Pattern Association

Associative memory neural nets are single-layer nets in which the weights are determined in such a way that the net can store a set of pattern associations.

- Each association is an input-output vector pair, s: t.
- If each vector t is the same as the vectors with which it is associated, then the net is called an ***autoassociative memory.***
- If the t's are different from the s's, the net is called a ***heteroassociative memory.***
- In each of these cases, the net not only learns the specific pattern pairs that were used for training, but also is able to recall the desired response pattern when given an input stimulus that is similar, but not identical, to the training input.

Before training an associative memory neural net, the **original patterns must be converted to an appropriate representation** for computation. In a simple example, the **original pattern might consist of "on" and "off" signals**, and the conversion could be "on" = (+1), "off" = (**0)** (**binary representation**) or "on" = (+**1),** "off" =(-1) (**bipolar representation**).

TRAINING ALGORITHMS FOR PATTERN ASSOCIATION

1- Hebb Rule for Pattern Association:

- The Hebb rule is the simplest and most common method of determining the weights for an associative memory neural net.
- we denote our training vector pairs (input training-target output vectors) as **s:** t. We then denote our testing input vector as **x,** which may or may not be the same as one of the training input vectors.
- In the training algorithm of hebb rule the weights initially adjusted to 0, then updated using the following formula:

$w_{ij}(new) = w_{ij}(old) + x_i y_j$; (i = 1, . . . , *n; j* = 1, . . . , m):

where,

$$x_i = s_i$$

$$y_j = t_j$$

Outer products:

The weights found by using the Hebb rule (with all weights initially 0) can also be described in terms of outer products of the input vector-output vector pairs s:t. The outer product of two vectors

$\mathbf{s = (s_1, \ldots\ldots, s_i, \ldots\ldots, s_n)}$; $\mathbf{t = (t_1, \ldots\ldots, t_j, \ldots\ldots, t_m)}$

$$\mathbf{w = s^T t}$$

To store a set of associations s(p) : t(p), p = 1, . . . , P, where

$\mathbf{s(p) = (s_1(p), \ldots., s_i(p), \ldots., s_n(p))}$;

$\mathbf{t(p) = (t_1(p), \ldots\ldots., t_j(p), \ldots\ldots., t_m(p))}$

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

This is the sum of the outer product matrices required to store each association separately. In general, we shall use the preceding formula or the more concise vector matrix form,

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

- Several authors normalize the weights found by the Hebb rule by a factor of ***1/n,*** where ***n*** is the number of units in the system

2- Delta Rule for Pattern Association

In its original form, the delta rule assumed that the activation function for the output unit was the identity function. Thus, using y for the computed output for the input vector **x,** we have

$$y_J = net_J = \sum_{i=1}^{n} x_i w_{iJ}$$

The weights can be updated using the following equation:

$$\Delta w_{ij} = \alpha\ (t_j - y_j)\ x_i$$

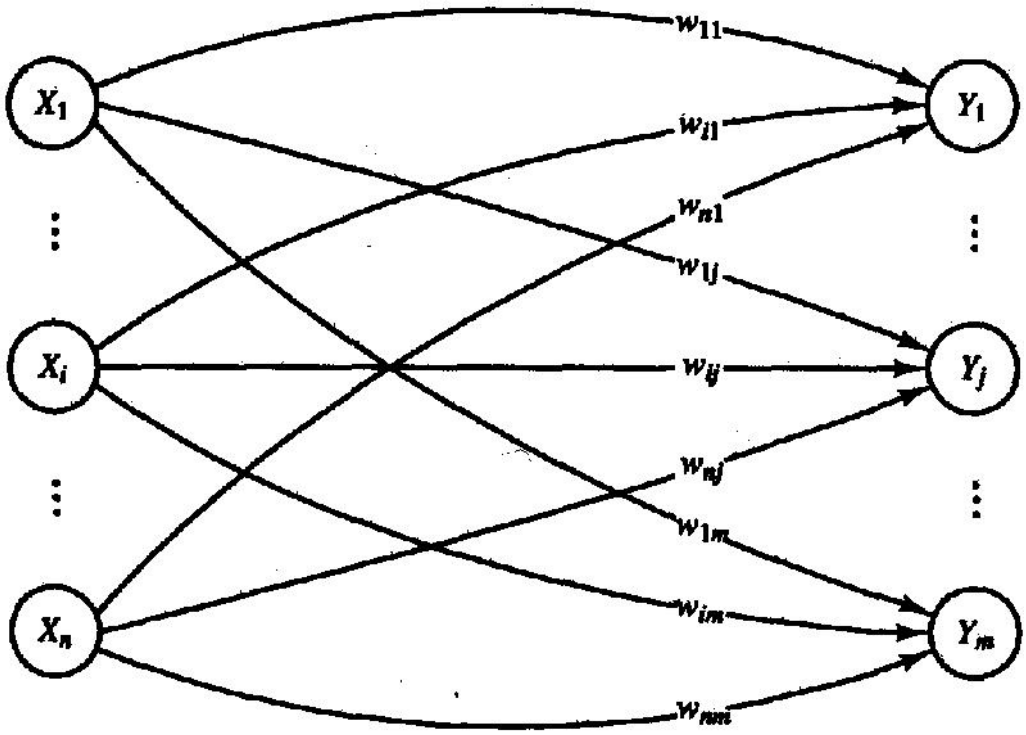
A simple extension allows for the use of any differentiable activation function; we shall call this the ***extended delta rule***. The update for the weight from the I'th input unit to the J'th output unit is:

$$\Delta w_{IJ} = \alpha\ (t_J - y_J)\ x_I f\ '(net_J)$$

1- HETEROASSOCIATIVE MEMORY NEURAL NETWORK

- Associative memory neural networks are nets in which the weights are determined in such a way that the net can store a set of P pattern associations.
- In heteroassociative memory the number of input units differ than that of output units.
- Each association is a pair of vectors (s(p), t(p)), with p = 1, 2, . . . , P. Each vector s(p) is an n-tuple (has n components), and each t(p) is an m-tuple.
- The weights may be found using the Hebb rule or the delta rule
- The net will find an appropriate output vector that corresponds to an input vector **x** that may be either one of the stored patterns s(p) or a new pattern (such as one of the training patterns corrupted by noise).

- The architecture of a heteroassociative memory neural network is as shown:



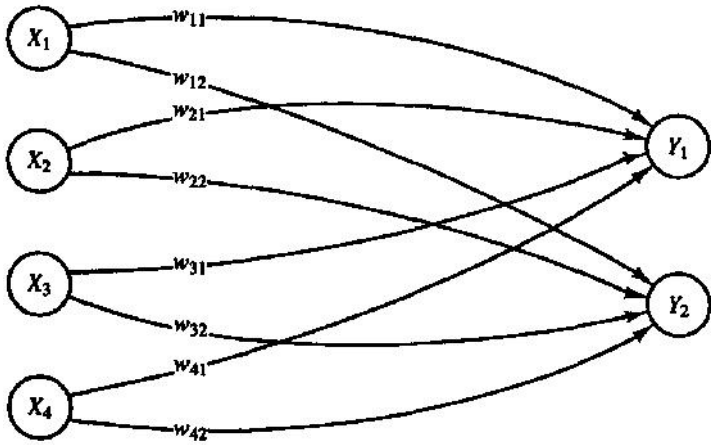
- For bipolar targets the activation of the output units:

$$y_j = f(net_j) = \begin{cases} 1 & if \quad net_j > 0 \\ 0 & if \quad net_j = 0 \\ -1 & if \quad net_j < 0 \end{cases}$$

- If the target responses of the net are binary, a suitable activation function is given by

$$f(x) = \begin{cases} 1 & if \quad x > 0 \\ 0 & if \quad x \leq 0 \end{cases}$$

Example-1: A heteroassociative neural net for a mapping from input vectors with four components to output vectors with two components is shown in the figure. The net is to be trained to store the following mapping from input row vector $s = (s_1, s_2, s_3, s_4)$ and output target row vector $t = (t_1, t_2)$ using the Hebb rule.



P		s_1	s_2	s_3	s_4		t_1	t_2
1	s(	1	0	0	0)	t(	1	0)
2	s(	1	1	0	0)	t(	1	0)
3	s(	0	0	0	1)	t(	0	1)
4	s(	0	0	1	1)	t(	0	1)

Sol:

The training is accomplished by the Hebb rule, which is defined as:

$$w_{ij}(new) = w_{ij}(old) + x_i y_j \quad ; \qquad \text{i.e., } \Delta w_{ij} = x_i y_j$$

$$x_i = s_i$$

$$y_j = t_j$$

Training:

$$\mathbf{W = 0}$$

Note: only the weights that change at each step of the process are shown):

1. For the first pattern p=1, **s: t** pair (1, 0, 0, 0):(1, 0):

xl = 1; x2 = x3 = x4 = 0.; yl = 1; y2 = 0.

$w_{11}(new) = w_{11}(old) + x_1 y_1 = 0 + 1 = 1$

(all other weights remain 0)

2. For the second pattern p=2, **s: t** pair (1, 1, 0, 0):(1, 0):

xl = x2 = 1 ; x3 = x4 = 0.; yl = 1; y2 = 0.

$w_{11}(new) = w_{11}(old) + x_1 y_1 = 1 + 1 = 2$

$w_{21}(new) = w_{21}(old) + x_2 y_1 = 0 + 1 = 1$

(all other weights remain 0)

3. For the third pattern p=3, **s: t** pair (0, 0, 0, 1):(0, 1):

xl = x2 = x3 = 0 x4 = 1; yl = 0; y2 = 1.

$W_{42}(new) = w_{42}(old) + x_4 y_2 = 0 + 1 = 1$

(all other weights remain unchanged)

4. For the fourth pattern p=4, **s: t** pair (0, 0, 1, 1):(0, 1):

xl = x2 = 0; x3 = x4 = 1; yl = 0; y2 = 1.

$W_{32}(new) = w_{32}(old) + x_3 y_2 = 0 + 1 = 1$

$W_{42}(new) = w_{42}(old) + x_4 y_2 = 1 + 1 = 2$

(all other weights remain unchanged)

The weight matrix is

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

Now let us find the weight vector using outer products instead of the algorithm for the Hebb rule.

The weight matrix to store the pattern pair (p) is given by the outer product of the vector s(p) and t(p):

$$\mathbf{W(p) = s(p)^T t(p)}$$

For p = 1 ; s = [1, 0, 0, 0] and t = [1, 0], the weight matrix is

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} [1 \quad 0] = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Similarly, to store the second pair, p = 2 ; s = [1, 1, 0, 0] and t = [1, 0]
The weight matrix is

$$\begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} [1 \quad 0] = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

To store the third pattern pair, p =3 ; s = [0, 0, 0, 1] and t = [0, 1] the weight matrix is

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

And to store the fourth pattern pair, p = 4; s = [0, 0, 1, 1] and t = [0, 1] the weight matrix is

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

The weight matrix to store all four pattern pairs is the sum of the weight matrices to store each pattern pair separately, namely,

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

We can also find the weight matrix to store all four patterns directly using the outer product

$$\mathbf{W} = \mathbf{s}^T \mathbf{t}$$

$$W = \begin{pmatrix}1 & 1 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 0 & 1\\0 & 0 & 1 & 1\end{pmatrix}\begin{pmatrix}1 & 0\\1 & 0\\0 & 1\\0 & 1\end{pmatrix}$$

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