SOFT COMPUTING

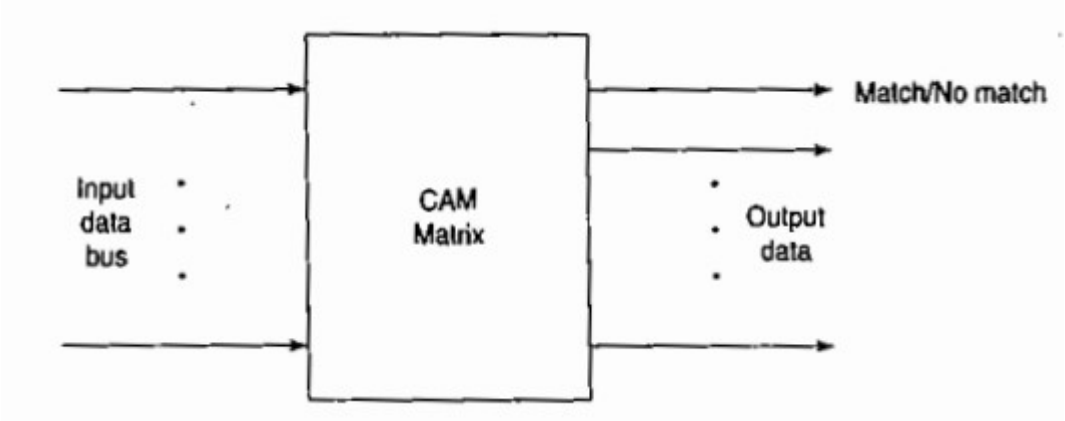
Associative Memory

Associative Memory

- An associative memory network can store a set of patterns as memories.
- When the associative memory is being presented with a key pattern, it responds by producing one of the scored patterns, which closely resembles or relates to the key pattern.
- Associative memory makes a parallel search within a stored data file.
- The concept behind this search is to output any one or all stored items which match the given search argument and to retrieve the stored data either completely or partially.

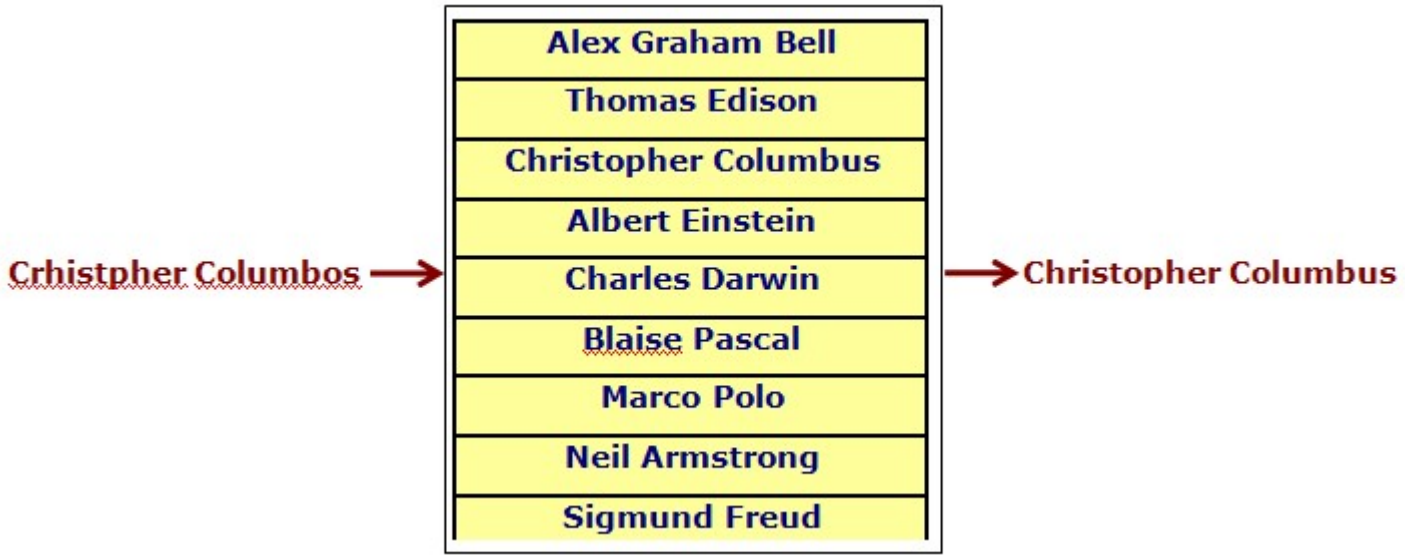
Associative Memory

- 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.



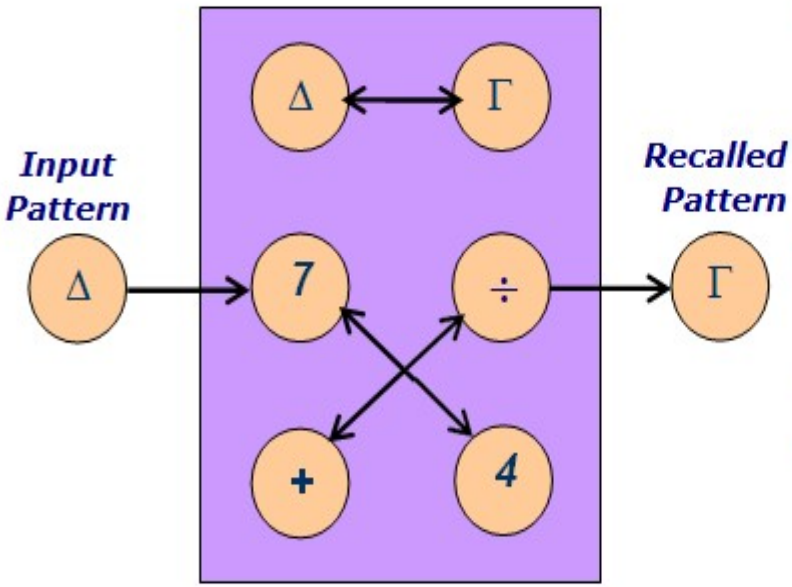
Associative memory

- 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



Associative memory

- If the given memory is content-addressable, then using the erroneous string "Crhistpher Columbos" as key is sufficient to retrieve the correct name "Christopher Columbus."
- This type of memory is robust and fault-tolerant, because this type of memory exhibits some form of error-correction capability.
- The memory allows the recall of information based on partial knowledge of its contents.



The associated pattern pairs

(Δ, Γ), (÷, +), (7, 4).

The association is represented by the symbol ↔

The associated pattern pairs are stored the memory.

Types of associative memory

auto-associative

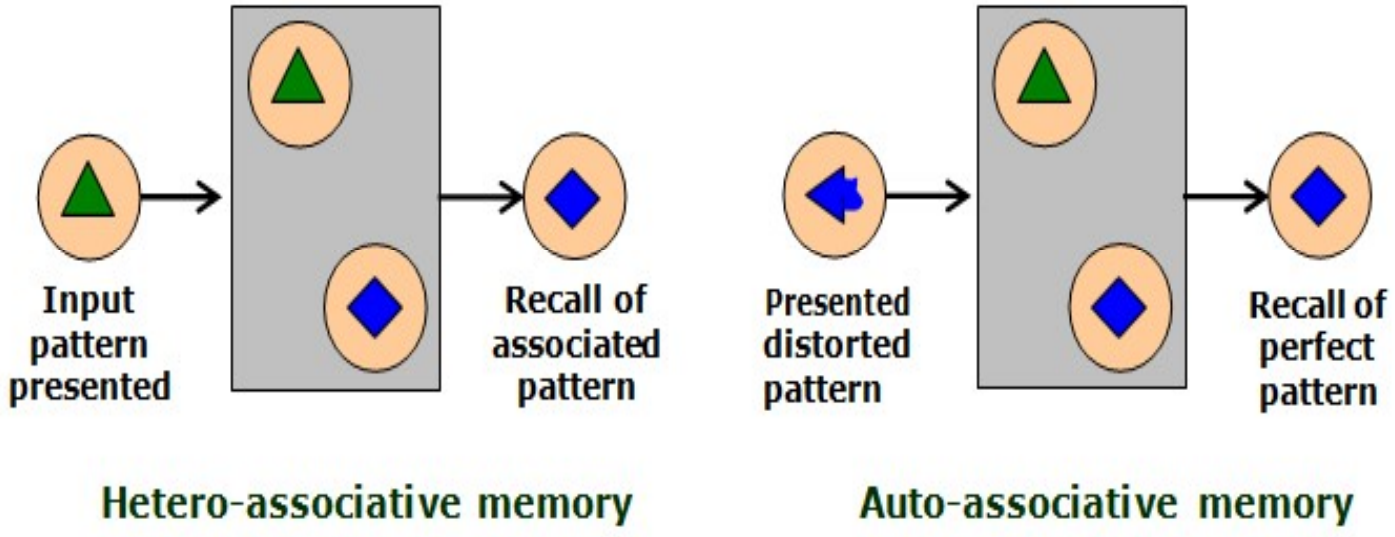
hetero-associative

- **An auto-associative** memory retrieves previously stored pattern that most closely resembles the current pattern.
- Each of this association is an input-output vector pairs.
- In this each of the output vectors is same as the input vectors with which it is associated.
- 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.

Types of associative memory

auto-associative

hetero-associative



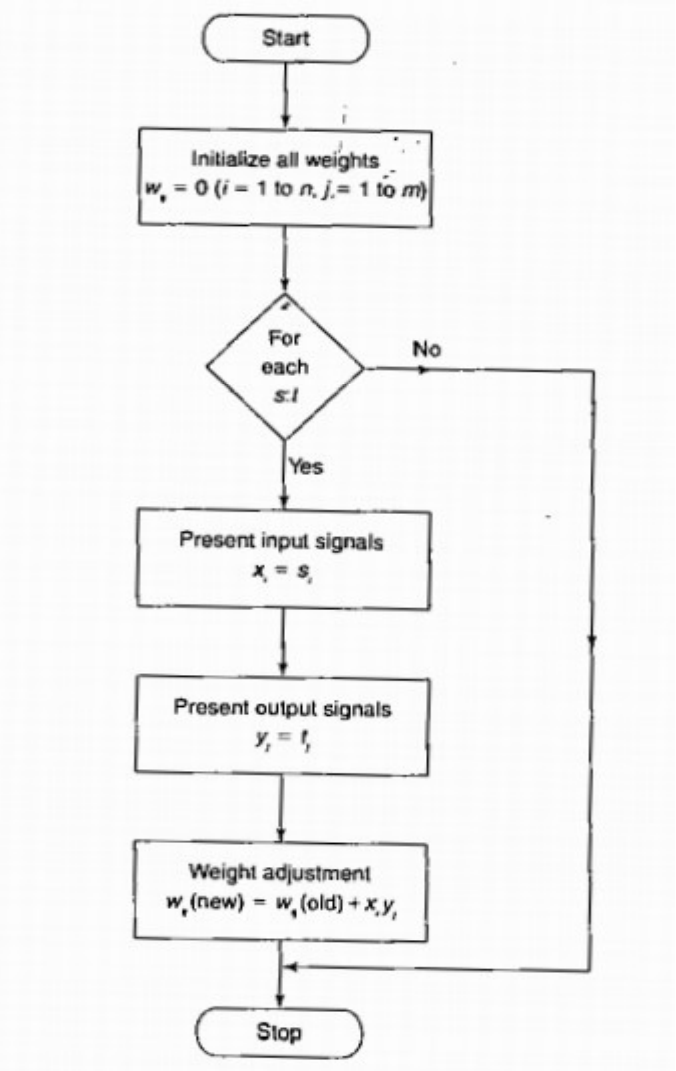
Training Algorithms for Pattern Association

1. Hebb Rule

The weights are updated

until there is no weight change.

$$w_{ij}(\text{new}) = w_{ij}(\text{old}) + x_i y_j \quad (\text{for } i = 1 \text{ to } n, j = 1 \text{ to } m)$$



Training Algorithms for Pattern Association

2. Outer Products Rule

Input=> s = (s1, ... ,sj, ... ,sn)

Output=> t= (t1,⋯,tj,⋯,tm)

The weight matrix W = {wij} can be given

$$w_{ij} = \sum_{p=1}^{P} s_i^T(p)\, t_j(p)$$

$$ST = s^T t = \begin{bmatrix} s_1 \\ \vdots \\ s_i \\ \vdots \\ s_n \end{bmatrix}_{n\times 1} [t_1 \dots t_j \dots t_m]_{1\times m}$$

$$W = \begin{bmatrix} s_1t_1 & \dots & s_1t_j & \dots & s_1t_m \\ \vdots & & \vdots & & \vdots \\ s_it_1 & \dots & s_it_j & \dots & s_it_m \\ \vdots & & \vdots & & \vdots \\ s_nt_1 & \dots & s_nt_j & \dots & s_nt_m \end{bmatrix}_{n\times m}$$

Associative memory models

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)**

These models follow different neural network architectures to memorize information.

Associative memory models

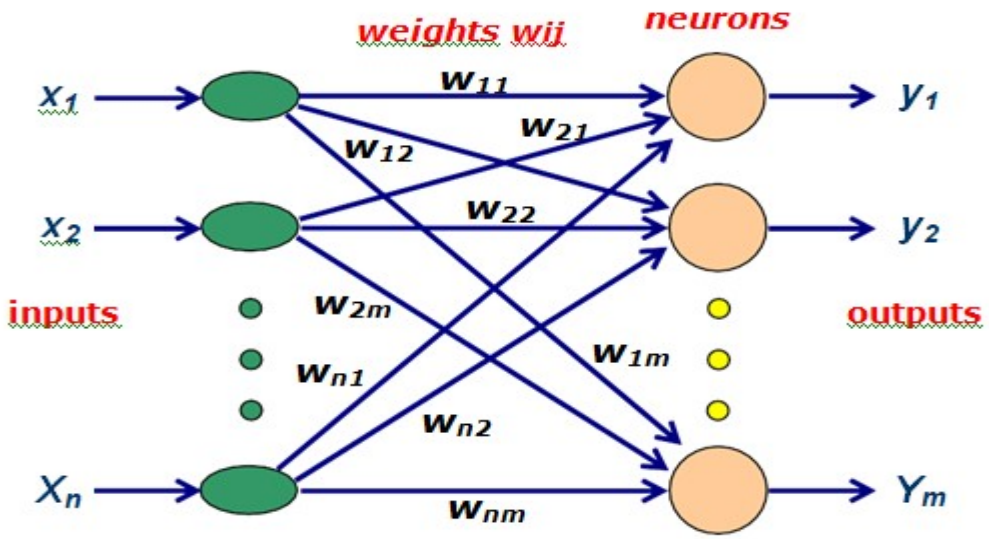
The **Linear associator model,** is a feed forward type network, consists, two layers of processing units, one serving as the input layer while the other as the output layer.

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.

Linear Associator Model

- All **n** input units are connected to all **m** output units via connection weight matrix $\mathbf{W} = [w_{ij}]_{n \times m}$
- The connection weight matrix stores the **p** different associated pattern pairs $\{(X_k, Y_k) \mid k = 1, 2, \ldots, p\}$.



Linear Associator Model

- An associative memory constructs the connection weight matrix **W** such that when an input pattern is presented, then the stored pattern associated with the input pattern is retrieved
- **Encoding** : During encoding the weight values of correlation matrix **Wk** for an associated pattern pair (X_k, Y_k) are computed as:

$$(w_{ij})_k = (x_i)_k (yj)_k$$

- **Weight matrix** : Construction of weight matrix **W** is accomplished ***p*** by summing those individual correlation matrices W_k, ie, $W = \alpha$

  α is the constant of proportionality, for normalizing, usually set to **1/p** to store **p** different associated pattern pairs.

Linear Associator Model

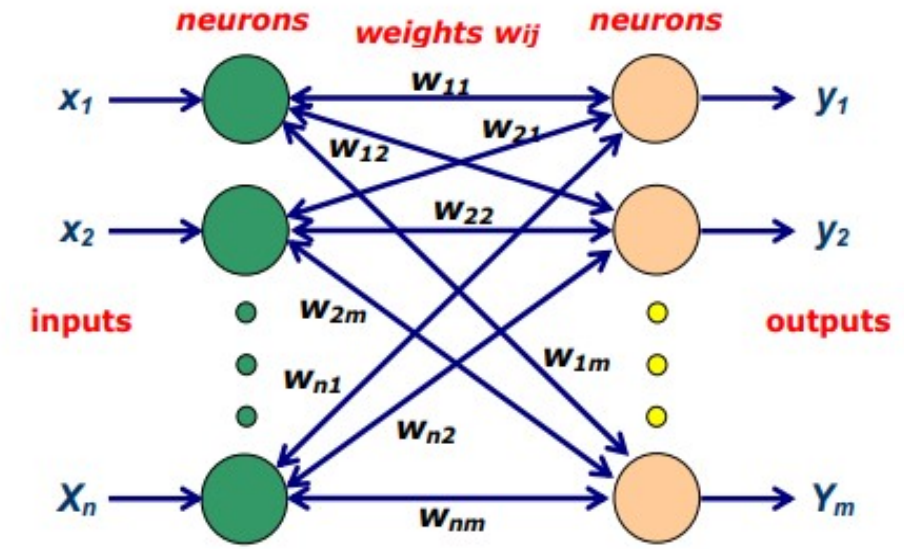
- **Decoding** : given an input pattern **X**, the decoding or retrieving is accomplished ***m*** by computing, first the net *Input* as $\text{input}_j = \sum_{j=1}^{m} x_i \, w_{ij}$

  input $_i$ stands for the weighted sum of the input or activation value of node **j** , for **j = 1, 2, . . , n.** and $\mathbf{x_i}$ is the **i th** component of pattern $\mathbf{X_k}$, and then determine the units *output* using a bipolar output function:

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

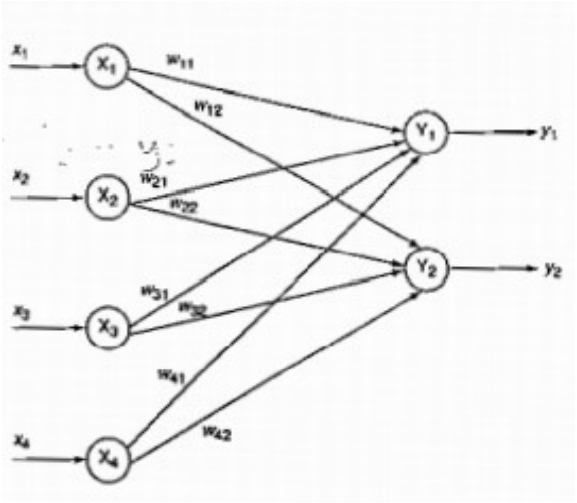
where θ_j is the threshold value of output neuron **j** .

Bidirectional Associative Memory (BAM)



Hetero Associative Memory

s1	s2	s3	s4	t1	t2
1	0	1	0	1	0
1	0	0	1	1	0
1	1	0	0	0	1
0	0	1	1	0	1



1. Apply Hebb rule to determine the weights in matrix form?
2. Apply Outer Product rule to determine the weights in matrix form?

1. Hebb Rule

Step 0: Initialize the weights, the initial weights are taken as zero.

Step 1: For first pair (1, 0, 1, 0):(1, 0)

Step 2: Set the activations of input units:

$$x_1 = 1, \quad x_2 = 0, \quad x_3 = 1, \quad x_4 = 0$$

Step 3: Set the activations of output unit:

$$y_1 = 1, \quad y_2 = 0$$

$$W = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 0 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}$$

$$w_{11}(\text{new}) = w_{11}(\text{old}) + x_1 y_1 = 0 + 1 \times 1 = 1$$
$$w_{21}(\text{new}) = w_{21}(\text{old}) + x_2 y_1 = 0 + 0 \times 1 = 0$$
$$w_{31}(\text{new}) = w_{31}(\text{old}) + x_3 y_1 = 0 + 1 \times 1 = 1$$
$$w_{41}(\text{new}) = w_{41}(\text{old}) + x_4 y_1 = 0 + 0 \times 1 = 0$$
$$w_{12}(\text{new}) = w_{12}(\text{old}) + x_1 y_2 = 0 + 1 \times 0 = 0$$
$$w_{22}(\text{new}) = w_{22}(\text{old}) + x_2 y_2 = 0 + 0 \times 0 = 0$$
$$w_{32}(\text{new}) = w_{32}(\text{old}) + x_3 y_2 = 0 + 1 \times 0 = 0$$
$$w_{42}(\text{new}) = w_{42}(\text{old}) + x_4 y_2 = 0 + 0 \times 0 = 0$$

2. Outer Product Rule

$$W = \sum_{p=1}^{P} s^T(p)\, t(p)$$

$(1\ 0\ 1\ 0)$, $t = (1\ 0)$. For $p = 1$,

$$s^T(p)\, t(p) = s^T(1)\, t(1)$$

$$= \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}_{4\times 1} \begin{bmatrix} 1 & 0 \end{bmatrix}_{1\times 2} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}_{4\times 2}$$

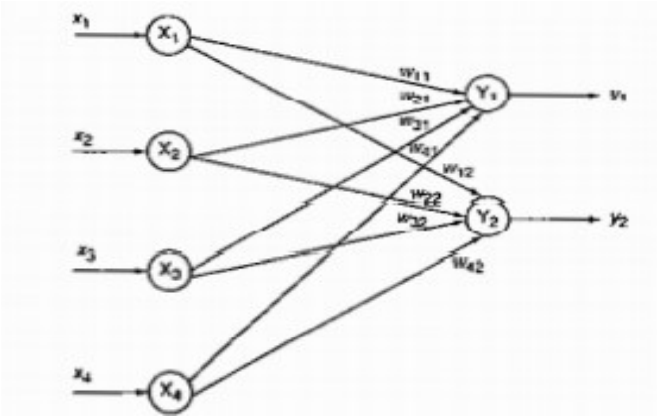
$$W = \sum_{p=1}^{4} s^T(p)\, t(p)$$

$$= s^T(1)t(1) + s^T(2)t(2) + s^T(3)t(3) + s^T(4)t(4)$$

$$= \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

$$W = \begin{bmatrix} 2 & 1 \\ 0 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}$$

s1	s2	s3	s4	t1	t2
1	0	0	0	0	1
1	1	0	0	0	1
0	0	0	1	1	0
0	0	1	1	1	0

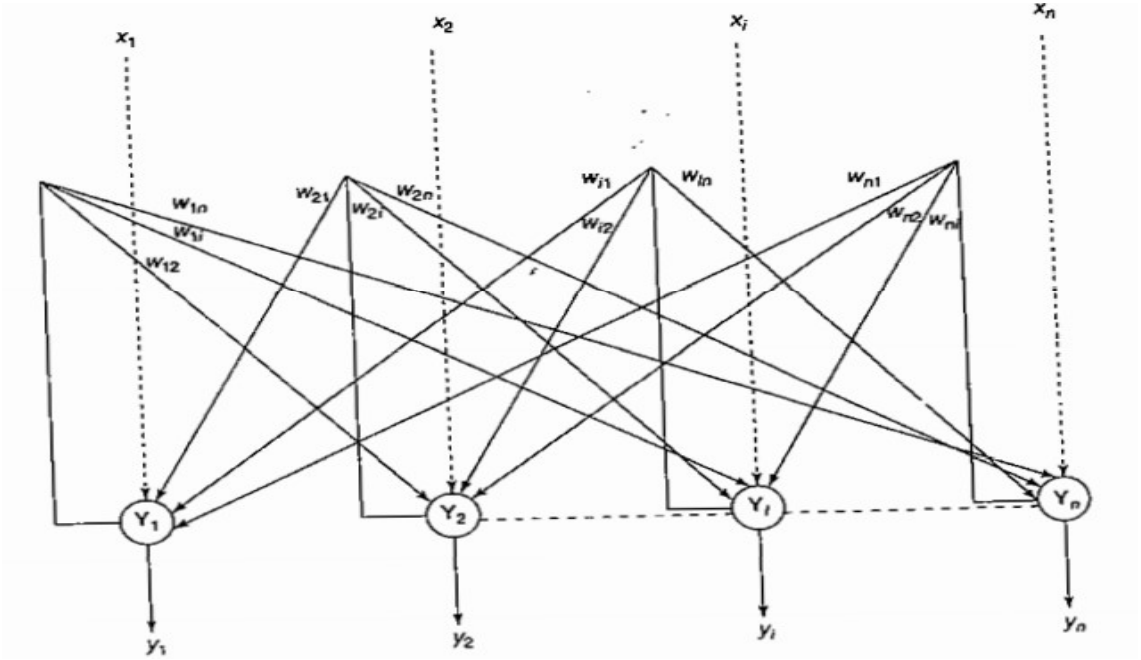


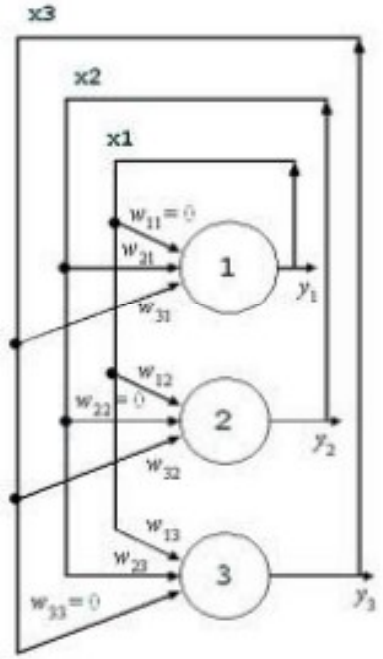
Test the performance of the network using its training input as testing input.

Hopfield Networks

- A Hopfield network is a simple assembly of perceptrons that is able to overcome the XOR problem.
- The array of neurons is fully connected, although neurons do not have self-loops.
- The Hopfield's model consist of processing elements with two outputs, one inverting and the otherr non-inverting.
- It consists of a single layer which contains one or more fully connected recurrent neurons.
- A Hopfield network is single-layered and recurrent network: the neurons are fully connected, i.e., every neuron is connected to every other neuron.
- Given two neurons i and j there is a connectivity weight wij between them which is symmetric wij = wji with zero self-connectivity wii = 0.

Hopfield Networks





Discrete Hopfield Networks

- The outputs from each processing element are fed back to the input of other processing elements but not to itself.
- For storing a set of binary patterns s(p), p = I to P, where s(p) (s1 (p), ... , sj(p), ... , sn(p)), the weight matrix W is given as

$$w_{ij} = \sum_{p=1}^{P} [2s_i(p) - 1][2s_j(p) - 1], \quad \text{for } i \neq j$$

- For storing a set of bipolar input patterns, s(p) the weight matrix W is given as

$$w_{ij} = \sum_{p=1}^{P} s_i(p)s_j(p), \quad \text{for } i \neq j$$

Testing

Step 0: *Initialize the weights to store patterns, i.e., weights obtained from training algorithm using Hebb rule.*

Step 1: When the activations of the net are not converged, then perform Steps 2–8.

Step 2: Perform Steps 3–7 for each input vector X.

Step 3: Make the initial activations of the net equal to the external input vector X:

$$y_i = x_i \ (i = 1 \text{ to } n)$$

Step 4: Perform Steps 5–7 for each unit Y_i. (Here, the units are updated in random order.)

Step 5: Calculate the net input of the network:

$$y_{in_i} = x_i + \sum_j y_j w_{ji}$$

Step 6: Apply the activations over the net input to calculate the output:

$$y_i = \begin{cases} 1 & \text{if } y_{in_i} > \theta_i \\ y_i & \text{if } y_{in_i} = \theta_i \\ 0 & \text{if } y_{in_i} < \theta_i \end{cases}$$

where θ_i is the threshold and is normally taken as zero.

Step 7: Now feed back (transmit) the obtained output y_i to all other units. Thus, the activation vectors are updated.

Step 8: Finally, test the network for convergence.

Analysis of Energy Function and Storage Capacity on Discrete Hopfield Net

- The energy function, also called as Lyapunov function
- It determines the stability property of a discrete Hopfield network.
- The state of a System for a neural network is the vector of activations of the units. Hence, if it is possible to find an energy function for an iterative neural net, the net will converge to a stable set of activations

$$E_f = -\frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1 \\ j\neq i}}^{n} y_i\, y_j\, w_{ij} - \sum_{i=1}^{n} x_i\, y_i + \sum_{i=1}^{n} \theta_i\, y_i$$

$$\Delta E_f = E_f\left(y_j^{(k+1)}\right) - E_f\left(y_i^{(k)}\right)$$
$$= -\left(\sum_{\substack{j=1 \\ j\neq i}}^{n} y_j^{(k)} w_{ij} + x_i - \theta_i\right)\left(y_i^{(k+1)} - y_i^{(k)}\right)$$

There exist two cases in which a change Δy_i will occur in the activation of neuron Y_i. If y_i is positive, then it will change to zero if

$$\left[x_i + \sum_{j=1}^{n} y_j w_{ji}\right] < \theta_i$$

This results in a negative change for y_i and $\Delta E_f < 0$. On the other hand, if y_i is zero, then it will change to positive if

$$\left[x_i + \sum_{j=1}^{n} y_j w_{ji}\right] > \theta_i$$

Hence Δy_i is positive only if net input is positive and Δy_i is negative only if net input is negative.