1. Implement AND function using perceptron networks for bipolar inputs and targets.

**Solution:** Table 1 shows the truth table for AND function with bipolar inputs and targets:

**Table 1**

| $x_1$ | $x_2$ | $t$ |
|---|---|---|
| 1 | 1 | 1 |
| 1 | −1 | −1 |
| −1 | 1 | −1 |
| −1 | −1 | −1 |

The perceptron network, which uses perceptron learning rule, is used to train the AND function. The network architecture is as shown in Figure 1. The input patterns are presented to the network one by one. When all the four input patterns are presented, then one epoch is said to be completed. The initial weights and threshold are set to zero, i.e., $w_1 = w_2 = b = 0$ and $\theta = 0$. The learning rate $\alpha$ is set equal to 1.

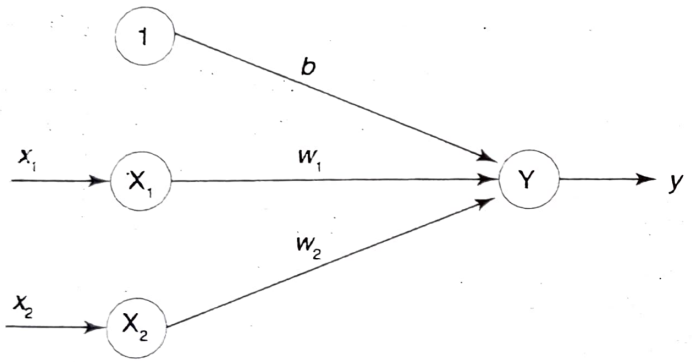


**Figure 1** Perceptron network for AND function.

For the first input pattern, $x_1 = 1, x_2 = 1$ and $t = 1$, with weights and bias, $w_1 = 0$, $w_2 = 0$ and $b = 0$:

- Calculate the net input

$$y_{in} = b + x_1 w_1 + x_2 w_2$$
$$= 0 + 1 \times 0 + 1 \times 0 = 0$$

- The output $y$ is computed by applying activations over the net input calculated:

$$y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} > 0 \\ 0 & \text{if } y_{in} = 0 \\ -1 & \text{if } y_{in} < 0 \end{cases}$$

Here we have taken $\theta = 0$. Hence, when, $y_{in} = 0$, $y = 0$.

- Check whether $t = y$. Here, $t = 1$ and $y = 0$, so $t \neq y$, hence weight updation takes place:

$$w_i(\text{new}) = w_i(\text{old}) + \alpha t x_i$$
$$w_1(\text{new}) = w_1(\text{old}) + \alpha t x_1 = 0 + 1 \times 1 \times 1 = 1$$
$$w_2(\text{new}) = w_2(\text{old}) + \alpha t x_2 = 0 + 1 \times 1 \times 1 = 1$$
$$b(\text{new}) = b(\text{old}) + \alpha t = 0 + 1 \times 1 = 1$$

Here, the change in weights are

$$\Delta w_1 = \alpha t x_1;$$
$$\Delta w_2 = \alpha t x_2;$$
$$\Delta b = \alpha t$$

The weights $w_1 = 1$, $w_2 = 1$, $b = 1$ are the final weights after first input pattern is presented. The same process is repeated for all the input patterns. The process can be stopped when all the targets become equal to the calculated output or when a separating line is obtained using the final weights for separating the positive responses from negative responses. Table 2 shows the training of perceptron network until its

**Table 2**

| Input | | | Target | Net input | Calculated output | Weight changes | | | Weights | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | $w_1$ | $w_2$ | $b$ |
| $x_1$ | $x_2$ | 1 | ($t$) | ($y_{in}$) | ($y$) | $\Delta w_1$ | $\Delta w_2$ | $\Delta b$ | (0 | 0 | 0) |
| EPOCH-1 | | | | | | | | | | | |
| 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | −1 | 1 | −1 | 1 | 1 | −1 | 1 | −1 | 0 | 2 | 0 |
| −1 | 1 | 1 | −1 | 2 | 1 | +1 | −1 | −1 | 1 | 1 | −1 |
| −1 | −1 | 1 | −1 | −3 | −1 | 0 | 0 | 0 | 1 | 1 | −1 |
| EPOCH-2 | | | | | | | | | | | |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | −1 |
| 1 | −1 | 1 | −1 | −1 | −1 | 0 | 0 | 0 | 1 | 1 | −1 |
| −1 | 1 | 1 | −1 | −1 | −1 | 0 | 0 | 0 | 1 | 1 | −1 |
| −1 | −1 | 1 | −1 | −3 | −1 | 0 | 0 | 0 | 1 | 1 | −1 |

target and calculated output converge for all the patterns.

The final weights and bias after second epoch are

$$w_1 = 1, w_2 = 1, \ b = -1$$

Since the threshold for the problem is zero, the equation of the separating line is

$$x_2 = -\frac{w_1}{w_2}x_1 - \frac{b}{w_2}$$

Here

$$w_1x_1 + w_2x_2 + b > \theta$$
$$w_1x_1 + w_2x_2 + b > 0$$

Thus, using the final weights we obtain

$$x_2 = -\frac{1}{1}x_1 - \frac{(-1)}{1}$$
$$x_2 = -x_1 + 1$$

It can be easily found that the above straight line separates the positive response and negative response region, as shown in Figure 2.

The same methodology can be applied for implementing other logic functions such as OR, AND-NOT, NAND, etc. If there exists a threshold value $\theta \neq 0$, then two separating lines have to be obtained, i.e., one to separate positive response from zero and the other for separating zero from the negative response.

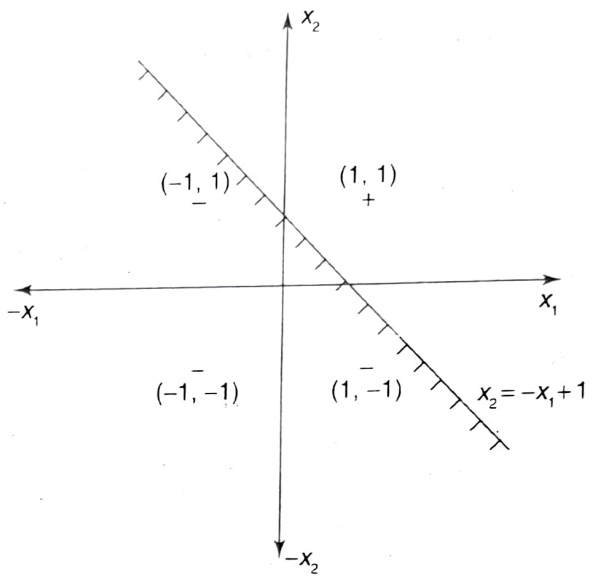


**Figure 2** Decision boundary for AND function in perceptron training ($\theta = 0$).

2. Implement OR function with binary inputs and bipolar targets using perceptron training algorithm upto 3 epochs.

**Solution:** The truth table for OR function with binary inputs and bipolar targets is shown in Table 3.

**Table 3**

| $x_1$ | $x_2$ | $t$ |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | −1 |

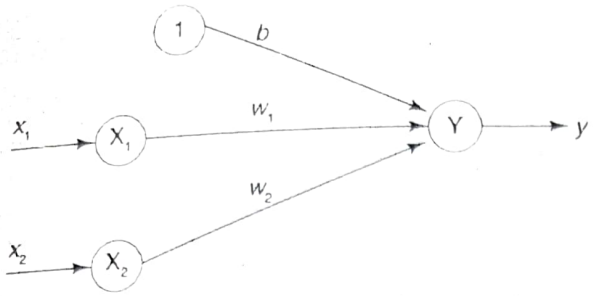


**Figure 3** Perceptron network for OR function.

The perceptron network, which uses perceptron learning rule, is used to train the OR function. The network architecture is shown in Figure 3. The initial values of the weights and bias are taken as zero, i.e.,

$$w_1 = w_2 = b = 0$$

Also the learning rate is 1 and threshold is 0.2. So, the activation function becomes

$$f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} > 0.2 \\ 0 & \text{if } -0.2 \le y_{in} \le 0.2 \end{cases}$$

The network is trained as per the perceptron training algorithm and the steps are as in problem 1 (given for first pattern). Table 4 gives the network training for 3 epochs.

The final weights at the end of third epoch are

$$w_1 = 2, w_2 = 1, b = -1$$

Further epochs have to be done for the convergence of the network.

3. Find the weights using perceptron network for ANDNOT function when all the inputs are presented only one time. Use bipolar inputs and targets.

**Solution:** The truth table for ANDNOT function is shown in Table 5.

**Table 5**

| $x_1$ | $x_2$ | $t$ |
|---|---|---|
| 1 | 1 | −1 |
| 1 | −1 | 1 |
| −1 | 1 | −1 |
| −1 | −1 | −1 |

The network architecture of ANDNOT function is shown as in Figure 4. Let the initial weights be zero and $\alpha = 1, \theta = 0$. For the first input sample, we compute the net input as

$$y_{in} = b + \sum_{i=1}^{n} x_i w_i = b + x_1 w_1 + x_2 w_2$$

$$= 0 + 1 \times 0 + 1 \times 0 = 0$$

**Table 4**

| Input | | | Target | Net input | Calculated output | Weight changes | | | Weights | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $x_1$ | $x_2$ | 1 | ($t$) | ($y_{in}$) | ($y$) | $\Delta w_1$ | $\Delta w_2$ | $\Delta b$ | $w_1$ (0 | $w_2$ 0 | $b$ 0) |
| EPOCH-1 | | | | | | | | | | | |
| 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 2 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | −1 | 1 | 1 | 0 | 0 | −1 | 1 | 1 | 0 |
| EPOCH-2 | | | | | | | | | | | |
| 1 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | −1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | −1 |
| EPOCH-3 | | | | | | | | | | | |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | −1 |
| 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 2 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 2 | 1 | 0 |
| 0 | 0 | 1 | −1 | 0 | 0 | 0 | 0 | −1 | 2 | 1 | −1 |

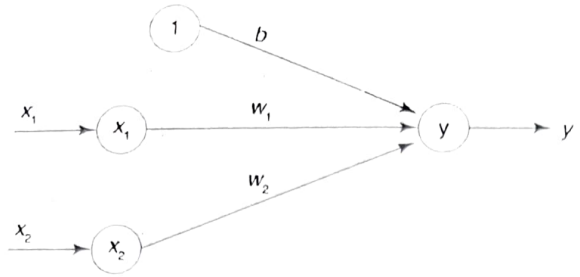


**Figure 4** Network for ANDNOT function.

Applying the activation function over the net input, we obtain

$$y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} > 0 \\ 0 & \text{if } -0 \le y_{in} \le 0 \\ -1 & \text{if } y_{in} < -0 \end{cases}$$

Hence, the output $y = f(y_{in}) = 0$. Since $t \neq y$, the new weights are computed as

$$w_1(\text{new}) = w_1(\text{old}) + \alpha t x_1 = 0 + 1 \times -1 \times 1 = -1$$

$$w_2(\text{new}) = w_2(\text{old}) + \alpha t x_2 = 0 + 1 \times -1 \times 1 = -1$$

$$b(\text{new}) = b(\text{old}) + \alpha t = 0 + 1 \times -1 = -1$$

The weights after presenting the first sample are

$$w = [-1 \quad -1 \quad -1]$$

For the second input sample, we calculate the net input as

$$y_{in} = b + \sum_{i=1}^{n} x_i w_i = b + x_1 w_1 + x_2 w_2$$

$$= -1 + 1 \times -1 + (-1 \times -1)$$

$$= -1 - 1 + 1 = -1$$

The output $y = f(y_{in})$ is obtained by applying activation function, hence $y = -1$.

Since $t \neq y$, the new weights are calculated as

$$w_1(\text{new}) = w_1(\text{old}) + \alpha t x_1 = -1 + 1 \times 1 \times 1 = 0$$

$$w_2(\text{new}) = w_2(\text{old}) + \alpha t x_2 = -1 + 1 \times 1 \times -1 = -2$$

$$b(\text{new}) = b(\text{old}) + \alpha t = -1 + 1 \times 1 = 0$$

The weights after presenting the second sample are

$$w = [0 \quad -2 \quad 0]$$

For the third input sample, $x_1 = -1$, $x_2 = 1$, $t = -1$, the net input is calculated as,

$$y_{in} = b + \sum_{i=1}^{n} x_i w_i = b + x_1 w_1 + x_2 w_2$$

$$= 0 + -1 \times 0 + 1 \times -2 = 0 + 0 - 2 = -2$$

The output is obtained as $y = f(y_{in}) = -1$. Since $t = y$, no weight changes. Thus, even after presenting the third input sample, the weights are

$$w = [0 \quad -2 \quad 0]$$

For the fourth input sample, $x_1 = -1, x_2 = -1$, $t = -1$, the net input is calculated as

$$y_{in} = b + \sum_{i=1}^{n} x_i w_i = b + x_1 w_1 + x_2 w_2$$

$$= 0 + -1 \times 0 + (-1 \times -2)$$

$$= 0 + 0 + 2 = 2$$

The output is obtained as $y = f(y_{in}) = 1$. Since $t \neq y$, the new weights on updating are given as

$$w_1(\text{new}) = w_1(\text{old}) + \alpha t x_1 = 0 + 1 \times -1 \times -1 = 1$$

$$w_2(\text{new}) = w_2(\text{old}) + \alpha t x_2 = -2 + 1 \times -1 \times -1 = -1$$

$$b(\text{new}) = b(\text{old}) + \alpha t = 0 + 1 \times -1 = -1$$

The weights after presenting fourth input sample are

$$w = [1 \quad -1 \quad -1]$$

One epoch of training for ANDNOT function using perceptron network is tabulated in Table 6.

**Table 6**

| Input | | | Target | Net input | Calculated output | Weights | | |
|---|---|---|---|---|---|---|---|---|
| $x_1$ | $x_2$ | 1 | ($t$) | ($y_{in}$) | ($y$) | $w_1$ (0 | $w_2$ 0 | $b$ 0) |
| 1 | 1 | 1 | −1 | 0 | 0 | −1 | −1 | −1 |
| 1 | −1 | 1 | 1 | −1 | −1 | 0 | −2 | 0 |
| −1 | 1 | 1 | −1 | −2 | −1 | 0 | −2 | 0 |
| −1 | −1 | 1 | −1 | 2 | 1 | 1 | −1 | −1 |

4. Find the weights required to perform the following classification using perceptron network. The vectors (1, 1, 1, 1) and (−1, 1 −1, −1) are belonging to the class (so have target value 1), vectors (1, 1, 1, −1) and (1, −1, −1, 1) are not belonging to the class (so have target value −1). Assume learning rate as 1 and initial weights as 0.

**Solution:** The truth table for the given vectors is given in Table 7.

**Table 7**

| Input | | | | | |
|---|---|---|---|---|---|
| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $b$ | Target ($t$) |
| 1 | 1 | 1 | 1 | 1 | 1 |
| −1 | 1 | −1 | −1 | 1 | 1 |
| 1 | 1 | 1 | −1 | 1 | −1 |
| 1 | −1 | −1 | 1 | 1 | −1 |

Let $w_1 = w_2 = w_3 = w_4 = b = 0$ and the learning rate $\alpha = 1$. Since the threshold $\theta = 0.2$, so the activation function is

$$y = \begin{cases} 1 & \text{if } y_{in} > 0.2 \\ 0 & \text{if } -0.2 \le y_{in} \le 0.2 \\ -1 & \text{if } y_{in} < -0.2 \end{cases}$$

The net input is given by

$$y_{in} = b + x_1 w_1 + x_2 w_2 + x_3 w_3 + x_4 w_4$$

The training is performed and the weights are tabulated in Table 8.

Thus, in the third epoch, all the calculated outputs become equal to targets and the network has converged. The network convergence can also be checked by forming separating line equations for separating positive response regions from zero and zero from negative response region.

The network architecture is shown in Figure 5.

5. Classify the two-dimensional input pattern shown in Figure 6 using perceptron network. The symbol "*" indicates the data representation to be +1 and "•" indicates data to be −1. The patterns are I-F. For pattern I, the target is +1, and for F, the target is −1.

**Table 8**

| Inputs | | | | | Target | Net input | Output | Weight changes | | | | | Weights | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ($x_1$ | $x_2$ | $x_3$ | $x_4$ | $b$) | ($t$) | ($y_{in}$) | ($y$) | ($\Delta w_1$ | $\Delta w_2$ | $\Delta w_3$ | $\Delta w_4$ | $\Delta b$) | ($w_1$ | $w_2$ | $w_3$ | $w_4$ | $b$) |
| | | | | | | | | | | | | | (0 | 0 | 0 | 0 | 0) |
| EPOCH-1 | | | | | | | | | | | | | | | | | |
| ( 1 | 1 | 1 | 1 | 1) | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| (−1 | 1 | −1 | −1 | 1) | 1 | −1 | −1 | −1 | 1 | −1 | −1 | 1 | 0 | 2 | 0 | 0 | 2 |
| ( 1 | 1 | 1 | −1 | 1) | −1 | 4 | 1 | −1 | −1 | −1 | 1 | −1 | −1 | 1 | −1 | 1 | 1 |
| ( 1 | −1 | −1 | 1 | 1) | −1 | 1 | 1 | −1 | 1 | 1 | −1 | −1 | −2 | 2 | 0 | 0 | 0 |
| EPOCH-2 | | | | | | | | | | | | | | | | | |
| ( 1 | 1 | 1 | 1 | 1) | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | −1 | 3 | 1 | 1 | 1 |
| (−1 | 1 | −1 | −1 | 1) | 1 | 3 | 1 | 0 | 0 | 0 | 0 | 0 | −1 | 3 | 1 | 1 | 1 |
| ( 1 | 1 | 1 | −1 | 1) | −1 | 4 | 1 | −1 | −1 | −1 | 1 | −1 | −2 | 2 | 0 | 2 | 0 |
| ( 1 | −1 | −1 | 1 | 1) | −1 | −2 | −1 | 0 | 0 | 0 | 0 | 0 | −2 | 2 | 0 | 2 | 0 |
| EPOCH-3 | | | | | | | | | | | | | | | | | |
| ( 1 | 1 | 1 | 1 | 1) | 1 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | −2 | 2 | 0 | 2 | 0 |
| (−1 | 1 | −1 | −1 | 1) | 1 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | −2 | 2 | 0 | 2 | 0 |
| ( 1 | 1 | 1 | −1 | 1) | −1 | −2 | −1 | 0 | 0 | 0 | 0 | 0 | −2 | 2 | 0 | 2 | 0 |
| ( 1 | −1 | −1 | 1 | 1) | −1 | −2 | −1 | 0 | 0 | 0 | 0 | 0 | −2 | 2 | 0 | 2 | 0 |

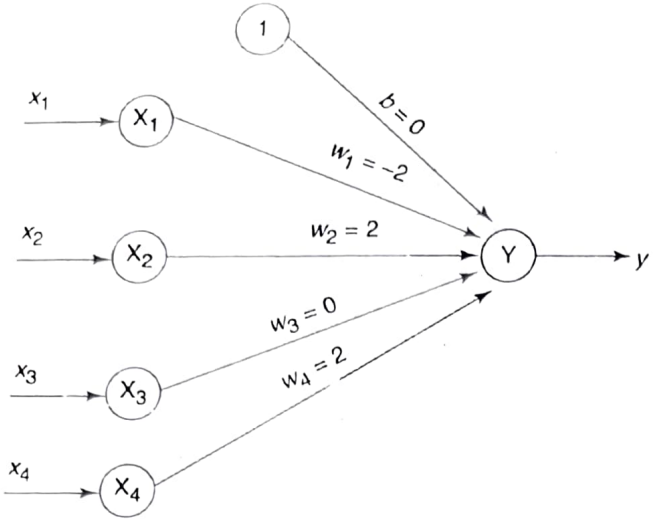


**Figure 5** Network architecture.

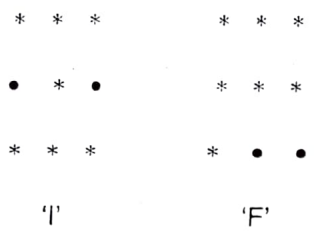


**Figure 6** I-F data representation.

**Solution:** The training patterns for this problem are tabulated in Table 9.

**Table 9**

| | Input | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Pattern | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | $x_8$ | $x_9$ | 1 | Target ($t$) |
| I | 1 | 1 | 1 | −1 | 1 | −1 | 1 | 1 | 1 | 1 | 1 |
| F | 1 | 1 | 1 | 1 | 1 | 1 | 1 | −1 | −1 | 1 | −1 |

The initial weights are all assumed to be zero, i.e., $\theta = 0$ and $\alpha = 1$. The activation function is given by

$$y = \begin{cases} 1 & \text{if } y_{in} > 0 \\ 0 & \text{if } -0 \le y_{in} \le 0 \\ -1 & \text{if } y_{in} < -0 \end{cases}$$

For the first input sample, $x_1 = [1\ 1\ 1\ -1\ 1\ -1\ 1\ 1\ 1]$, $t = 1$, the net input is calculated as

$$y_{in} = b + \sum_{i=1}^{9} x_i w_i$$

$$= b + x_1 w_1 + x_2 w_2 + x_3 w_3 + x_4 w_4 + x_5 w_5 + x_6 w_6 + x_7 w_7 + x_8 w_8 + x_9 w_9$$

$$= 0 + 1 \times 0 + 1 \times 0 + 1 \times 0 + (-1) \times 0 + 1 \times 0 + (-1) \times 0 + 1 \times 0 + 1 \times 0$$

$$y_{in} = 0$$

Therefore, by applying the activation function the output is given by $y = f(y_{in}) = 0$. Now since $t \neq y$, the new weights are computed as

$$w_1(\text{new}) = w_1(\text{old}) + \alpha t x_1 = 0 + 1 \times 1 \times 1 = 1$$
$$w_2(\text{new}) = w_2(\text{old}) + \alpha t x_2 = 0 + 1 \times 1 \times 1 = 1$$
$$w_3(\text{new}) = w_3(\text{old}) + \alpha t x_3 = 0 + 1 \times 1 \times 1 = 1$$
$$w_4(\text{new}) = w_4(\text{old}) + \alpha t x_4 = 0 + 1 \times 1 \times -1 = -1$$
$$w_5(\text{new}) = w_5(\text{old}) + \alpha t x_5 = 0 + 1 \times 1 \times 1 = 1$$
$$w_6(\text{new}) = w_6(\text{old}) + \alpha t x_6 = 0 + 1 \times 1 \times -1 = -1$$
$$w_7(\text{new}) = w_7(\text{old}) + \alpha t x_7 = 0 + 1 \times 1 \times 1 = 1$$
$$w_8(\text{new}) = w_8(\text{old}) + \alpha t x_8 = 0 + 1 \times 1 \times 1 = 1$$
$$w_9(\text{new}) = w_9(\text{old}) + \alpha t x_9 = 0 + 1 \times 1 \times 1 = 1$$
$$b(\text{new}) = b(\text{old}) + \alpha t = 0 + 1 \times 1 = 1$$

The weights after presenting first input sample are

$$w = [1\ 1\ 1\ -1\ 1\ -1\ 1\ 1\ 1\ 1]$$

For the second input sample, $x_2 = [1\ 1\ 1\ 1\ 1\ 1\ 1\ -1\ -1\ 1]$, $t = -1$, the net input is calculated as

$$y_{in} = b + \sum_{i=1}^{9} x_i w_i$$

$$= b + x_1 w_1 + x_2 w_2 + x_3 w_3 + x_4 w_4 + x_5 w_5 + x_6 w_6 + x_7 w_7 + x_8 w_8 + x_9 w_9$$

$$= 1 + 1 \times 1 + 1 \times 1 + 1 \times 1 + 1 \times -1 + 1 \times 1 + 1 \times -1 + 1 \times 1 + (-1) \times 1 + (-1) \times 1$$

$$y_{in} = 2$$

Therefore the output is given by $y = f(y_{in}) = 1$. Since $t \neq y$, the new weights are

$$w_1(\text{new}) = w_1(\text{old}) + \alpha t x_1 = 1 + 1 \times -1 \times 1 = 0$$
$$w_2(\text{new}) = w_2(\text{old}) + \alpha t x_2 = 1 + 1 \times -1 \times 1 = 0$$
$$w_3(\text{new}) = w_3(\text{old}) + \alpha t x_3 = 1 + 1 \times -1 \times 1 = 0$$
$$w_4(\text{new}) = w_4(\text{old}) + \alpha t x_4 = -1 + 1 \times -1 \times 1 = -2$$

$$w_5(\text{new}) = w_5(\text{old}) + \alpha t x_5 = 1 + 1 \times -1 \times 1 = 0$$
$$w_6(\text{new}) = w_6(\text{old}) + \alpha t x_6 = -1 + 1 \times -1 \times 1 = -2$$
$$w_7(\text{new}) = w_7(\text{old}) + \alpha t x_7 = 1 + 1 \times -1 \times 1 = 0$$
$$w_8(\text{new}) = w_8(\text{old}) + \alpha t x_8 = 1 + 1 \times -1 \times -1 = 2$$
$$w_9(\text{new}) = w_9(\text{old}) + \alpha t x_9 = 1 + 1 \times -1 \times -1 = 2$$
$$b(\text{new}) = b(\text{old}) + \alpha t = 1 + 1 \times -1 = 0$$

The weights after presenting the second input sample are

$$w = [0\ 0\ 0\ \ -2\ 0\ \ -2\ 0\ 2\ 2\ 0]$$

The network architecture is as shown in Figure 7. The network can be further trained for its convergence.

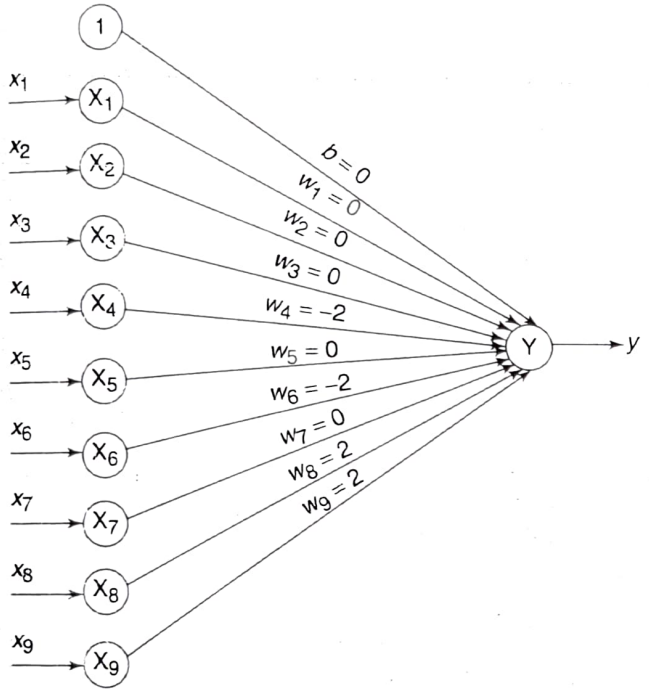


**Figure 7** Network architecture.

6. Implement OR function with bipolar inputs and targets using Adaline network.

**Solution:** The truth table for OR function with bipolar inputs and targets is shown in Table 10.

**Table 10**

| $x_1$ | $x_2$ | 1 | $t$ |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 1 | −1 | 1 | 1 |
| −1 | 1 | 1 | 1 |
| −1 | −1 | 1 | −1 |

Initially all the weights and links are assumed to be small random values, say 0.1, and the learning rate is also set to 0.1. Also here the least mean square error may be set. The weights are calculated until the least mean square error is obtained.

The initial weights are taken to be $w_1 = w_2 = b = 0.1$ and the learning rate $\alpha = 0.1$. For the first input sample, $x_1 = 1$, $x_2 = 1$, $t = 1$, we calculate the net input as

$$y_{in} = b + \sum_{i=1}^{n} x_i w_i = b + \sum_{i=1}^{2} x_i w_i$$
$$= b + x_1 w_1 + x_2 w_2$$
$$= 0.1 + 1 \times 0.1 + 1 \times 0.1 = 0.3$$

Now compute $(t - y_{in}) = (1 - 0.3) = 0.7$. Updating the weights we obtain,

$$w_i(\text{new}) = w_i(\text{old}) + \alpha(t - y_{in})x_i$$

where $\alpha(t - y_{in})x_i$ is called as weight change $\Delta w_i$. The new weights are obtained as

$$w_1(\text{new}) = w_1(\text{old}) + \Delta w_1 = 0.1 + 0.1 \times 0.7 \times 1$$
$$= 0.1 + 0.07 = 0.17$$
$$w_2(\text{new}) = w_2(\text{old}) + \Delta w_2 = 0.1$$
$$+ 0.1 \times 0.7 \times 1 = 0.17$$
$$b(\text{new}) = b(\text{old}) + \Delta b = 0.1 + 0.1 \times 0.7 = 0.17$$

where

$$\Delta w_1 = \alpha(t - y_{in})x_1$$
$$\Delta w_2 = \alpha(t - y_{in})x_2$$
$$\Delta b = \alpha(t - y_{in})$$

Now we calculate the error:

$$E = (t - y_{in})^2 = (0.7)^2 = 0.49$$

The final weights after presenting first input sample are

$$w = [0.17\ \ 0.17\ \ 0.17]$$

and error $E = 0.49$.

These calculations are performed for all the input samples and the error is calculated. One epoch is completed when all the input patterns are presented. Summing up all the errors obtained for each input sample during one epoch will give the total mean square error of that epoch. The network training is continued until this error is minimized to a very small value.

Adopting the method above, the network training is done for OR function using Adaline network and is tabulated below in Table 11 for $\alpha = 0.1$.

The total mean square error after each epoch is given as in Table 12.

Thus from Table 12, it can be noticed that as training goes on, the error value gets minimized. Hence, further training can be continued for further minimization of error. The network architecture of Adaline network for OR function is shown in Figure 8.

**Table 12**

| Epoch | Total mean square error |
|---|---|
| Epoch 1 | 3.02 |
| Epoch 2 | 1.938 |
| Epoch 3 | 1.5506 |
| Epoch 4 | 1.417 |
| Epoch 5 | 1.377 |

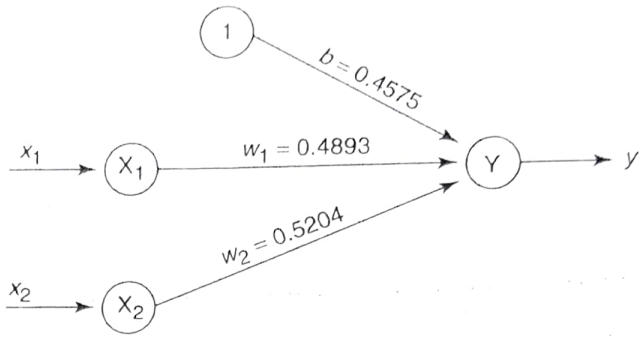


**Figure 8** Network architecture of Adaline.

**Table 11**

| Inputs | | | Target | Net input | | Weight changes | | | Weights | | | Error |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $x_1$ | $x_2$ | 1 | $t$ | $y_{in}$ | $(t - y_{in})$ | $\Delta w_1$ | $\Delta w_2$ | $\Delta b$ | $w_1$ (0.1 | $w_2$ 0.1 | $b$ 0.1) | $(t - y_{in})^2$ |
| EPOCH-1 | | | | | | | | | | | | |
| 1 | 1 | 1 | 1 | 0.3 | 0.7 | 0.07 | 0.07 | 0.07 | 0.17 | 0.17 | 0.17 | 0.49 |
| 1 | −1 | 1 | 1 | 0.17 | 0.83 | 0.083 | −0.083 | 0.083 | 0.253 | 0.087 | 0.253 | 0.69 |
| −1 | 1 | 1 | 1 | 0.087 | 0.913 | −0.0913 | 0.0913 | 0.0913 | 0.1617 | 0.1783 | 0.3443 | 0.83 |
| −1 | −1 | 1 | −1 | 0.0043 | −1.0043 | 0.1004 | 0.1004 | −0.1004 | 0.2621 | 0.2787 | 0.2439 | 1.01 |
| EPOCH-2 | | | | | | | | | | | | |
| 1 | 1 | 1 | 1 | 0.7847 | 0.2153 | 0.0215 | 0.0215 | 0.0215 | 0.2837 | 0.3003 | 0.2654 | 0.046 |
| 1 | −1 | 1 | 1 | 0.2488 | 0.7512 | 0.7512 | −0.0751 | 0.0751 | 0.3588 | 0.2251 | 0.3405 | 0.564 |
| −1 | 1 | 1 | 1 | 0.2069 | 0.7931 | −0.7931 | 0.0793 | 0.0793 | 0.2795 | 0.3044 | 0.4198 | 0.629 |
| −1 | −1 | 1 | −1 | −0.1641 | −0.8359 | 0.0836 | 0.0836 | −0.0836 | 0.3631 | 0.388 | 0.336 | 0.699 |
| EPOCH-3 | | | | | | | | | | | | |
| 1 | 1 | 1 | 1 | 1.0873 | −0.0873 | −0.087 | −0.087 | −0.087 | 0.3543 | 0.3793 | 0.3275 | 0.0076 |
| 1 | −1 | 1 | 1 | 0.3025 | +0.6975 | 0.0697 | −0.0697 | 0.0697 | 0.4241 | 0.3096 | 0.3973 | 0.487 |
| −1 | 1 | 1 | 1 | 0.2827 | 0.7173 | −0.0717 | 0.0717 | 0.0717 | 0.3523 | 0.3813 | 0.469 | 0.515 |
| −1 | −1 | 1 | −1 | −0.2647 | −0.7353 | 0.0735 | 0.0735 | −0.0735 | 0.4259 | 0.4548 | 0.3954 | 0.541 |
| EPOCH-4 | | | | | | | | | | | | |
| 1 | 1 | 1 | 1 | 1.2761 | −0.2761 | −0.0276 | −0.0276 | −0.0276 | 0.3983 | 0.4272 | 0.3678 | 0.076 |
| 1 | −1 | 1 | 1 | 0.3389 | 0.6611 | 0.0661 | −0.0661 | 0.0661 | 0.4644 | 0.3611 | 0.4339 | 0.437 |
| −1 | 1 | 1 | 1 | 0.3307 | 0.6693 | −0.0669 | 0.0669 | 0.0699 | 0.3974 | 0.428 | 0.5009 | 0.448 |
| −1 | −1 | 1 | −1 | −0.3246 | −0.6754 | 0.0675 | 0.0675 | −0.0675 | 0.465 | 0.4956 | 0.4333 | 0.456 |
| EPOCH-5 | | | | | | | | | | | | |
| 1 | 1 | 1 | 1 | 1.3939 | −0.3939 | −0.0394 | −0.0394 | −0.0394 | 0.4256 | 0.4562 | 0.393 | 0.155 |
| 1 | −1 | 1 | 1 | 0.3634 | 0.6366 | 0.0637 | −0.0637 | 0.0637 | 0.4893 | 0.3925 | 0.457 | 0.405 |
| −1 | 1 | 1 | 1 | 0.3609 | 0.6391 | −0.0639 | 0.0639 | 0.0639 | 0.4253 | 0.4654 | 0.5215 | 0.408 |
| −1 | −1 | 1 | −1 | −0.3603 | −0.6397 | 0.064 | 0.064 | −0.064 | 0.4893 | 0.5204 | 0.4575 | 0.409 |

7. Use Adaline network to train ANDNOT function with bipolar inputs and targets. Perform 2 epochs of training.

**Solution:** The truth table for ANDNOT function with bipolar inputs and targets is shown in Table 13.

**Table 13**

| $x_1$ | $x_2$ | 1 | $t$ |
|---|---|---|---|
| 1 | 1 | 1 | −1 |
| 1 | −1 | 1 | 1 |
| −1 | 1 | 1 | −1 |
| −1 | −1 | 1 | −1 |

Initially the weights and bias have assumed a random value say 0.2. The learning rate is also set to 0.2. The weights are calculated until the least mean square error is obtained. The initial weights are $w_1 = w_2 = b = 0.2$, and $\alpha = 0.2$. For the first input sample $x_1 = 1$, $x_2 = 1$, $t = -1$, we calculate the net input as

$$y_{in} = b + x_1 w_1 + x_2 w_2$$
$$= 0.2 + 1 \times 0.2 + 1 \times 0.2 = 0.6$$

Now compute $(t - y_{in}) = (-1 - 0.6) = -1.6$. Updating the weights we obtain

$$w_i(\text{new}) = w_i(\text{old}) + \alpha(t - y_{in})x_i$$

The new weights are obtained as

$$w_1(\text{new}) = w_1(\text{old}) + \alpha(t - y_{in})x_1$$
$$= 0.2 + 0.2 \times (-1.6) \times 1 = -0.12$$

$$w_2(\text{new}) = w_2(\text{old}) + \alpha(t - y_{in})x_2$$
$$= 0.2 + 0.2 \times (-1.6) \times 1 = -0.12$$

$$b(\text{new}) = b(\text{old}) + \alpha(t - y_{in})$$
$$= 0.2 + 0.2 \times (-1.6) = -0.12$$

Now we compute the error,

$$E = (t - y_{in})^2 = (-1.6)^2 = 2.56$$

The final weights after presenting first input sample are $w = [-0.12 \; -0.12 \; -0.12]$ and error $E = 2.56$.

The operational steps are carried for 2 epochs of training and network performance is noted. It is tabulated as shown in Table 14.

The total mean square error at the end of two epochs is summation of the errors of all input samples as shown in Table 15.

**Table 15**

| Epoch | Total mean square error |
|---|---|
| Epoch 1 | 5.71 |
| Epoch 2 | 2.43 |

Hence from Table 15, it is clearly understood that the mean square error decreases as training progresses. Also, it can be noted that at the end of the sixth epoch, the error becomes approximately equal to 1. The network architecture for ANDNOT function using Adaline network is shown in Figure 9.

**Table 14**

| Inputs | | | Target | Net input | | Weight changes | | | Weights | | | Error |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $x_1$ | $x_2$ | 1 | $t$ | $y_{in}$ | $(t - y_{in})$ | $\Delta w_1$ | $\Delta w_2$ | $\Delta b$ | $w_1$ (0.2 | $w_2$ 0.2 | $b$ 0.2) | $(t - y_{in})^2$ |
| EPOCH-1 | | | | | | | | | | | | |
| 1 | 1 | 1 | −1 | 0.6 | −1.6 | −0.32 | −0.32 | −0.32 | −0.12 | −0.12 | −0.12 | 2.56 |
| 1 | −1 | 1 | 1 | −0.12 | 1.12 | 0.22 | −0.22 | 0.22 | 0.10 | −0.34 | 0.10 | 1.25 |
| −1 | 1 | 1 | −1 | −0.34 | −0.66 | 0.13 | −0.13 | −0.13 | 0.24 | −0.48 | −0.03 | 0.43 |
| −1 | −1 | 1 | −1 | 0.21 | −1.2 | 0.24 | 0.24 | −0.24 | 0.48 | −0.23 | −0.27 | 1.47 |
| EPOCH-2 | | | | | | | | | | | | |
| 1 | 1 | 1 | −1 | −0.02 | −0.98 | −0.195 | −0.195 | −0.195 | 0.28 | −0.43 | −0.46 | 0.95 |
| 1 | −1 | 1 | 1 | 0.25 | 0.76 | 0.15 | −0.15 | 0.15 | 0.43 | −0.58 | −0.31 | 0.57 |
| −1 | 1 | 1 | −1 | −1.33 | 0.33 | −0.065 | 0.065 | 0.065 | 0.37 | −0.51 | −0.25 | 0.106 |
| −1 | −1 | 1 | −1 | −0.11 | −0.90 | 0.18 | 0.18 | −0.18 | 0.55 | −0.38 | 0.43 | 0.8 |

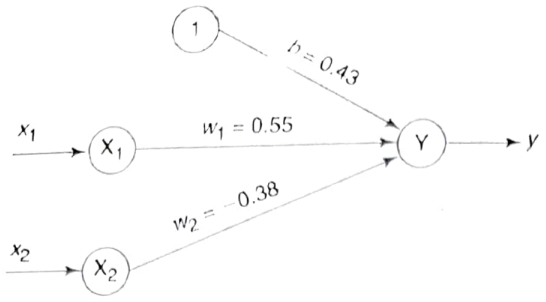


**Figure 9** Network architecture for ANDNOT function using Adaline network.

8 Using Madaline network, implement XOR function with bipolar inputs and targets. Assume the required parameters for training of the network.

**Solution:** The training pattern for XOR function is given in Table 16.

**Table 16**

| $x_1$ | $x_2$ | 1 | $t$ |
|---|---|---|---|
| 1 | 1 | 1 | −1 |
| 1 | −1 | 1 | 1 |
| −1 | 1 | 1 | 1 |
| −1 | −1 | 1 | −1 |

The Madaline Rule I (MRI) algorithm in which the weights between the hidden layer and output layer remain fixed is used for training the network. Initializing the weights to small random values, the network architecture is as shown in Figure 10, with initial weights. From Figure 10, the initial weights and bias are $[w_{11}\ w_{21}\ b_1] = [0.05\ 0.2\ 0.3]$, $[w_{12}\ w_{22}\ b_2] = [0.1\ 0.2\ 0.15]$ and $[v_1\ v_2\ b_3] = [0.5\ 0.5\ 0.5]$. For first

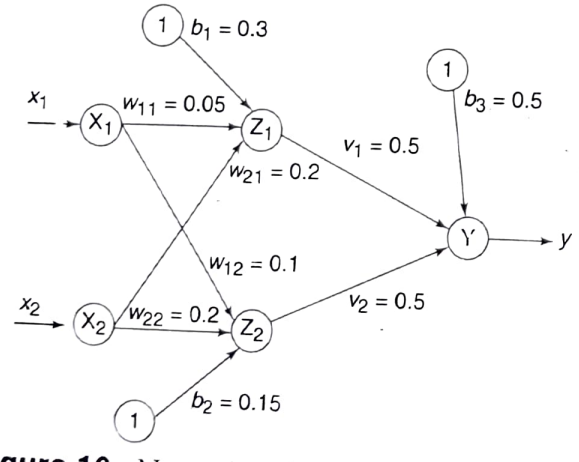


**Figure 10** Network architecture of Madaline for XOR functions (initial weights given).

input sample, $x_1 = 1$, $x_2 = 1$, target $t = -1$, and learning rate $\alpha$ equal to 0.5:

- Calculate net input to the hidden units:

$$z_{in1} = b_1 + x_1 w_{11} + x_2 w_{21}$$
$$= 0.3 + 1 \times 0.05 + 1 \times 0.2 = 0.55$$
$$z_{in2} = b_2 + x_1 w_{12} + x_2 w_{22}$$
$$= 0.15 + 1 \times 0.1 + 1 \times 0.2 = 0.45$$

- Calculate the output $z_1, z_2$ by applying the activations over the net input computed. The activation function is given by

$$f(z_{in}) = \begin{cases} 1 & \text{if } z_{in} \geq 0 \\ -1 & \text{if } z_{in} < 0 \end{cases}$$

Hence,

$$z_1 = f(z_{in1}) = f(0.55) = 1$$
$$z_2 = f(z_{in2}) = f(0.45) = 1$$

- After computing the output of the hidden units, then find the net input entering into the output unit:

$$y_{in} = b_3 + z_1 v_1 + z_2 v_2$$
$$= 0.5 + 1 \times 0.5 + 1 \times 0.5 = 1.5$$

- Apply the activation function over the net input $y_{in}$ to calculate the output $y$:

$$y = f(y_{in}) = f(1.5) = 1$$

- Since $t \neq y$, weight updation has to be performed. Also since $t = -1$, the weights are updated on $z_1$ and $z_2$ that have positive net input. Since here both net inputs $z_{in1}$ and $z_{in2}$ are positive, updating the weights and bias on both hidden units, we obtain

$$w_{ij}(\text{new}) = w_{ij}(\text{old}) + \alpha(t - z_{\text{in}j})x_i$$
$$b_j(\text{new}) = b_j(\text{old}) + \alpha(t - z_{\text{in}j})$$

This implies:

$$w_{11}(\text{new}) = w_{11}(\text{old}) + \alpha(t - z_{\text{in}1})x_1$$
$$= 0.05 + 0.5(-1 - 0.55) \times 1 = -0.725$$
$$w_{12}(\text{new}) = w_{12}(\text{old}) + \alpha(t - z_{\text{in}2})x_1$$
$$= 0.1 + 0.5(-1 - 0.45) \times 1 = -0.625$$
$$b_1(\text{new}) = b_1(\text{old}) + \alpha(t - z_{\text{in}1})$$
$$= 0.3 + 0.5(-1 - 0.55) = -0.475$$

$$w_{21}(\text{new}) = w_{21}(\text{old}) + \alpha(t - z_{in1})x_2$$
$$= 0.2 + 0.5(-1 - 0.55) \times 1 = -0.575$$
$$w_{22}(\text{new}) = w_{22}(\text{old}) + \alpha(t - z_{in2})x_2$$
$$= 0.2 + 0.5(-1 - 0.45) \times 1 = -0.525$$
$$b_2(\text{new}) = b_2(\text{old}) + \alpha(t - z_{in2})$$
$$= 0.15 + 0.5(-1 - 0.45) = -0.575$$

All the weights and bias between the input layer and hidden layer are adjusted. This completes the training for the first epoch. The same process is repeated until the weight converges. It is found that the weight converges at the end of 3 epochs. Table 17 shows the training performance of Madaline network for XOR function.

The network architecture for Madaline network with final weights for XOR function is shown in Figure 11.

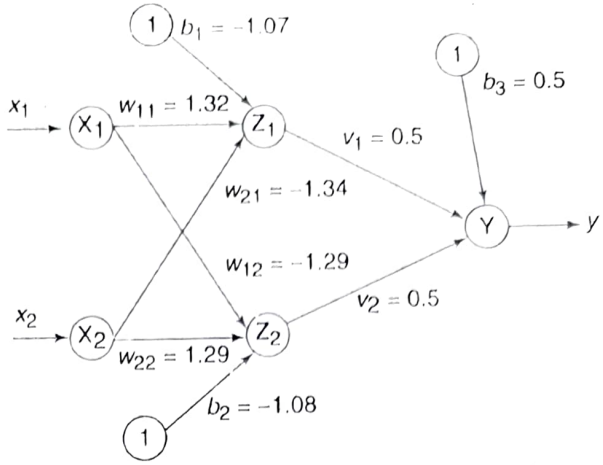


**Figure 11** Madaline network for XOR function (final weights given).

9. Using back-propagation network, find the new weights for the net shown in Figure 12. It is presented with the input pattern [0, 1] and the target output is 1. Use a learning rate $\alpha = 0.25$ and binary sigmoidal activation function.

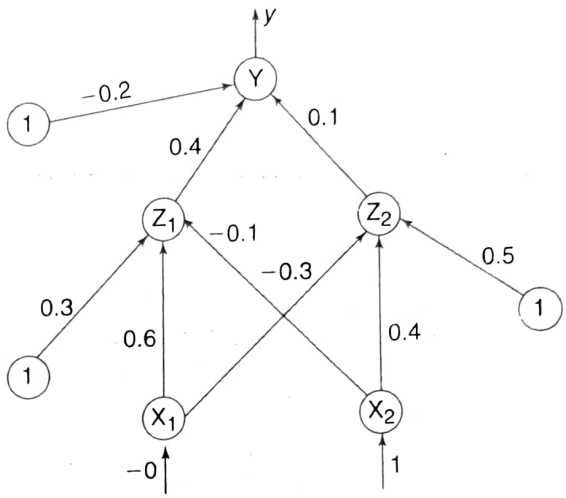


**Figure 12** Network.

**Solution:** The new weights are calculated based on the training algorithm in Section 3.5.4. The initial weights are $[v_{11}\ v_{21}\ v_{01}] = [0.6\ -0.1\ 0.3]$,

**Table 17**

| Inputs | | | Target | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $x_1$ | $x_2$ | 1 | $(t)$ | $z_{in1}$ | $z_{in2}$ | $z_1$ | $z_2$ | $y_{in}$ | $y$ | $w_{11}$ | $w_{21}$ | $b_1$ | $w_{12}$ | $w_{22}$ | $b_2$ |
| EPOCH-1 | | | | | | | | | | | | | | | |
| 1 | 1 | 1 | −1 | 0.55 | 0.45 | 1 | 1 | 1.5 | 1 | −0.725 | −0.58 | −0.475 | −0.625 | −0.525 | −0.575 |
| 1 | −1 | 1 | 1 | −0.625 | −0.675 | −1 | −1 | −0.5 | −1 | 0.0875 | −1.39 | 0.34 | −0.625 | −0.525 | −0.575 |
| −1 | 1 | 1 | 1 | −1.1375 | −0.475 | −1 | −1 | −0.5 | −1 | 0.0875 | −1.39 | 0.34 | −1.3625 | 0.2125 | 0.1625 |
| −1 | −1 | 1 | −1 | 1.6375 | 1.3125 | 1 | 1 | 1.5 | 1 | 1.4065 | −0.069 | −0.98 | −0.207 | 1.369 | −0.994 |
| EPOCH-2 | | | | | | | | | | | | | | | |
| 1 | 1 | 1 | −1 | 0.3565 | 0.168 | 1 | 1 | 1.5 | 1 | 0.7285 | −0.75 | −1.66 | −0.791 | −0.207 | −1.58 |
| 1 | −1 | 1 | 1 | −0.1845 | −3.154 | −1 | −1 | −0.5 | −1 | 1.3205 | −1.34 | −1.068 | −0.791 | 0.785 | −1.58 |
| −1 | 1 | 1 | 1 | −3.728 | −0.002 | −1 | −1 | −0.5 | −1 | 1.3205 | −1.34 | −1.068 | −1.29 | 0.785 | −1.08 |
| −1 | −1 | 1 | −1 | −1.0495 | −1.071 | −1 | −1 | −0.5 | −1 | 1.3205 | −1.34 | −1.068 | −1.29 | 1.29 | −1.08 |
| EPOCH-3 | | | | | | | | | | | | | | | |
| 1 | 1 | 1 | −1 | −1.0865 | −1.083 | −1 | −1 | −0.5 | −1 | 1.32 | −1.34 | −1.07 | −1.29 | 1.29 | −1.08 |
| 1 | −1 | 1 | 1 | 1.5915 | −3.655 | 1 | −1 | 0.5 | 1 | 1.32 | −1.34 | −1.07 | −1.29 | 1.29 | −1.08 |
| −1 | 1 | 1 | 1 | −3.728 | 1.501 | −1 | 1 | 0.5 | 1 | 1.32 | −1.34 | −1.07 | −1.29 | 1.29 | −1.08 |
| −1 | −1 | 1 | −1 | −1.0495 | −1.701 | −1 | −1 | −0.5 | −1 | 1.32 | −1.34 | −1.07 | −1.29 | 1.29 | −1.08 |

$[v_{12}\ v_{22}\ v_{02}] = [-0.3\ 0.4\ 0.5]$ and $[w_1\ w_2\ w_0] = [0.4\ 0.1\ -0.2]$, and the learning rate is $\alpha = 0.25$. Activation function used is binary sigmoidal activation function and is given by

$$f(x) = \frac{1}{1 + e^{-x}}$$

Given the output sample $[x_1, x_2] = [0, 1]$ and target $t = 1$,

- Calculate the net input: For $z_1$ layer

$$z_{in1} = v_{01} + x_1 v_{11} + x_2 v_{21}$$
$$= 0.3 + 0 \times 0.6 + 1 \times -0.1 = 0.2$$

For $z_2$ layer

$$z_{in2} = v_{02} + x_1 v_{12} + x_2 v_{22}$$
$$= 0.5 + 0 \times -0.3 + 1 \times 0.4 = 0.9$$

Applying activation to calculate the output, we obtain

$$z_1 = f(z_{in1}) = \frac{1}{1 + e^{-z_{in1}}} = \frac{1}{1 + e^{-0.2}} = 0.5498$$

$$z_2 = f(z_{in2}) = \frac{1}{1 + e^{-z_{in2}}} = \frac{1}{1 + e^{-0.9}} = 0.7109$$

- Calculate the net input entering the output layer. For $y$ layer

$$y_{in} = w_0 + z_1 w_1 + z_2 w_2$$
$$= -0.2 + 0.5498 \times 0.4 + 0.7109 \times 0.1$$
$$= 0.09101$$

Applying activations to calculate the output, we obtain

$$y = f(y_{in}) = \frac{1}{1 + e^{-y_{in}}} = \frac{1}{1 + e^{-0.09101}} = 0.5227$$

- Compute the error portion $\delta_k$:

$$\delta_k = (t_k - y_k) f'(y_{ink})$$

Now

$$f'(y_{in}) = f(y_{in})[1 - f(y_{in})] = 0.5227[1 - 0.5227]$$
$$f'(y_{in}) = 0.2495$$

This implies

$$\delta_1 = (1 - 0.5227)\ (0.2495) = 0.1191$$

Find the changes in weights between hidden and output layer:

$$\Delta w_1 = \alpha \delta_1 z_1 = 0.25 \times 0.1191 \times 0.5498$$
$$= 0.0164$$
$$\Delta w_2 = \alpha \delta_1 z_2 = 0.25 \times 0.1191 \times 0.7109$$
$$= 0.02117$$
$$\Delta w_0 = \alpha \delta_1 = 0.25 \times 0.1191 = 0.02978$$

- Compute the error portion $\delta_j$ between input and hidden layer ($j = 1$ to 2):

$$\delta_j = \delta_{inj} f'(z_{inj})$$

$$\delta_{inj} = \sum_{k=1}^{m} \delta_k w_{jk}$$

$$\delta_{inj} = \delta_1 w_{j1} \quad [\because \text{only one output neuron}]$$
$$\Rightarrow \delta_{in1} = \delta_1 w_{11} = 0.1191 \times 0.4 = 0.04764$$
$$\Rightarrow \delta_{in2} = \delta_1 w_{21} = 0.1191 \times 0.1 = 0.01191$$

Error, $\delta_1 = \delta_{in1} f'(z_{in1})$

$$f'(z_{in1}) = f(z_{in1})\ [1 - f(z_{in1})]$$
$$= 0.5498[1 - 0.5498] = 0.2475$$
$$\delta_1 = \delta_{in1} f'(z_{in1})$$
$$= 0.04764 \times 0.2475 = 0.0118$$

Error, $\delta_2 = \delta_{in2} f'(z_{in2})$

$$f'(z_{in2}) = f(z_{in2})\ [1 - f(z_{in2})]$$
$$= 0.7109[1 - 0.7109] = 0.2055$$
$$\delta_2 = \delta_{in2} f'(z_{in2})$$
$$= 0.01191 \times 0.2055 = 0.00245$$

Now find the changes in weights between input and hidden layer:

$$\Delta v_{11} = \alpha \delta_1 x_1 = 0.25 \times 0.0118 \times 0 = 0$$
$$\Delta v_{21} = \alpha \delta_1 x_2 = 0.25 \times 0.0118 \times 1 = 0.00295$$
$$\Delta v_{01} = \alpha \delta_1 = 0.25 \times 0.0118 = 0.00295$$
$$\Delta v_{12} = \alpha \delta_2 x_1 = 0.25 \times 0.00245 \times 0 = 0$$
$$\Delta v_{22} = \alpha \delta_2 x_2 = 0.25 \times 0.00245 \times 1 = 0.0006125$$
$$\Delta v_{02} = \alpha \delta_2 = 0.25 \times 0.00245 = 0.0006125$$

- Compute the final weights of the network:

$$v_{11}(\text{new}) = v_{11}(\text{old}) + \Delta v_{11} = 0.6 + 0 = 0.6$$
$$v_{12}(\text{new}) = v_{12}(\text{old}) + \Delta v_{12} = -0.3 + 0 = -0.3$$
$$v_{21}(\text{new}) = v_{21}(\text{old}) + \Delta v_{21}$$
$$= -0.1 + 0.00295 = -0.09705$$
$$v_{22}(\text{new}) = v_{22}(\text{old}) + \Delta v_{22}$$
$$= 0.4 + 0.0006125 = 0.4006125$$
$$w_1(\text{new}) = w_1(\text{old}) + \Delta w_1 = 0.4 + 0.0164$$
$$= 0.4164$$
$$w_2(\text{new}) = w_2(\text{old}) + \Delta w_2 = 0.1 + 0.02117$$
$$= 0.12117$$
$$v_{01}(\text{new}) = v_{01}(\text{old}) + \Delta v_{01} = 0.3 + 0.00295$$
$$= 0.30295$$
$$v_{02}(\text{new}) = v_{02}(\text{old}) + \Delta v_{02}$$
$$= 0.5 + 0.0006125 = 0.5006125$$
$$w_0(\text{new}) = w_0(\text{old}) + \Delta w_0 = -0.2 + 0.02978$$
$$= -0.17022$$

Thus, the final weights have been computed for the network shown in Figure 12.

10. Find the new weights, using back-propagation network for the network shown in Figure 13. The network is presented with the input pattern [−1, 1] and the target output is +1. Use a learning rate of $\alpha = 0.25$ and bipolar sigmoidal activation function.

**Solution:** The initial weights are $[v_{11}\, v_{21}\, v_{01}] = [0.6\ -0.1\ 0.3]$, $[v_{12}\ v_{22}\ v_{02}] = [-0.3\ 0.4\ 0.5]$ and $[w_1\ w_2\ w_0] = [0.4\ 0.1\ -0.2]$, and the learning rate is $\alpha = 0.25$.

Activation function used is binary sigmoidal activation function and is given by

$$f(x) = \frac{2}{1 + e^{-x}} - 1 = \frac{1 - e^{-x}}{1 + e^{-x}}$$

Given the input sample $[x_1, x_2] = [-1, 1]$ and target $t = 1$:

- Calculate the net input: For $z_1$ layer

$$z_{in1} = v_{01} + x_1 v_{11} + x_2 v_{21}$$
$$= 0.3 + (-1) \times 0.6 + 1 \times -0.1 = -0.4$$

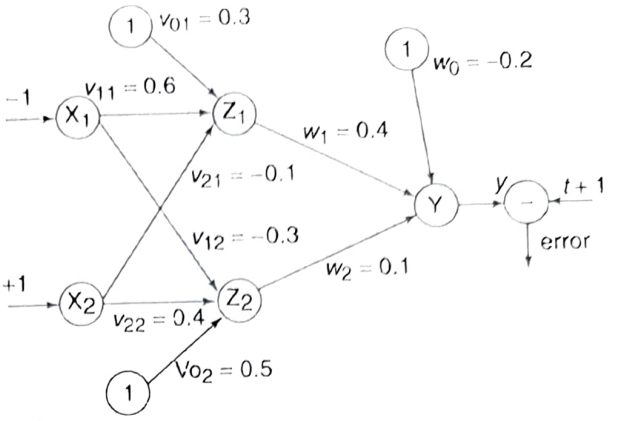


**Figure 13** Network.

For $z_2$ layer

$$z_{in2} = v_{02} + x_1 v_{12} + x_2 v_{22}$$
$$= 0.5 + (-1) \times -0.3 + 1 \times 0.4 = 1.2$$

Applying activation to calculate the output, we obtain

$$z_1 = f(z_{in1}) = \frac{1 - e^{-z_{in1}}}{1 + e^{-z_{in1}}} = \frac{1 - e^{0.4}}{1 + e^{0.4}} = -0.1974$$
$$z_2 = f(z_{in2}) = \frac{1 - e^{-z_{in1}}}{1 + e^{-z_{in2}}} = \frac{1 - e^{-1.2}}{1 + e^{-1.2}} = 0.537$$

- Calculate the net input entering the output layer. For $y$ layer

$$y_{in} = w_0 + z_1 w_1 + z_2 w_2$$
$$= -0.2 + (-0.1974) \times 0.4 + 0.537 \times 0.1$$
$$= -0.22526$$

Applying activations to calculate the output, we obtain

$$y = f(y_{in}) = \frac{1 - e^{-y_{in}}}{1 + e^{-y_{in}}} = \frac{1 - e^{0.22526}}{1 + e^{0.22526}} = -0.1122$$

- Compute the error portion $\delta_k$:

$$\delta_k = (t_k - y_k) f'(y_{ink})$$

Now

$$f'(y_{in}) = 0.5[1 + f(y_{in})]\,[1 - f(y_{in})]$$
$$= 0.5[1 - 0.1122][1 + 0.1122] = 0.4937$$

This implies

$$\delta_1 = (1 + 0.1122)\ (0.4937) = 0.5491$$

Find the changes in weights between hidden and output layer:

$$\Delta w_1 = \alpha \delta_1 z_1 = 0.25 \times 0.5491 \times -0.1974 = -0.0271$$

$$\Delta w_2 = \alpha \delta_1 z_2 = 0.25 \times 0.5491 \times 0.537 = 0.0737$$

$$\Delta w_0 = \alpha \delta_1 = 0.25 \times 0.5491 = 0.1373$$

- Compute the error portion $\delta_j$ between input and hidden layer ($j = 1$ to 2):

$$\delta_j = \delta_{inj} f'(z_{inj})$$

$$\delta_{inj} = \sum_{k=1}^{m} \delta_k w_{jk}$$

$$\delta_{inj} = \delta_1 w_{j1} \quad [\because \text{only one output neuron}]$$

$$\Rightarrow \delta_{in1} = \delta_1 w_{11} = 0.5491 \times 0.4 = 0.21964$$

$$\Rightarrow \delta_{in2} = \delta_1 w_{21} = 0.5491 \times 0.1 = 0.05491$$

$$\text{Error, } \delta_1 = \delta_{in1} f'(z_{in1}) = 0.21964 \times 0.5 \times (1 + 0.1974)(1 - 0.1974) = 0.1056$$

$$\text{Error, } \delta_2 = \delta_{in2} f'(z_{in2}) = 0.05491 \times 0.5 \times (1 - 0.537)(1 + 0.537) = 0.0195$$

Now find the changes in weights between input and hidden layer:

$$\Delta v_{11} = \alpha \delta_1 x_1 = 0.25 \times 0.1056 \times -1 = -0.0264$$

$$\Delta v_{21} = \alpha \delta_1 x_2 = 0.25 \times 0.1056 \times 1 = 0.0264$$

$$\Delta v_{01} = \alpha \delta_1 = 0.25 \times 0.1056 = 0.0264$$

$$\Delta v_{12} = \alpha \delta_2 x_1 = 0.25 \times 0.0195 \times -1 = -0.0049$$

$$\Delta v_{22} = \alpha \delta_2 x_2 = 0.25 \times 0.0195 \times 1 = 0.0049$$

$$\Delta v_{02} = \alpha \delta_2 = 0.25 \times 0.0195 = 0.0049$$

- Compute the final weights of the network:

$$v_{11}(\text{new}) = v_{11}(\text{old}) + \Delta v_{11} = 0.6 - 0.0264 = 0.5736$$

$$v_{12}(\text{new}) = v_{12}(\text{old}) + \Delta v_{12} = -0.3 - 0.0049 = -0.3049$$

$$v_{21}(\text{new}) = v_{21}(\text{old}) + \Delta v_{21} = -0.1 + 0.0264 = -0.0736$$

$$v_{22}(\text{new}) = v_{22}(\text{old}) + \Delta v_{22} = 0.4 + 0.0049 = 0.4049$$

$$w_1(\text{new}) = w_1(\text{old}) + \Delta w_1 = 0.4 - 0.0271 = 0.3729$$

$$w_2(\text{new}) = w_2(\text{old}) + \Delta w_2 = 0.1 + 0.0737 = 0.1737$$

$$v_{01}(\text{new}) = v_{01}(\text{old}) + \Delta v_{01} = 0.3 + 0.0264 = 0.3264$$

$$v_{02}(\text{new}) = v_{02}(\text{old}) + \Delta v_{02} = 0.5 + 0.0049 = 0.5049$$

$$w_0(\text{new}) = w_0(\text{old}) + \Delta w_0 = -0.2 + 0.1373 = -0.0627$$

Thus, the final weight has been computed for the network shown in Figure 13.

## 3.12 Review Questi