Hebb Rule Algorithmic

Step 0: Set all the initial weights to zero, i.e.,
$W_{ij} = 0$ (i = 1 to n, j = 1 to m)

Step 1: For each training target input output vector pairs s:t, perform Steps 2-4.

Step 2: Activate the input layer units to current training input, $X_i=S_i$ (for i = 1 to n)

Step 3: Activate the output layer units to current target output,
$y_j = t_j$ (for j = 1 to m)

Step 4: Start the weight adjustment

$$w_{ij}(new) = w_{ij}(old) + x_i y_j (i = 1 to n j = 1 to m)$$

2. Outer Products Rule: Outer products rule is a method for finding weights of an associative net.

Input=> $s = (s_1, \ldots, s_i, \ldots, s_n)$
Output=> $t = (t_1, \ldots, t_j, \ldots, t_m)$

The outer product of the two vectors is the product of the matrices $S = s^T$ and T = t, i.e., between [n X 1] marrix and [1 x m] matrix. The transpose is to be taken for the input matrix given.

$ST = s^T t$ =>

$$\begin{bmatrix} s_1 \\ \vdots \\ s_i \\ \vdots \\ s_n \end{bmatrix} \cdot \begin{bmatrix} t_1 & \cdots & t_j & \cdots & t_m \end{bmatrix}$$

This weight matrix is same as the weight matrix obtained by Hebb rule to store the pattern association s:t. For storing a set of associations, s(p):t(p), p = 1 to P, wherein,
$s(p) = (s_1(p), \ldots, s_i(p), \ldots, s_n(p))$
$t(p) = (t_1(p), \cdots t_j(p), \cdots t_m(p))$

the weight matrix $W = \{w_{ij}\}$ can be given as

$$w = \sum_{p=1}^{n} S^T(p) . S(p)$$

ASSOCIATIVE MEMORY:

- **Associative memory** is also known as content addressable memory (CAM) or associative storage or associative array.
- It is a special type of memory that is optimized for performing searches through data, as opposed to providing a simple direct access to the data based on the address.

- It can store the set of patterns as memories when the associative memory is being presented with a key pattern, it responds by producing one of the stored pattern which closely resembles or relates to the key pattern.
- It can be viewed as **data correlation** here. Input data is correlated with that of stored data in the CAM.

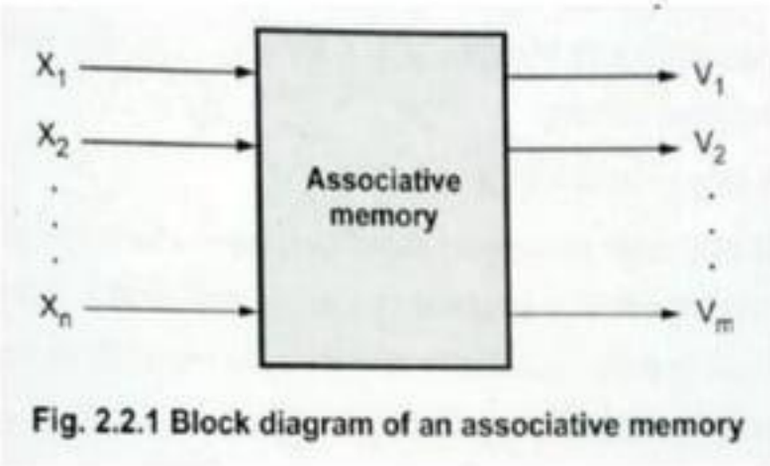

Fig. 2.2.1 Block diagram of an associative memory

There two types of associative memories

i. **Auto Associative Memory Network**

ii. **Hetero Associative memory Network**

2. AUTO ASSOCIATIVE MEMORY NETWORK:

- An auto-associative memory recovers a previously stored pattern that most closely relates to the current pattern. It is also known as an auto-associative correlate.
- In the auto associative memory network, the training input vector and training output vector are the same.

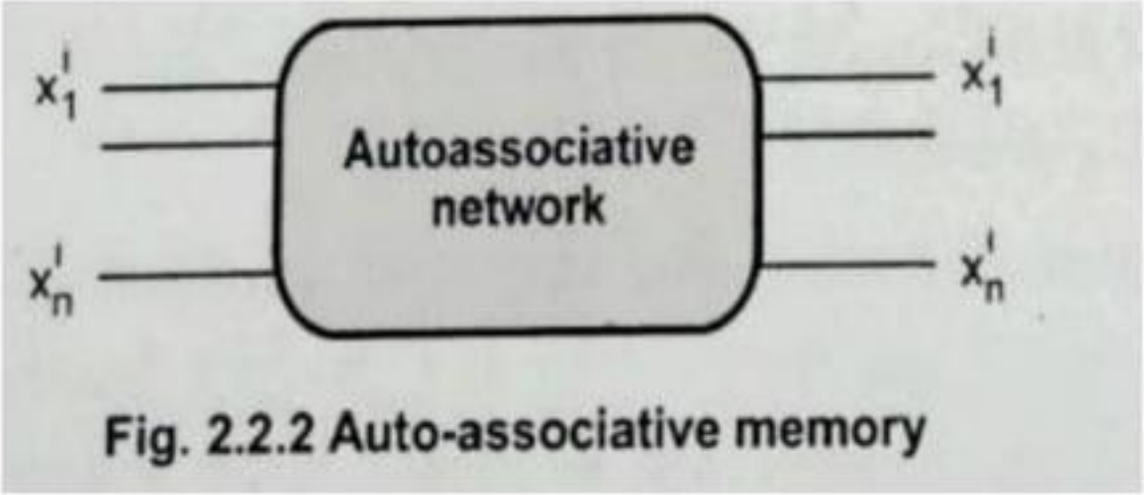

Fig. 2.2.2 Auto-associative memory

AUTO ASSOCIATIVE MEMORY ALGORITHM:

Training Algorithm

For training, this network is using the Hebb or Delta learning rule.

Step 1 – Initialize all the weights to zero as $w_{ij} = 0$ i=1ton, j=1ton

Step 2 – Perform steps 3-4 for each input vector.

Step 3 – Activate each input unit as follows –

$$x_i = s_i (i = 1ton)$$

Step 4 – Activate each output unit as follows –

$$y_j = s_j (j = 1ton)$$

Step 5 – Adjust the weights as follows –

$$w_{ij}(new) = w_{ij}(old) + x_i y_j$$

The weight can also be determine form the Hebb Rule or Outer Products Rule learning

$$\mathrm{w} = \sum_{p=1}^{n} S^{\mathrm{T}}(p) . \mathrm{S}(p)$$

Testing Algorithm

Step 1 – Set the weights obtained during training for Hebb's rule.

Step 2 – Perform steps 3-5 for each input vector.

Step 3 – Set the activation of the input units equal to that of the input vector.

Step 4 – Calculate the net input to each output unit j = 1 to n;

$$y_{inj} = \sum_{i=1}^{n} x_i w_{ij}$$

Step 5 – Apply the following activation function to calculate the output

$$y_j = f(y_{inj}) = \begin{cases} +1 & if y_{inj} > 0 \\ -1 & if y_{inj} \leqslant 0 \end{cases}$$

2. HETERO ASSOCIATIVE MEMORY NETWORK:

- In a hetero-associate memory, the training input and the target output vectors are different.
- The weights are determined in a way that the network can store a set of pattern associations. The association here is a pair of training input target output vector pairs (s (p), t(p)), with p = 1,2,…p.

- Each vector s(p) has n components and each vector t(p) has m components. The determination of weights is done either by using Hebb rule or delta rule.
- The net finds an appropriate output vector, which corresponds to an input vector x, that may be either one of the stored patterns or a new pattern.

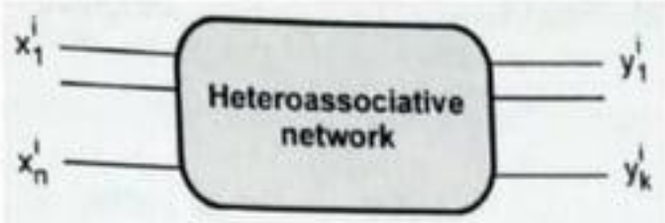

Fig. 2.2.3 Auto-associative memory

HETERO ASSOCIATIVE MEMORY ALGORITHM

Training Algorithm

Step 1 – Initialize all the weights to zero as $w_{ij} = 0$ i= 1 to n, j= 1 to m

Step 2 – Perform steps 3-4 for each input vector.

Step 3 – Activate each input unit as follows –

$$x_i = s_i(i = 1ton)$$

Step 4 – Activate each output unit as follows –

$$y_j = s_j(j = 1tom)$$

Step 5 – Adjust the weights as follows –

$$w_{ij}(new) = w_{ij}(old) + x_i y_j$$

The weight can also be determine form the Hebb Rule or Outer Products Rule learning

$$W = \sum_{p=1}^{n} S^T(p) . S(p)$$

Testing Algorithm

Step 1 – Set the weights obtained during training for Hebb's rule.

Step 2 – Perform steps 3-5 for each input vector.

Step 3 – Set the activation of the input units equal to that of the input vector.

Step 4 – Calculate the net input to each output unit j = 1 to m;

$$y_{inj} = \sum_{i=1}^{n} x_i w_{ij}$$

Step 5 – Apply the following activation function to calculate the output

$$y_j = f(y_{inj}) = \begin{cases} +1 & if y_{inj} > 0 \\ 0 & if y_{inj} = 0 \\ -1 & if y_{inj} < 0 \end{cases}$$