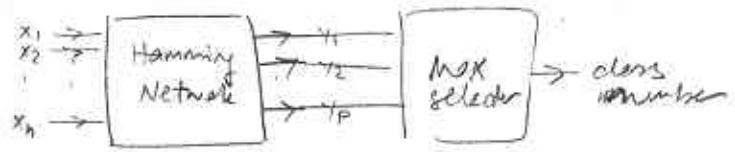
MATCHING AND SELF-ORGANIZING NETWORKS.

⇒ HAMMING NET AND MAXNET.

We will investigate a two layer classifier of binary bipolar vectors. The aim of the network is to choose a class whose Hamming distance from input is minimum (Min. Hamming distance classifier).



Here we have p classes. When $x = \begin{pmatrix} x_1 \\ x_2 \\ x_n \end{pmatrix}$ is presented the network the Hamming network gives an output $\begin{pmatrix} y_1 \\ y_p \end{pmatrix}$ such that the maximum entry corresponds to the class to which x has minimum Hamming distance. Another way of implementing the similar thing is to use what is called a MAXNET, whose goal is to make all other entries of y zero, and leave only the maximum entry nonzero. This is a recurrent network:

Suppose that p classes are characterized by p patterns

$$s^{(1)} = \begin{pmatrix} s_1^{(1)} \\ \vdots \\ s_n^{(1)} \end{pmatrix}, \quad \dots, \quad s^{(p)} = \begin{pmatrix} s_1^{(p)} \\ \vdots \\ s_n^{(p)} \end{pmatrix}$$

And an input pattern $x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ is presented. Our aim is to determine to which pattern x has minimum Hamming distance.

Note that $\boxed{s^{(i)t} x = n - 2\,HD(x, s^{(i)})}$

$$\Rightarrow \quad \frac{1}{2} s^{(i)t} x + \frac{n}{2} = n - HD(x, s^{(i)})$$

Let us define a Hamming matrix:

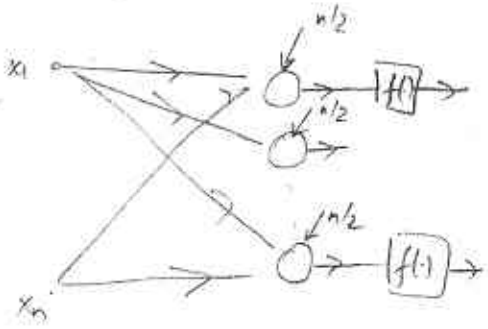
$$W_H = \frac{1}{2} \begin{bmatrix} s^{(1)t} \\ s^{(p)t} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} s_1^{(1)} & \cdots & s_n^{(1)} \\ & & \\ s_1^{(p)} & \cdots & s_n^{(p)} \end{bmatrix}$$

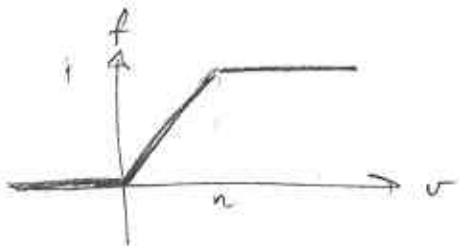
Let us define v as:

$$v = W_H x + b \qquad b = \begin{pmatrix} n/2 \\ n/2 \end{pmatrix} \Rightarrow \text{bias terms}$$

$$\Rightarrow \quad v_1 = \frac{1}{2} s^{(1)t} x + n/2 \quad \cdots \quad v_i = \frac{1}{2} s^{(i)t} x + n/2 \quad \cdots$$

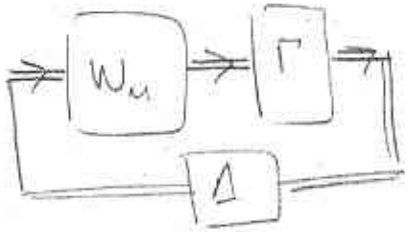
Hence the maximum entry v_i will give $v_i = n - HD(x, s^{(i)})$, hence the minimum HD. Hence x is close to it's class.





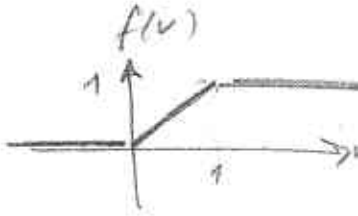
To force the outputs to be between $0 \le y_i \le 1$, we apply a special activation function $f(v)$ as shown above. Hence the outputs are always smaller than 1!

For the MAXNET part, we use the following function



For W_M we use

$$W_M = \begin{bmatrix} 1 & -\varepsilon & -\varepsilon & -\varepsilon \\ -\varepsilon & 1 & -\varepsilon & \\ & & 1 & -\varepsilon \\ & -\varepsilon & & 1 \end{bmatrix}$$



where $0 < \varepsilon < 1/p$. Here, when W_M multiplies the output of Hamming network, all outputs will decrease. The minimum one will reach zero first. To guarantee that, the activation function given above is used. Hence, after finite number of steps, only the node corresponding to the maximum output will be nonzero, all other entries will be zero.

EXAMPLE:

C I T

$s^{(1)} = (1\ 1\ 1\ 1\ -1\ -1\ 1\ 1\ 1)^t$ C

$s^{(2)} = (-1\ 1\ -1\ -1\ 1\ -1\ -1\ 1\ -1)^t$ I

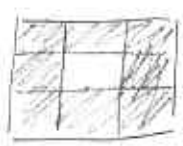
$s^{(3)} = (1\ 1\ 1\ -1\ 1\ -1\ -1\ 1\ -1)^t$ T

The Hamming network matrix W_H is then:

$$W_H = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 \\ -1 & 1 & -1 & -1 & +1 & -1 & -1 & 1 & -1 \\ 1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 \end{bmatrix}$$

For the maxnet, choose $\varepsilon = 0.2 < 1/3$.

Now, let us find the result of the following pattern.



→ close to C. $x = (1\ 1\ 1\ 1\ -1\ 1\ 1\ 1)$

$$v = \frac{1}{2} W_H x + \begin{pmatrix} 1/2 \\ 9/2 \\ 4/2 \end{pmatrix} = \begin{bmatrix} 7/2 \\ -5/2 \\ -1/2 \end{bmatrix} + \begin{bmatrix} 9/2 \\ 1/2 \\ 1/2 \end{bmatrix} = \begin{bmatrix} 16/2 \\ 4/2 \\ 8/2 \end{bmatrix}$$

$$o = \Gamma(v) = \frac{1}{9} v = \begin{bmatrix} 8/9 \\ 2/9 \\ 4/9 \end{bmatrix} \rightarrow \text{max. output}$$

$$W_{AA} = \begin{bmatrix} 1 & -0.2 & -0.2 \\ -0.2 & 1 & -0.2 \\ -0.2 & -0.2 & 1 \end{bmatrix}$$

$$W_M o = \begin{pmatrix} 0.78 \\ -0.06 \\ 0.18 \end{pmatrix} \rightarrow o(1) = \begin{pmatrix} 0.78 \\ 0 \\ 0.18 \end{pmatrix}$$

$$W_M o(1) = \begin{pmatrix} 0.74 \\ -0.19 \\ 0.01 \end{pmatrix} \rightarrow o(2) = \begin{pmatrix} 0.74 \\ 0 \\ 0.01 \end{pmatrix} \rightarrow o(3) = \Gamma\begin{pmatrix} 0.74 \\ -0.148 \\ -0.138 \end{pmatrix} = \begin{pmatrix} 0.74 \\ 0 \\ 0 \end{pmatrix}$$

Difference with Hopfield network: Size is smaller as compared to the Hopfield associative memory. (In Hopfield, W is $n \times n$, whereas here W_H is $n \times p$ and W_M is $p \times p$. If $p \ll n$, this method is more advantageous. On the other hand, Hopfield memory gives autoassociation i.e. if we give a distorted image of a stored pattern, Hopfield network may recover the undistorted pattern. However, here we only can recover the class membership. Hence the proposed architecture is not an associative memory