Example 2.1 Design a Hebb net to implement logical AND function using bipolar input-output patterns. The following example is illustrated below:

Solution The training pattern for an AND logic function is shown below

Training Patterns

	Input			Target	
	x_1	x_2	b		y
X_1	−1	−1	1	y_1	−1
X_2	−1	1	1	y_2	−1
X_3	1	−1	1	y_3	−1
X_4	1	1	1	y_4	1

A single layer network with 2 input neurons, one bias and one output neuron is considered. The initial weights are set to zero.

$$W(old) = [0 \quad 0 \quad 0]^T$$

Case I :

For the first input, $X_1 = [-1 \ -1 \ \ 1]$ and the target, $y_1 = [-1]$ the updated weight is:

$$W(new) = W(old) + X_1^T y_1$$
$$= [0 \quad 0 \quad 0]^T + [-1 \quad -1 \quad 1]^T [-1]$$
$$= [0 \quad 0 \quad 0]^T + [1 \quad 1 \quad -1]^T$$
$$= [1 \quad 1 \quad -1]^T$$

Case II :

On applying the second input vector, $X_2 = [-1 \;\; 1 \;\; 1]$ and the corresponding target, $y_2 = [-1]$ the new weight vector is :

$$W(new) = W(old) + X_2^T y_2$$
$$= [1 \;\; 1 \;\; -1]^T + [-1 \;\; 1 \;\; 1]^T[-1]$$
$$= [1 \;\; 1 \;\; -1]^T + [1 \;\; -1 \;\; -1]^T$$
$$= [2 \;\; 0 \;\; -2]^T$$

Case III:

For the third input pattern, $X_3 = [1 \;\; -1 \;\; 1]$ and the corresponding target, $y_3 = [-1]$ the new weight vector is:

$$W(new) = W(old) + X_3^T y_3$$
$$= [2 \;\; 0 \;\; -2]^T + [1 \;\; -1 \;\; 1]^T[-1]$$
$$= [2 \;\; 0 \;\; -2]^T + [-1 \;\; 1 \;\; -1]^T$$
$$= [1 \;\; 1 \;\; -3]^T$$

Case IV:

Applying the fourth input pattern, $X_4 = [1 \;\; 1 \;\; 1]$ and the corresponding target, $y_4 = [1]$ the final weight vector is:

$$W(new) = W(old) + X_4^T y_4$$
$$= [1 \;\; 1 \;\; -3]^T + [1 \;\; 1 \;\; 1]^T[1]$$
$$= [1 \;\; 1 \;\; -3]^T + [1 \;\; 1 \;\; 1]^T$$
$$= [2 \;\; 2 \;\; -2]^T$$

The Hebb net architecture with the final weight vector is shown in Fig. 2.3.

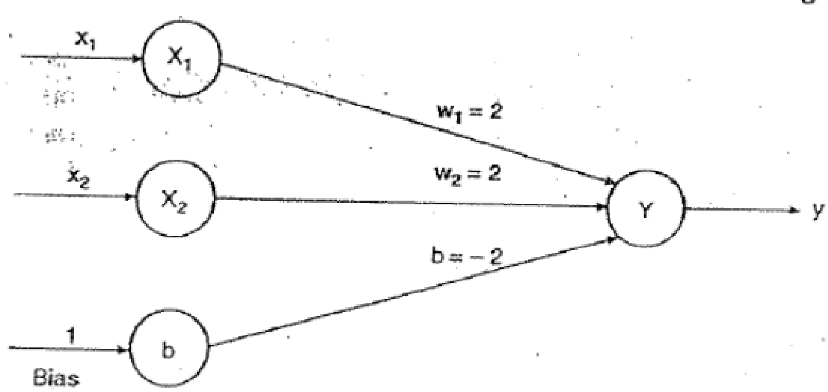


Figure 2.3 *Hebb Net for AND Function*

Example 2.2 Design or develop a Hebb net to implement logical OR function using bipolar input-output patterns. The following example is illustrated below:

Solution The training patterns for logical OR function in shown below.

Training Patterns

	Input			Target	
	x_1	x_2	b		y
X_1	−1	−1	1	y_1	−1
X_2	−1	1	1	y_2	1
X_3	1	−1	1	y_3	1
X_4	1	1	1	y_4	1

A single layer network with 2 input neurons, one bias and one output neuron is considered.

The initial weights are set to zero.

$W(old) = [0 \;\; 0 \;\; 0]^T$

Case I:

For the first input, $X_1 = [-1 \;\; -1 \;\; 1]$ and the target, $y_1 = [-1]$ the updated weight is:

$$W(new) = W(old) + X_1^T y_1$$
$$= [0 \;\; 0 \;\; 0]^T + [-1 \;\; -1 \;\; 1]^T [-1]$$
$$= [0 \;\; 0 \;\; 0]^T + [1 \;\; 1 \;\; -1]^T$$
$$= [1 \;\; 1 \;\; -1]^T$$

Case II:

On applying the second input vector, $X_2 = [-1 \;\; 1 \;\; 1]$ and the corresponding target, $y_2 = [1]$ the new weight vector is:

$$W(new) = W(old) + X_2^T y_2$$
$$= [1 \;\; 1 \;\; -1]^T + [-1 \;\; 1 \;\; 1]^T [1]$$
$$= [1 \;\; 1 \;\; -1]^T + [-1 \;\; 1 \;\; 1]^T$$
$$= [0 \;\; 2 \;\; 0]^T.$$

Case III:

For the third input pattern, $X_3 = [1 \ -1 \ 1]$ and the corresponding target, $y_3 = [1]$ the new weight vector is :

$$W(new) = W(old) + X_3^T y_3$$
$$= [0 \ 2 \ 0]^T + [1 \ -1 \ 1]^T[1]$$
$$= [0 \ 2 \ 0]^T + [1 \ -1 \ 1]^T$$
$$= [1 \ 1 \ 1]^T$$

Case IV:

Applying the fourth input pattern, $X_4 = [1 \ 1 \ 1]$ and the corresponding target,

$y_4 = [1]$ the final weight vector is :

$$W(new) = W(old) + X_4^T y^4$$
$$= [1 \ 1 \ 1]^T + [1 \ 1 \ 1]^T[1]$$
$$= [1 \ 1 \ 1]^T + [1 \ 1 \ 1]^T$$
$$= [2 \ 2 \ 2]^T$$

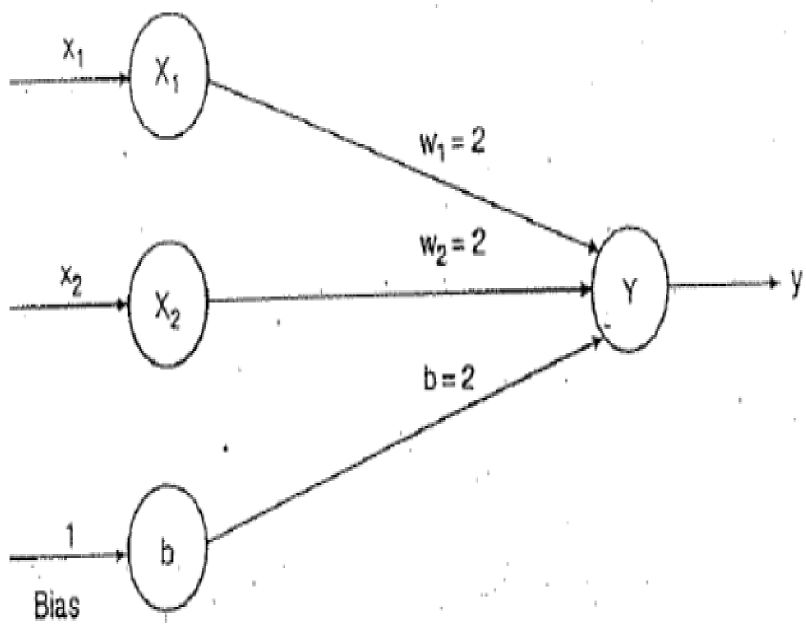


Figure 2.5 *Hebb Net for OR Function*

H.W

Example 2.3 Apply Hebb rule method to train patterns that define the AND NOT function and find the solution.

Solution The training patterns for an AND NOT logic function is shown below:

$[y = x_1 \text{ AND NOT } x_2]$

Training Patterns

	Input			Target	
	x_1	x_2	b		y
X_1	−1	−1	1	y_1	−1
X_2	−1	1	1	y_2	−1
X_3	1	−1	1	y_3	1
X_4	1	1	1	y_4	−1

A single layer network with 2 input neurons, one bias and one output neuron is considered.

H.W

Exercise Problems

3.22 Realize the McCulloch-Pitts neuron model for NAND gate and NOR gate.

3.23 Design a Hebb net for OR and NOT logic functions.

3.24 Realize ANDNOT function using Hebb net. Also form the decision boundary separating line.

3.25 Develop OR function using Hebb net for binary inputs and bipolar targets.

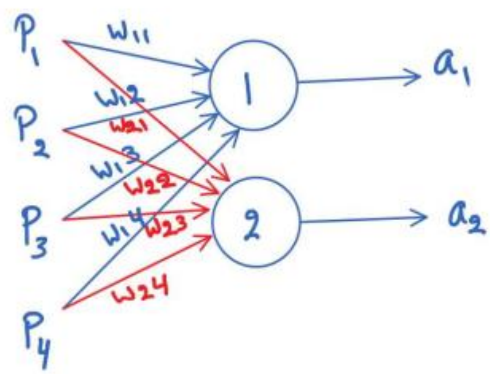
6.2.1.4 Example:

Let $\mathbf{p}_1 = \begin{bmatrix} 0.5 \\ -0.5 \\ 0.5 \\ -0.5 \end{bmatrix}, \mathbf{t}_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ and $\mathbf{p}_2 = \begin{bmatrix} 0.5 \\ 0.5 \\ -0.5 \\ -0.5 \end{bmatrix}, \mathbf{t}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. Use Hebbian learning rule to train the neural network. Graph the network. (Hint: Don't use bias)

Solution:

Take, initial weights are:

$$w_{1,i} = 0 \quad , w_{2,i} = 0 \quad i = 1, 2, 3, 4$$



Presenting the first input pattern $(\mathbf{p}_1, \mathbf{t}_1)$

$$w_{1,1} = t_1\, p_1 = (1)(0.5) = 0.5$$

$$w_{1,2} = t_1\, p_2 = (1)(-0.5) = -0.5$$

$$w_{1,3} = t_1\, p_3 = (1)(0.5) = 0.5$$

$$w_{1,4} = t_1\, p_4 = (1)(-0.5) = -0.5$$

$$w_{2,1} = t_2\, p_1 = (-1)(0.5) = -0.5$$

$$w_{2,2} = t_2\, p_2 = (-1)(-0.5) = 0.5$$

$$w_{2,3} = t_2\, p_3 = (-1)(0.5) = -0.5$$

$$w_{2,4} = t_2\, p_4 = (-1)(-0.5) = 0.5$$

Now, if we presenting the pattern $(\mathbf{p}_2, \mathbf{t}_2)$, we get wrong output

$$a_1 = w_{1,1}p_1 + w_{1,2}p_2 + w_{1,3}p_3 + w_{1,4}p_4$$

$$= (0.5)(0.5) + (-0.5)(0.5) + (0.5)(-0.5) + (-0.5)(-0.5) = 0$$

$$a_2 = w_{2,1}p_1 + w_{2,2}p_2 + w_{2,3}p_3 + w_{2,4}p_4$$

$$= (-0.5)(0.5) + (0.5)(0.5) + (-0.5)(-0.5) + (0.5)(-0.5) = 0$$

6.2.1.6 Example:

Is the following problem linear separable or not, explain your answer. Apply neural network to solve this problem.

p_1	p_2	t
1	1	0
−1	−1	0
1	−1	0
3	3	1

Solution:

Multiple – input neuron with threshold function $f(n) = \begin{cases} 0 & n < T \\ 1 & n > T \end{cases}$ can solve this problem as the following:

Method 3 (Hebb rule):

Use Hebb rule with threshold function $f(n) = \begin{cases} 0 & n < 7 \\ 1 & n > 7 \end{cases}$

$$\mathbf{W} = \mathbf{TP}^T = [0 \quad 0 \quad 0 \quad 1]\begin{bmatrix} 1 & 1 \\ -1 & -1 \\ 1 & -1 \\ 3 & 3 \end{bmatrix} = [3 \quad 3]$$

Now, check

$$\Rightarrow \ \mathbf{a} = f(\mathbf{WP}) = f\left([3 \quad 3]\begin{bmatrix} 1 & -1 & 1 & 3 \\ 1 & -1 & -1 & 3 \end{bmatrix}\right) = f\left(\begin{bmatrix} 6 \\ -6 \\ 0 \\ 18 \end{bmatrix}\right) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$