Perceptron Networks → These are single layer feed forward networks and are also called single perceptrons. Various perceptron networks were designed. However a simple perceptron network was discovered by Block in 1962.

The key points:

1. Consists of sensory unit (input), associator unit (hidden unit), response unit (output unit)
2. Sensory units are connected to associator unit with fixed weights having values 1, 0 or −1 which are assigned random.
3. Binary activation function is used.
4. Response unit has an activation of 1, 0 and −1. The binary step with fixed threshold θ is used as activation for associator.
5. output of perceptron is

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

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

6. The perceptron learning rule is used in the weight updation ~~on~~ between the association unit and response unit.
7. The error calculation is based on comparison of the values of targets with calculated outputs

8) weights are updated

$w_i(new) = w_i(old) + \alpha t x_i$

$b(new) = b(old) + \alpha t$

α is learning rate, t is target is +1 or -1.

Perceptron learning rule

learning signal is difference between the desired and actual response of neuron.

Consider finite n number of input training vectors. The target is either +1 or -1.

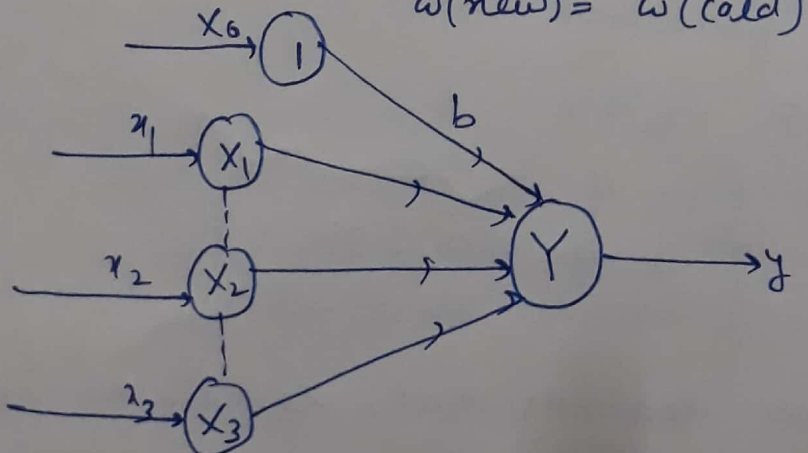
$$y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} > \theta \\ 0 & \text{if } -\theta \leq y_{in} \leq \theta \\ -1 & \text{if } y_{in} < -\theta \end{cases}$$

if $y \neq t$ then

$w(new) = w(old) + \alpha t x$ (α learning rate)

else we have

$w(new) = w(old)$



to form a Madaline network. These networks are trained using delta learning rule. Back-propagation network is the most commonly used network in the real time applications. The error is back-propagated here and is fine tuned for achieving better performance. The basic difference between the back-propagation network and radial basis function network is the activation function used. The radial basis function network mostly uses Gaussian activation function. Apart from these networks, some special supervised learning networks such as time delay neural networks, functional link networks, tree neural networks and wavelet neural networks have also been discussed.

3.12 Solved Problems

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 α is set equal to 1.

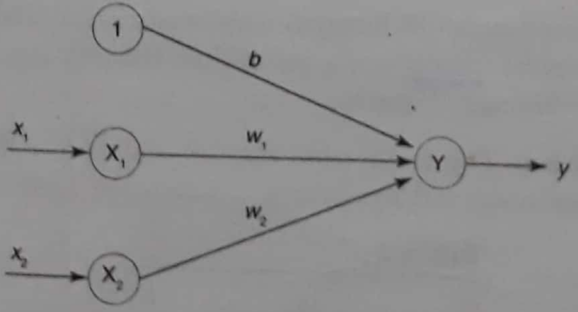


Figure 1 Perceptron network for AND function.

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

$$y_{in} = b + x_1w_1 + x_2w_2$$
$$= 0 + 1 \times 0 + 1 \times 0 = 0$$

- 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:

$$w_i(\text{new}) = w_i(\text{old}) + \alpha tx_i$$
$$w_1(\text{new}) = w_1(\text{old}) + \alpha tx_1 = 0 + 1 \times 1 \times 1 = 1$$
$$w_2(\text{new}) = w_2(\text{old}) + \alpha tx_2 = 0 + 1 \times 1 \times 1 = 1$$
$$b(\text{new}) = b(\text{old}) + \alpha t = 0 + 1 \times 1 = 1$$

Here, the change in weights are

$$\Delta w_1 = \alpha tx_1;$$
$$\Delta w_2 = \alpha tx_2;$$
$$\Delta b = \alpha t$$

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

$w_1, w_2 = 0$
$b = 0$

$y_{in} > 0$
$y_{in} < 0 \rightarrow -1$
$y_{in} = 0$ 0

Table 2

Input			Target	Net input	Calculated output	Weight changes			Weights		
x_1	x_2	1	$\boxed{(t)}$	(y_{in})	$\boxed{(y)}$	Δw_1	Δw_2	Δb	w_1 (0	w_2 0	b 0)
					$y = f[y_{in}]$						
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.

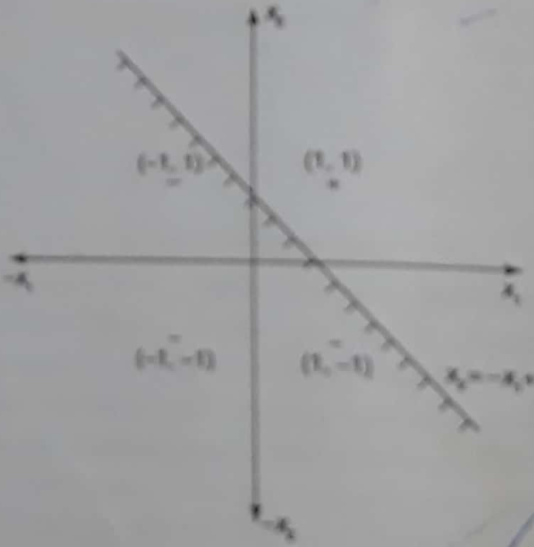


Figure 2 Decision boundary for AND function in perceptron training ($\theta = 0$).

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

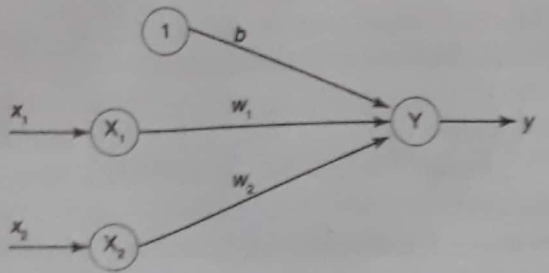


Figure 3 Perceptron network for OR function.

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		
									w_1	w_2	b
x_1	x_2	1	(t)	(y_{in})	(y)	Δw_1	Δw_2	Δb	(0	0	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	1
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	−1	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

ingle layer

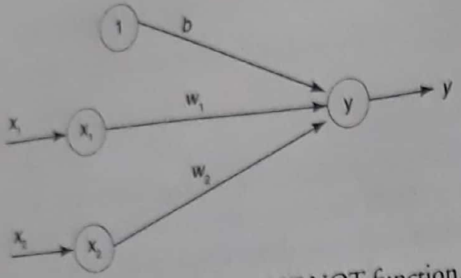


Figure 4 Network for ANDNOT function.

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

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

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	Weights		
x_1	x_2	1	(t)	(y_{in})	output (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

ith [illegible] outputs