## Example:

Let us first consider the operation AND

**Table 6.3** Example of perceptron learning: the logical operation AND

| | Inputs | | Desired output | Initial weights | | Actual output | Error | Final weights | |
|---|---|---|---|---|---|---|---|---|---|
| Epoch | $x_1$ | $x_2$ | $Y_d$ | $w_1$ | $w_2$ | $Y$ | $e$ | $w_1$ | $w_2$ |
| 1 | 0 | 0 | 0 | 0.3 | −0.1 | 0 | 0 | 0.3 | −0.1 |
| | 0 | 1 | 0 | 0.3 | −0.1 | 0 | 0 | 0.3 | −0.1 |
| | 1 | 0 | 0 | 0.3 | −0.1 | 1 | −1 | 0.2 | −0.1 |
| | 1 | 1 | 1 | 0.2 | −0.1 | 0 | 1 | 0.3 | 0.0 |
| 2 | 0 | 0 | 0 | 0.3 | 0.0 | 0 | 0 | 0.3 | 0.0 |
| | 0 | 1 | 0 | 0.3 | 0.0 | 0 | 0 | 0.3 | 0.0 |
| | 1 | 0 | 0 | 0.3 | 0.0 | 1 | −1 | 0.2 | 0.0 |
| | 1 | 1 | 1 | 0.2 | 0.0 | 1 | 0 | 0.2 | 0.0 |
| 3 | 0 | 0 | 0 | 0.2 | 0.0 | 0 | 0 | 0.2 | 0.0 |
| | 0 | 1 | 0 | 0.2 | 0.0 | 0 | 0 | 0.2 | 0.0 |
| | 1 | 0 | 0 | 0.2 | 0.0 | 1 | −1 | 0.1 | 0.0 |
| | 1 | 1 | 1 | 0.1 | 0.0 | 0 | 1 | 0.2 | 0.1 |
| 4 | 0 | 0 | 0 | 0.2 | 0.1 | 0 | 0 | 0.2 | 0.1 |
| | 0 | 1 | 0 | 0.2 | 0.1 | 0 | 0 | 0.2 | 0.1 |
| | 1 | 0 | 0 | 0.2 | 0.1 | 1 | −1 | 0.1 | 0.1 |
| | 1 | 1 | 1 | 0.1 | 0.1 | 1 | 0 | 0.1 | 0.1 |
| 5 | 0 | 0 | 0 | 0.1 | 0.1 | 0 | 0 | 0.1 | 0.1 |
| | 0 | 1 | 0 | 0.1 | 0.1 | 0 | 0 | 0.1 | 0.1 |
| | 1 | 0 | 0 | 0.1 | 0.1 | 0 | 0 | 0.1 | 0.1 |
| | 1 | 1 | 1 | 0.1 | 0.1 | 1 | 0 | 0.1 | 0.1 |

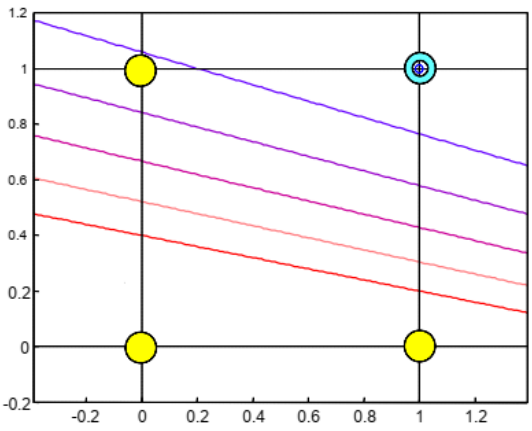
Threshold: $\theta = 0.2$; learning rate: $\alpha = 0.1$.

## Learning Algorithm: Training Perceptron Networks (cont.)

Function AND

| $x_1$ | $x_2$ | $y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

epoc: 16
epoc: 12
epoc: 8
epoc: 4
Initial




Initial Values

$w_1 = 0.5$

$w_2 = 2.5$

$\theta$ (b) = 1.0

$\alpha = 0.2$

## Perceptron Learning Rule depends on initial values

$\alpha * (t - o)$

$\sum X_i * W_i$

| | Perceptron Learning Example: AND Gate | | | | | | | | | Bias: Let X0 = 1 | Alpha = 0.5 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Input | | | | | | Output | | | Alpha = | 0.5 | | |
| | X0 | X1 | X2 | 1.0 * W0 | X1* W1 | X2* W2 | Total | target | actual | Alpha * (t-o) | W0 | W1 | W2 |
| | | | | | | | | | | | 0.1 | 0.1 | 0.1 |
| 1 | 1 | 0 | 0 | 0.10 | 0.00 | 0.00 | 0.10 | 0 | 1 | -0.50 | -0.40 | 0.10 | 0.10 |
| | 1 | 0 | 1 | -0.40 | 0.00 | 0.10 | -0.30 | 0 | 0 | 0.00 | -0.40 | 0.10 | 0.10 |
| | 1 | 1 | 0 | -0.40 | 0.10 | 0.00 | -0.30 | 0 | 0 | 0.00 | -0.40 | 0.10 | 0.10 |
| | 1 | 1 | 1 | -0.40 | 0.10 | 0.10 | -0.20 | 1 | 0 | 0.50 | 0.10 | 0.60 | 0.60 |
| 2 | 1 | 0 | 0 | 0.10 | 0.00 | 0.00 | 0.10 | 0 | 1 | -0.50 | -0.40 | 0.60 | 0.60 |
| | 1 | 0 | 1 | -0.40 | 0.00 | 0.60 | 0.20 | 0 | 1 | -0.50 | -0.90 | 0.60 | 0.10 |
| | 1 | 1 | 0 | -0.90 | 0.60 | 0.00 | -0.30 | 0 | 0 | 0.00 | -0.90 | 0.60 | 0.10 |
| | 1 | 1 | 1 | -0.90 | 0.60 | 0.10 | -0.20 | 1 | 0 | 0.50 | -0.40 | 1.10 | 0.60 |
| 3 | 1 | 0 | 0 | -0.40 | 0.00 | 0.00 | -0.40 | 0 | 0 | 0.00 | -0.40 | 1.10 | 0.60 |
| | 1 | 0 | 1 | -0.40 | 0.00 | 0.60 | 0.20 | 0 | 1 | -0.50 | -0.90 | 1.10 | 0.10 |
| | 1 | 1 | 0 | -0.90 | 1.10 | 0.00 | 0.20 | 0 | 1 | -0.50 | -1.40 | 0.60 | 0.10 |
| | 1 | 1 | 1 | -1.40 | 0.60 | 0.10 | -0.70 | 1 | 0 | 0.50 | -0.90 | 1.10 | 0.60 |

The perceptron model

$$g(t)=f\left[\sum_{i=1}^{n} x_i(t)w_i(t)-\theta(t)\right]$$

$w_i(t+1) = w_i(t) + \Delta w_i(t)$
$\Delta w_i(t) = \eta\, e(t)\, x_i(t)$
$e(t) = y(t) - g(t)$

where:
{x,y} are the training set
$w_i(t)$ are the perceptron weights
g(t) is the perceptron output at the time t (or step t). g(t) can be 0 or 1
e(t) is the error between the desired output y(t) and the perceptron real output g(t)
$\eta$ is the learning rate with $0<\eta\leq 1$.

Example:
We want a perceptron to tell the difference between a bird and a non-bird animal.
The training set is made up by these animals: eagle, ostrich (birds) and bat, horse, fish (non-birds).

| Animal | $x_1$ | $x_2$ | $x_3$ | $x_4$ | y |
|---|---|---|---|---|---|
| Eagle | 1 | 1 | 1 | 1 | 1 |
| Ostrich | 0 | 1 | 1 | 1 | 1 |
| Bat | 1 | 0 | 1 | 1 | 0 |
| Horse | 0 | 0 | 0 | 1 | 0 |
| Fish | 0 | 1 | 0 | 1 | 0 |

Where:
$x_1$ is the first input which refers to the flight characteristic (1 fly, 0 don't fly)
$x_2$ is the second input which regards the egg-laying apparatus (1 yes, 0 no)
$x_3$ is the third input which refers to the question: “has it got wings?” (1 yes, 0 no)
$x_4$ is an imaginary input (whose value is always 1) which is used to remove the threshold $\theta$. In this case $w_4 = -\theta$.
We assume $\eta = 1$, so $\Delta w_i(t) = \eta\, e(t)\, x_i(t) = e(t)\, x_i(t)$.

Let's start the perceptron training.

Step 0:
Initializing the weights $w_1$, $w_2$, $w_3$, $w_4$ using random values.uri
$w_1(0)=0$, $w_2(0)=1$, $w_3(0)=1$, $w_4(0)=0$

Step 1:

| $w_1$ | $w_2$ | $w_3$ | $w_4$ |
|---|---|---|---|
| 0 | 1 | 1 | 0 |

Showing the first animal (eagle) to the perceptron:
$x_1(1)w_1(1) + x_2(1)w_2(1) + x_3(1)w_3(1) + x_4(1)w_4(1) = 1*0 + 1*1 + 1*1 + 1*0 = 2 > 0$
So, $g(1) = 1$
$e(1) = y(1) – g(1) = 1 – 1 = 0$
In this case $\Delta w_i(1) = e(1)\, x_i(1) = 0$
The weights don't change.

Step 2:

| $w_1$ | $w_2$ | $w_3$ | $w_4$ |
|---|---|---|---|
| 0 | 1 | 1 | 0 |

Showing the second animal (ostrich) to the perceptron:
$x_1(2)w_1(2) + x_2(2)w_2(2) + x_3(2)w_3(2) + x_4(2)w_4(2) = 0*0 + 1*1 + 1*1 + 1*0 = 2 > 0$
So, $g(2) = 1$
$e(2) = y(2) – g(2) = 1 – 1 = 0$
In this case $\Delta w_i(2) = e(2)\, x_i = 0$
The weights don't change.

Step 3:

| $w_1$ | $w_2$ | $w_3$ | $w_4$ |
|---|---|---|---|
| 0 | 1 | 1 | 0 |

Showing the third animal (bat) to the perceptron:
$x_1(3)w_1(3) + x_2(3)w_2(3) + x_3(3)w_3(3) + x_4(3)w_4(3) = 1*0 + 0*1 + 1*1 + 1*0 = 1 > 0$
So, $g(3) = 1$
$e(3) = y(3) – g(3) = 0 – 1 = -1$
In this case the weights must change.
$\Delta w_1(3) = e(3)\, x_1 = -1*1 = -1$
$\Delta w_2(3) = e(3)\, x_2 = -1*0 = 0$
$\Delta w_3(3) = e(3)\, x_3 = -1*1 = -1$
$\Delta w_4(3) = e(3)\, x_4 = -1*1 = -1$
So, according to the delta rule, new weights are:
$w_1(4) = w_1(3) + \Delta w_1(3) = 0 + (-1) = -1$
$w_2(4) = w_2(3) + \Delta w_2(3) = 1 + 0 = 1$
$w_3(4) = w_3(3) + \Delta w_3(3) = 1 + (-1) = 0$
$w_4(4) = w_4(3) + \Delta w_4(3) = 0 + (-1) = -1$

Step 4:

| $w_1$ | $w_2$ | $w_3$ | $w_4$ |
|---|---|---|---|
| -1 | 1 | 0 | -1 |

Showing the fourth animal (horse) to the perceptron:
$x_1(4)w_1(4) + x_2(4)w_2(4) + x_3(4)w_3(4) + x_4(4)w_4(4) = 0*-1 + 0*1 + 0*0 + 1*-1 = -1 \leq 0$
So, $g(4) = 0$
$e(4) = y(4) – g(4) = 0 – 0 = 0$
The weights don't change.

Step 5:

| $w_1$ | $w_2$ | $w_3$ | $w_4$ |
|---|---|---|---|
| -1 | 1 | 0 | -1 |

Showing the last animal (fish) to the perceptron:
$x_1(5)w_1(5) + x_2(5)w_2(5) + x_3(5)w_3(5) + x_4(4)w_4(4) = 0*-1 + 1*1 + 0*0 + 1*-1 = 0 \leq 0$
So, $g(5) = 0$
$e(5) = y – g(5) = 0 – 0 = 0$
The weights don't change.
After showing the last sample we say that one learning cycle (epoch) is finished.

Step 6:

| $w_1$ | $w_2$ | $w_3$ | $w_4$ |
|---|---|---|---|
| -1 | 1 | 0 | -1 |

Showing the first animal (eagle) to the perceptron for the second time (second epoch):
$x_1(6)w_1(6) + x_2(6)w_2(6) + x_3(6)w_3(6) + x_4(6)w_4(6) = 1*-1 + 1*1 + 1*0 + 1*-1 = -1 \leq 0$
So, $g(6) = 0$
$e(6) = y(6) – g(6) = 1 – 0 = 1$
In this case the weights must change.
$\Delta w_1(5) = e(6)\, x_1 = 1*1 = 1$
$\Delta w_2(6) = e(6)\, x_2 = 1*1 = 1$
$\Delta w_3(6) = e(6)\, x_3 = 1*1 = 1$
$\Delta w_4(6) = e(6)\, x_4 = 1*1 = 1$
So, according to the delta rule, new weights are:
$w_1(7) = w_1(6) + \Delta w_1(6) = (-1) + 1 = 0$
$w_2(7) = w_2(6) + \Delta w_2(6) = 1 + 1 = 2$
$w_3(7) = w_3(6) + \Delta w_3(6) = 0 + 1 = 1$
$w_4(7) = w_4(6) + \Delta w_4(6) = (-1) + 1 = 0$

The learning process will continue in this manner until the weights are good for all the samples.
When the error is null for the whole training set, the learning process is over.
After four epochs, the weights that settle the whole learning set are:

| $w_1$ | $w_2$ | $w_3$ | $w_4$ |
|---|---|---|---|
| 0 | 2 | 1 | -2 |

Training Set

$$\left\{\mathbf{p}_1 = \begin{bmatrix}-1\\1\\-1\end{bmatrix}, t_1 = \begin{bmatrix}1\end{bmatrix}\right\} \qquad \left\{\mathbf{p}_2 = \begin{bmatrix}1\\1\\-1\end{bmatrix}, t_2 = \begin{bmatrix}0\end{bmatrix}\right\}$$

Initial Weights

$$\mathbf{W} = \begin{bmatrix}0.5 & -1 & -0.5\end{bmatrix} \qquad b = 0.5$$

First Iteration

$$a = hardlim(\mathbf{W}\mathbf{p}_1 + b) = hardlim\left(\begin{bmatrix}0.5 & -1 & -0.5\end{bmatrix}\begin{bmatrix}-1\\1\\-1\end{bmatrix} + 0.5\right)$$

$$a = hardlim(-0.5) = 0 \qquad e = t_1 - a = 1 - 0 = 1$$

$$\mathbf{W}^{new} = \mathbf{W}^{old} + e\mathbf{p}^T = \begin{bmatrix}0.5 & -1 & -0.5\end{bmatrix} + (1)\begin{bmatrix}-1 & 1 & -1\end{bmatrix} = \begin{bmatrix}-0.5 & 0 & -1.5\end{bmatrix}$$

$$b^{new} = b^{old} + e = 0.5 + (1) = 1.5$$

27

## Second Iteration

$$a = hardlim\,(\mathbf{W}\mathbf{p}_2 + b) = hardlim\,(\begin{bmatrix}-0.5 & 0 & -1.5\end{bmatrix}\begin{bmatrix}1\\1\\-1\end{bmatrix} + (1.5))$$

$$a = hardlim\,(2.5) = 1$$

$$e = t_2 - a = 0 - 1 = -1$$

$$\mathbf{W}^{new} = \mathbf{W}^{old} + e\mathbf{p}^T = \begin{bmatrix}-0.5 & 0 & -1.5\end{bmatrix} + (-1)\begin{bmatrix}1 & 1 & -1\end{bmatrix} = \begin{bmatrix}-1.5 & -1 & -0.5\end{bmatrix}$$

$$b^{new} = b^{old} + e = 1.5 + (-1) = 0.5$$

## Check

$$a = hardlim\,(\mathbf{W}\mathbf{p}_1 + b) = hardlim\,(\begin{bmatrix}-1.5 & -1 & -0.5\end{bmatrix}\begin{bmatrix}-1\\1\\-1\end{bmatrix} + 0.5)$$

$$a = hardlim\,(1.5) = 1 = t_1$$

$$a = hardlim\,(\mathbf{W}\mathbf{p}_2 + b) = hardlim\,(\begin{bmatrix}-1.5 & -1 & -0.5\end{bmatrix}\begin{bmatrix}1\\1\\-1\end{bmatrix} + 0.5)$$

$$a = hardlim\,(-1.5) = 0 = t_2$$

**P4.4 Solve the following classification problem with the perceptron rule. Apply each input vector in order, for as many repetitions as it takes to ensure that the problem is solved. Draw a graph of the problem only after you have found a solution.**

$$\left\{\mathbf{p}_1 = \begin{bmatrix}2\\2\end{bmatrix}, t_1 = 0\right\} \left\{\mathbf{p}_2 = \begin{bmatrix}1\\-2\end{bmatrix}, t_2 = 1\right\} \left\{\mathbf{p}_3 = \begin{bmatrix}-2\\2\end{bmatrix}, t_3 = 0\right\} \left\{\mathbf{p}_4 = \begin{bmatrix}-1\\1\end{bmatrix}, t_4 = 1\right\}$$

**Use the initial weights and bias:**

$$\mathbf{W}(0) = \begin{bmatrix}0 & 0\end{bmatrix} \qquad b(0) = 0 \ .$$

We start by calculating the perceptron's output $a$ for the first input vector $\mathbf{p}_1$, using the initial weights and bias.

$$a = hardlim(\mathbf{W}(0)\mathbf{p}_1 + b(0))$$

$$= hardlim\left(\begin{bmatrix}0 & 0\end{bmatrix}\begin{bmatrix}2\\2\end{bmatrix} + 0\right) = hardlim(0) = 1$$

The output $a$ does not equal the target value $t_1$, so we use the perceptron rule to find new weights and biases based on the error.

$$e = t_1 - a = 0 - 1 = -1$$
$$\mathbf{W}(1) = \mathbf{W}(0) + e\mathbf{p}_1^T = \begin{bmatrix}0 & 0\end{bmatrix} + (-1)\begin{bmatrix}2 & 2\end{bmatrix} = \begin{bmatrix}-2 & -2\end{bmatrix}$$
$$b(1) = b(0) + e = 0 + (-1) = -1$$

We now apply the second input vector $\mathbf{p}_2$, using the updated weights and bias.

$$a = hardlim(\mathbf{W}(1)\mathbf{p}_2 + b(1))$$

$$= hardlim\left(\begin{bmatrix}-2 & -2\end{bmatrix}\begin{bmatrix}1\\-2\end{bmatrix} - 1\right) = hardlim(1) = 1$$

This time the output $a$ is equal to the target $t_2$. Application of the perceptron rule will not result in any changes.

$$\mathbf{W}(2) = \mathbf{W}(1)$$
$$b(2) = b(1)$$

We now apply the third input vector.

$$a = hardlim(\mathbf{W}(2)\mathbf{p}_3 + b(2))$$

$$= hardlim\left(\begin{bmatrix}-2 & -2\end{bmatrix}\begin{bmatrix}-2\\2\end{bmatrix} - 1\right) = hardlim(-1) = 0$$

The output in response to input vector $\mathbf{p}_3$ is equal to the target $t_3$, so there will be no changes.

$$\mathbf{W}(3) = \mathbf{W}(2)$$
$$b(3) = b(2)$$

We now move on to the last input vector $\mathbf{p}_4$.

$$a = hardlim(\mathbf{W}(3)\mathbf{p}_4 + b(3))$$

$$= hardlim\left(\begin{bmatrix}-2 & -2\end{bmatrix}\begin{bmatrix}-1\\1\end{bmatrix} - 1\right) = hardlim(-1) = 0$$

This time the output $a$ does not equal the appropriate target $t_4$. The perceptron rule will result in a new set of values for $\mathbf{W}$ and $b$.

$$e = t_4 - a = 1 - 0 = 1$$
$$\mathbf{W}(4) = \mathbf{W}(3) + e\mathbf{p}_4^T = \begin{bmatrix}-2 & -2\end{bmatrix} + (1)\begin{bmatrix}-1 & 1\end{bmatrix} = \begin{bmatrix}-3 & -1\end{bmatrix}$$
$$b(4) = b(3) + e = -1 + 1 = 0$$

We now must check the first vector $\mathbf{p}_1$ again. This time the output $a$ is equal to the associated target $t_1$.

$$a = hardlim(\mathbf{W}(4)\mathbf{p}_1 + b(4))$$

$$= hardlim\left(\begin{bmatrix}-3 & -1\end{bmatrix}\begin{bmatrix}2\\2\end{bmatrix} + 0\right) = hardlim(-8) = 0$$

Therefore there are no changes.

$$\mathbf{W}(5) = \mathbf{W}(4)$$
$$b(5) = b(4)$$

The second presentation of $\mathbf{p}_2$ results in an error and therefore a new set of weight and bias values.

$$a = hardlim(\mathbf{W}(5)\mathbf{p}_2 + b(5))$$

$$= hardlim\left(\begin{bmatrix}-3 & -1\end{bmatrix}\begin{bmatrix}1\\-2\end{bmatrix} + 0\right) = hardlim(-1) = 0$$

Here are those new values:

$$e = t_2 - a = 1 - 0 = 1$$
$$\mathbf{W}(6) = \mathbf{W}(5) + e\mathbf{p}_2^T = \begin{bmatrix}-3 & -1\end{bmatrix} + (1)\begin{bmatrix}1 & -2\end{bmatrix} = \begin{bmatrix}-2 & -3\end{bmatrix}$$
$$b(6) = b(5) + e = 0 + 1 = 1.$$

Cycling through each input vector once more results in no errors.

$$a = hardlim(\mathbf{W}(6)\mathbf{p}_3 + b(6)) = hardlim\left(\begin{bmatrix}-2 & -3\end{bmatrix}\begin{bmatrix}-2\\2\end{bmatrix} + 1\right) = 0 = t_3$$

$$a = hardlim(\mathbf{W}(6)\mathbf{p}_4 + b(6)) = hardlim\left(\begin{bmatrix}-2 & -3\end{bmatrix}\begin{bmatrix}-1\\1\end{bmatrix} + 1\right) = 1 = t_4$$

$$a = hardlim(\mathbf{W}(6)\mathbf{p}_1 + b(6)) = hardlim\left(\begin{bmatrix}-2 & -3\end{bmatrix}\begin{bmatrix}2\\2\end{bmatrix} + 1\right) = 0 = t_1$$

$$a = hardlim(\mathbf{W}(6)\mathbf{p}_2 + b(6)) = hardlim\left(\begin{bmatrix}-2 & -3\end{bmatrix}\begin{bmatrix}1\\-2\end{bmatrix} + 1\right) = 1 = t_2$$

Therefore the algorithm has converged. The final solution is:

$$\mathbf{W} = \begin{bmatrix}-2 & -3\end{bmatrix} \qquad b = 1.$$

**Procedure:**

Consider a one neuron perceptron with a single vector input having two elements. Suppose that we have the following training pattern,

$$\left\{p_1 = \begin{bmatrix}2\\2\end{bmatrix}, t_1 = 0\right\} \left\{p_2 = \begin{bmatrix}1\\-2\end{bmatrix}, t_2 = 1\right\} \left\{p_3 = \begin{bmatrix}-2\\2\end{bmatrix}, t_3 = 0\right\} \left\{p_4 = \begin{bmatrix}-1\\1\end{bmatrix}, t_4 = 1\right.$$

1- Initialize a new perceptron neuron with the above specifications

```
net = newp([-2 2;-2 +2],1);
p = [[2;2] [1;-2] [-2;2] [-1;1]];
t =[0 1 0 1];
```

2- Reset (initialize) the weights and bias for the pervious neuron

```
net = init(net);
wts = net.IW{1,1}
bias = net.b{1}
```

3- Redefine the network input weights and bias using random function

```
net.inputweights{1,1}.initFcn = 'rands';
net.biases{1}.initFcn = 'rands';
net = init(net);
wts = net.IW{1,1}
bias = net.b{1}
```

4- Set the bias to 0 and the weights to 1 and -0.8.

```
w = [1 -0.8];
net.b{1} = [0];
net.IW{1,1} = w;
```

5- Display the output of the network (simulate the network for each input)

```
a = sim(net, p)
```

6- Calculate the value of the error

```
e = t - a
```

7- Find the change in weights and bias using the function **learnp**

```
dw = learnp(w,p,[],[],[],[],e,[],[],[],[],[])
```

8- Update the values of the weights

```
w = w + dw
```

9- Set epochs to 1 and train the network; displays the new weights and output

```
net.trainParam.epochs = 1;
net = train(net,p,t);
```

10- Repeat step (9) by setting epochs to 4

```
net.trainParam.epochs = 4;
net = train(net,p,t);
```

# How a single layer perceptron works (cont.)

❑Example: AND Function

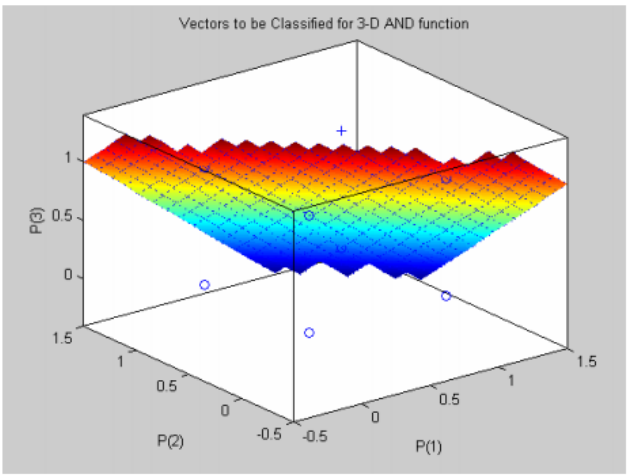
```
% Run_AND_Perceptron.m
P = [0 0 1 1; 0 1 0 1];  % AND Function
T = [0 0 0 1];
plotpv(P,T,[-1, 2, -1, 2]); % plot data
% initial weight vector and bias
W = [1 1]; b = 1;
plotpc(W,b);        % plot line
epoc = 1;     % number of epoc
for j=1:epoc
   for i=1:size(P,2)
      p = P(:,i);
      t = T(i);
      [W b] = perceptron(p',t,W,b);
      plotpc(W,b);
   end
end
% test data test = [0 0]'
test = [0 0];
output = hardlim(W*test'+b)
```

# AND Function by using MATLAB toolbox

```
net = newp([0 1; 0 1],1);   %This code creates a perceptron layer with one  2-element
                            % input (ranges [0 1] and [0 1]) and one neuron.
                            % Initial weight [0 0] and bias 0

P = [0 0 1 1; 0 1 0 1];      % And Input patterns
T = [0 0 0 1];               % Target

net.IW{1} = [1 1]            % display weight vector
net.b{1} = 1                 % display bias

net.trainParam.epochs = 2;   % train (learn) for a maximum of  2 epochs
net = train(net,P,T);

a = sim(net,P);              % simulate the network's output again (test pattern)
a = [0 1 1 1]                % Result from test pattern
```

# 3-D AND Function by using MATLAB toolbox

```
% 3D perceptron
net = newp([0 1; 0 1; 0 1],1);
P = [0 0 0; 0 0 1; 0 1 0; 0 1 1; 1 0 0; 1 0 1; 1 1 0; 1 1 1]';  % AND Function
T = [0; 0; 0; 0; 0; 0; 0; 1]';
% initial weight and bias
net.IW{1} = [1 1 1]; net.b{1} = 0;
% train or learning
net.trainParam.epochs = 1;
net = train(net,P,T);
plotpv(P,T);
plotpc(net.IW{1},net.b{1});
```




# OR Function by using MATLAB

```
net = newp([0 1; 0 1],1);   %This code creates a perceptron layer with one  2-element
                            % input (ranges [0 1] and [0 1]) and one neuron.
                            % Initial weight [0 0] and bias 0

P = [0 0 1 1; 0 1 0 1];      % OR Input patterns
T = [0 1 1 1];               % Target

net.IW{1} = [1 1]            % display weight vector
net.b{1} = 1                 % display bias

net.trainParam.epochs = 2;   % train (learn) for a maximum of  2 epochs
net = train(net,P,T);

a = sim(net,P);              % simulate the network's output again (test pattern)
a = [? ? ? ?]                % Result from test pattern
```