le 0 \qquad (3.4.12)$$

which implies the stability of the system.

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.4. Bi-directional Associative Memory*




- The weights of BAM is determined by the equation

$$\mathbf{W}^T=\mathbf{Y}\mathbf{U}^T \qquad (3.4.13)$$

- For the special case of orthonormal input and output patterns we have

$$\mathbf{f}(\mathbf{W}^T\mathbf{u}^r)= \mathbf{f}(\mathbf{Y}\mathbf{U}^T\mathbf{u}^r)=\mathbf{f}(\mathbf{y}^r)=\mathbf{y}^r \qquad (3.4.14)$$

and

$$\mathbf{f}(\mathbf{W}\mathbf{y}^r)= \mathbf{f}(\mathbf{U}\mathbf{Y}^T\mathbf{y}^r)=\mathbf{f}(\mathbf{u}^r)=\mathbf{u}^r \qquad (3.4.15)$$

indicating that exemplar are stable states of the network.

# CHAPTER III : *Neural Networks as Associative Memory*

## *3.4. Bi-directional Associative Memory*

- Whenever the initial state is set to one of the exemplar, the system remains there. For arbitrary initial states the network converges to one of the stored exemplars, depending on the basin of attraction in which $\mathbf{x}(0)$ lies.

- For the input patterns that are not orthonormal, the network behaves as it is explained for the Hopfield network.