8.5 HEBB NET

A simple learning rule for a neural net is Hebb rule which can be formulated as follows.

Hebb learning rule

If two interconnected neurons fire at the same time, the weight associated with their connection link should be increased.

A stronger form of learning is obtained if the weight is increased also in the case when both neurons do not fire at the same time. We thus get

The extended Hebb rule

If two interconnected neurons fire or do not fire at the same time, the weight associated with their connection link is increased.

A single-layer which is trained using the extended Hebb rule is called a *Hebb net.* If we apply bipolar activations, a possible formulation to the extended Hebb rule for a Hebb net is

$$w_i(new) = w_i(old) + x_i y \tag{8.5.1}$$

where x_i is the activation of an input unit X_i, $y-$ the activation of an output unit Y and w_i is the weight associated with the connection link between X_i and Y. In order to include bias we add the connection link between Y and an input unit B with constant activation 1, which is associated with the weight b.

The right-hand side of Eq. (8.5.1) implies that should X_i (or B) and Y fire and not fire alternately, the associated weight must be *decreased.*

The simplest form of using Hebb learning rule is to pass once through the training set and adjust the weights accordingly.

■ **Example 8.5.1** To obtain a separation line for the logic function AND one should find w_1, w_2 and b such that the truth table (Table 8.5.1).

■ **Table 8.5.1** Truth table for AND using bipolar activations.

x_1	x_2	$\rightarrow t$
1	1	1
1	−1	0
−1	1	0
−1	−1	0

will be obtained by a neural net. Consider the initial values $w_1 = w_2 = b = 0$ and denote

$$\Delta w_i = w_i(new) - w_i(old) = x_i t\,,\ 1 \le i \le 2$$
$$\Delta b = b(new) - b(old) = t \tag{8.5.2}$$

The four 'extended patterns', namely

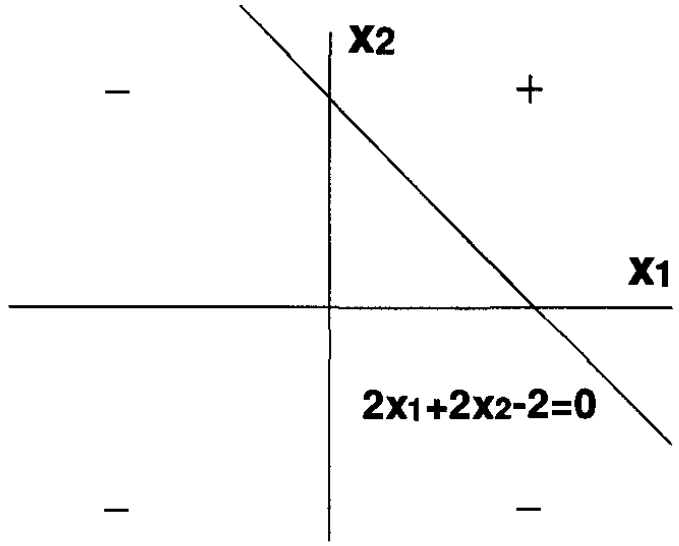
$$\begin{aligned} \boldsymbol{x}_1 &= (1,1,1) \\ \boldsymbol{x}_2 &= (1,-1,1) \\ \boldsymbol{x}_3 &= (-1,1,1) \\ \boldsymbol{x}_4 &= (-1,-1,1) \end{aligned}$$

enter the neural net and w_i, $1 \le i \le 2$ and b are adjusted using Eq. (8.5.2). The process is described in Table 8.5.2.

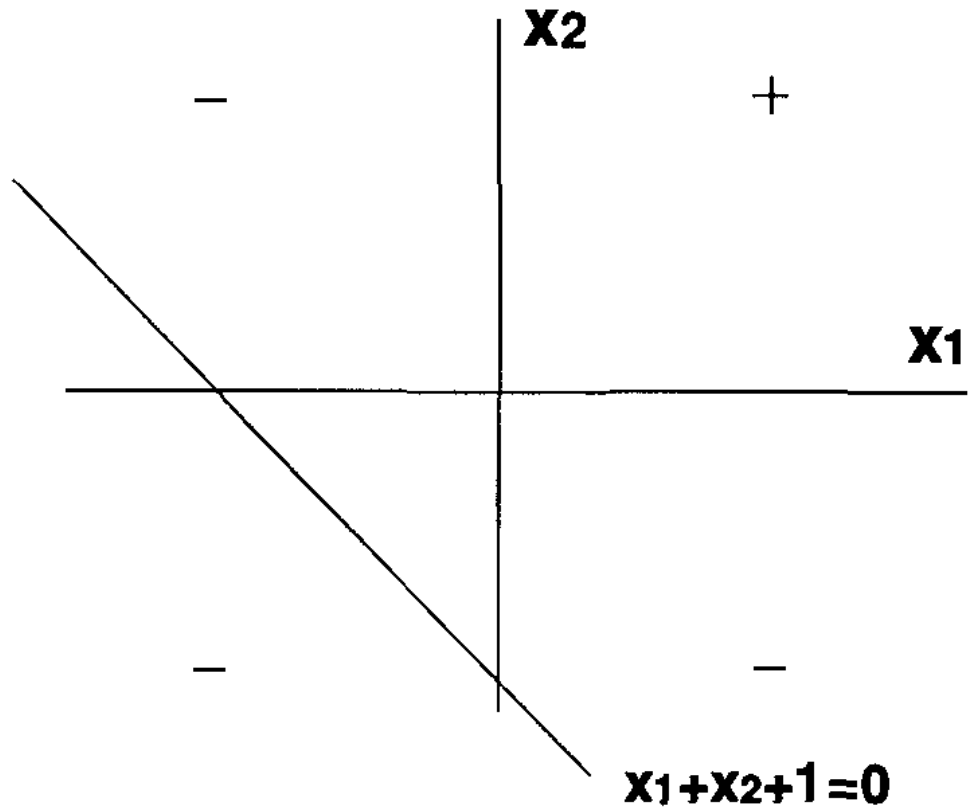
- **Table 8.5.2** Hebb rule applied to AND, using bipolar activations.

Input			Target	Weight Changes			Weights		
x_1	x_2	1	t	Δw_1	Δw_2	Δb	w_1	w_2	b
1	1	1	1	1	1	1	1	1	1
1	−1	1	−1	−1	1	−1	0	2	0
−1	1	1	−1	1	−1	−1	1	1	−1
−1	−1	1	−1	1	1	−1	2	2	−2

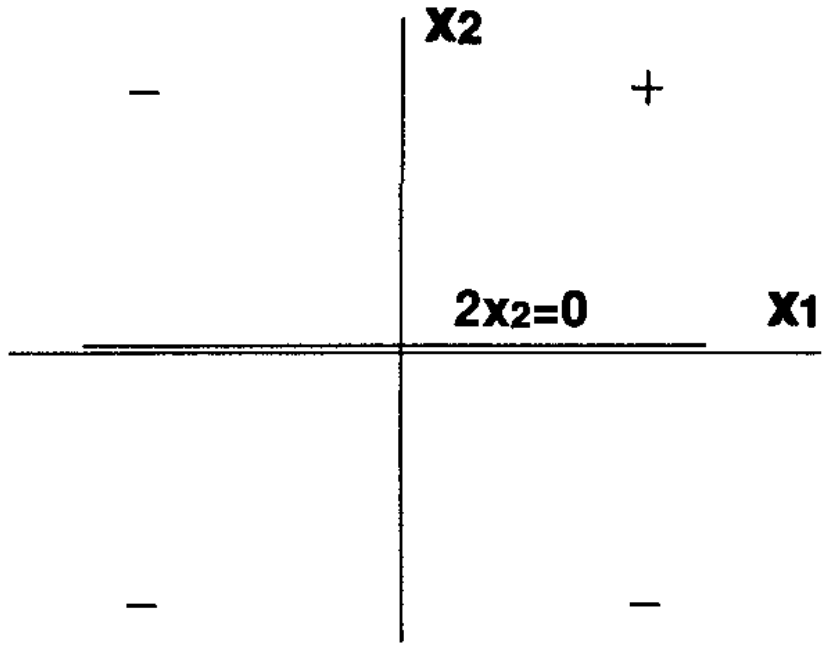
The final separation line is $2x_1 + 2x_2 - 2 = 0$ (Fig. 8.5.1). The decision boundaries suggested by the system at the interim stages are illustrated in Figs. (8.5.2) through (8.5.4). In this case the fourth training pattern is not needed and the decision boundary after the first three training patterns is already final.



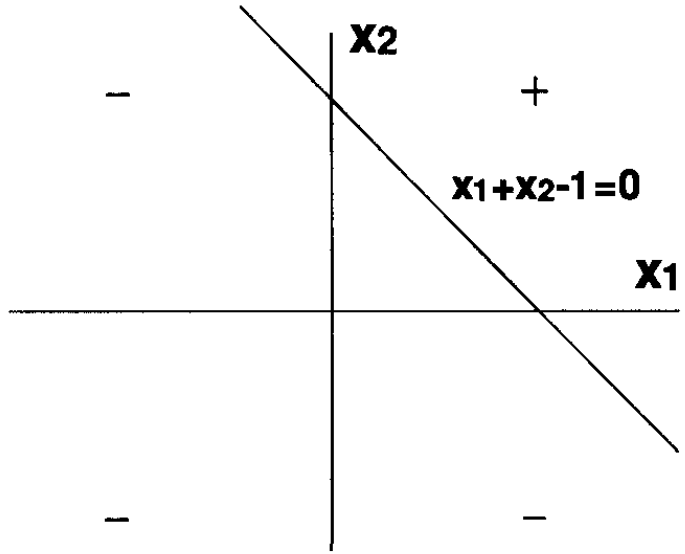
■ **Figure 8.5.1** Getting a separation line for AND.



Decision boundary after first training pattern.



Decision boundary after second training pattern.



Decision boundary after third training pattern.

The Hebb learning rule is limited and does not always provide a linear separator even if there is one. In the next example, the use of binary activations prevents the neural net from learning some of the patterns.

■ **Example 8.5.2** Consider a neural net which is assembled to model AND, using binary activations. The associated truth table is given in Table 8.3.1 and by applying Hebb rule and the initial conditions $w_1 = w_2 = b = 0$ we obtain

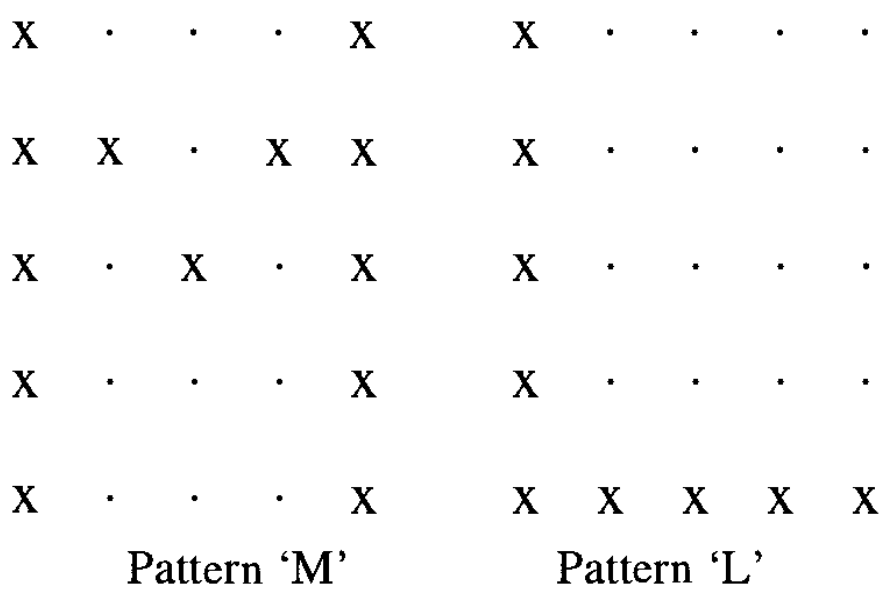
■ **Table 8.5.2** Hebb rule applied unsuccessfully to AND using binary activations.

Input			Target	Weight Changes			Weights		
x_1	x_2	1	t	Δw_1	Δw_2	Δb	w_1	w_2	b
1	1	1	1	1	1	1	1	1	1
1	0	1	0	0	0	0	1	1	1
0	1	1	0	0	0	0	1	1	1
0	0	1	0	0	0	0	1	1	1

Clearly, if the target t is 0, no learning occurs.

In the next example the Hebb net is applied to classify letters - represented by pixel matrices.

■ **Example 8.5.3** Consider a pattern classification problem where each pattern is either the letter 'M' or the letter 'L' represented by 5x5 pixel matrices (Fig. 8.5.5).



■ **Figure 8.5.5** Representation of 'M' and 'L' by pixels.

An arbitrary pattern is represented by a 25-component vector. The pixels 'x' and '.' are represented by the values 1 and –1 respectively and each vector is obtained by concatenating the rows of the corresponding pixel matrix starting at the top. The Hebb net consists of 25 input units and an output unit. There is no need to include here a bias unit.

The patterns 'M' and 'L' are therefore

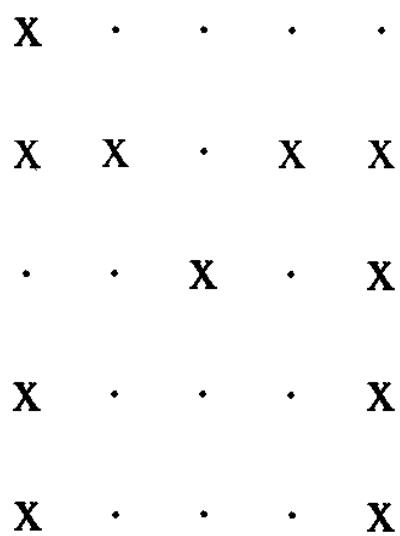
$$\text{'M'} = (1-1-1-11,11-111,1-11-11,1-1-1-11,1-1-1-11)^T$$

$$\text{'L'} = (1\ -1\ -1\ -1\ -1,\ 1\ -1\ -1\ -1\ -1,\ 1\ -1\ -1\ -1\ -1,\ 1\ -1\ -1\ -1\ -1,\ 11111)^T$$

and the desired outputs are 1 for 'M' and –1 for 'L'. If we start with $w_i = 0, 1 \le i \le 25$ and feed the system with the pattern 'M' and $t = 1$, we obtain that the weight change vector $\Delta w_1 = (\Delta w_{11}, \Delta w_{12}, \ldots, \Delta w_{1n})$ equals to 'M'. As we start with homogeneous initial conditions, the new weight vector is also 'M'. Since $t = -1$ (no response) for the training pattern 'L', the second weight change vector is '–L' and the final weight vector is

$$W_f = \text{'M'} - \text{'L'} = (00002, 02022, 00202, 00002, 0 - 2 - 2 - 20)^T$$

The net input into the output unit when 'M' is fed into the system is W_f^T 'M' $= 20 > 0$ and the response is 1. If 'L' enters the system, the net input is W_f^T 'L' $= -20 < 0$ and the response is -1 as desired. If an incoming pattern, given as a 25-component vector, includes noise or some wrong measurements it may still be classified as 'M' or 'L' provided that the noise and the errors in measurements are significantly small. For example, the pattern in Fig. 8.5.6 is similar to 'M'. Its representation as a 25-component vector is



■ **Figure 8.5.6** A pattern which resembles M.

$$\text{'M'} = (1\ -1\ -1\ -1\ -1, 11\ -111,\ -1\ -11\ -11, 1\ -1\ -1\ -11, 1\ -1\ -1\ -11)^T$$

and W_f^T 'M'=16, i.e. the pattern definitely produces a positive response and therefore classified in the M-class.

We now return and discuss the limitations of the Hebb net. In Example 8.5.2, using the Hebb rule with binary activations, prevented the *learning* of three out of four training patterns. Consequently, the Hebb neural net could not provide a linear separator for the AND logic function, although such a separator exists (Example 8.5.1). However, even if the Hebb net learns all the patterns and even if a linear separator exists, there is no guarantee that the final weights will indeed provide an appropriate separator.

8.6 THE PERCEPTRON

The limitations of the Hebb rule are not shared by another learning procedure which was also implemented in the earliest neural nets - the perceptron. Furthermore, sufficient conditions for the convergence of its iterative process exist.

The basic perceptron consists of three layers of neurons: a layer of *sensory units* S_i, $1 \le i \le m$; a layer of *associative units* X_i, $1 \le i \le n$ and a layer of *response units* Y_j, $1 \le j \le k$. The case of a single response unit is illustrated in Fig. 8.6.1. Each of the associative units is randomly connected to the sensory units with connection links over which the weights are prefixed. The sensory units transfer the stimuli from the measurement devices to the associative units and since no learning occurs at this stage, we may observe the associative units as input units and just consider the second and third layers (Fig. 8.6.2). We also assume a bias associative unit B. The activations x_i, $1 \le i \le n$ of the associative units are binary or bipolar, while that of Y is $1, 0$ or -1. The net input of the response unit Y is

$$y\_in = \sum_{i=1}^{n} w_i x_i + b \tag{8.6.1}$$

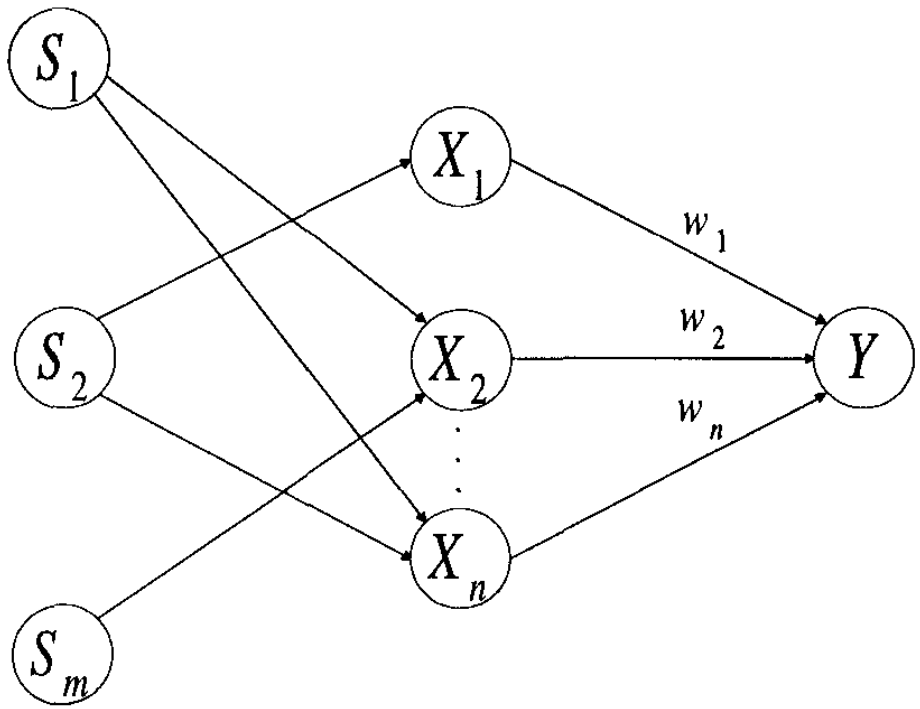


Figure 8.6.1 A perceptron model.

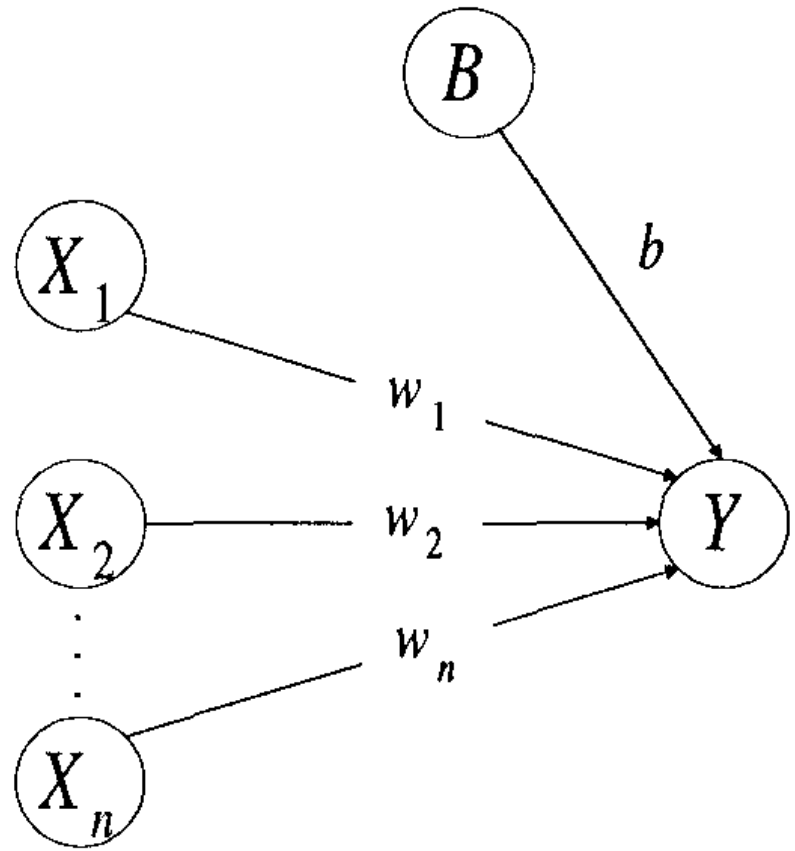


Figure 8.6.2 Associative and response units.

and its activation is defined by

$$f(y\_in) = \begin{cases} 1 & , \quad y\_in > \theta \\ 0 & , \quad -\theta \le y\_in \le \theta \\ -1 & , \quad y\_in < -\theta \end{cases} \tag{8.6.2}$$

Thus, the perceptron's output is 1 if the response unit's activation exceeds a given threshold θ. If the net input falls within a band of width θ around zero, the activation is set to zero. Otherwise it equals -1. The *target* is always $+1$ or -1. If it is 1, an error occurs whenever $f(y\_in)$ is either -1 or 0 and the weights associated with the connection links between the associative units and the response unit, will be adjusted by the perceptron learning rule.

The purpose of introducing the threshold θ is to be able to decisively distinguish between a positive and a negative response. This threshold determines a *neutral zone* between the two choices, and cannot be included in the bias b. Indeed, a positive response occurs if

$$\sum_{i=1}^{n} w_i x_i + b - \theta > 0 \tag{8.6.3}$$

while a negative one yields

$$\sum_{i=1}^{n} w_i x_i + b + \theta < 0 \tag{8.6.4}$$

Clearly, we cannot replace b and θ by a single parameter $(b-\theta)$ in Eq. (8.6.3) since in Eq. (8.6.4) that single parameter should be $b+\theta \neq b-\theta$.

For each pattern we calculate the response unit's activation $f(y\_in)$. If it is not equal to the target t the weights are adjusted by

$$w_i(new) = w_i(old) + \alpha t x_i \tag{8.6.5}$$

where α is a correction coefficient between 0 and 1 which can be observed as the *learning rate* of the perceptor. If, however, $f(y\_in) = t$ the weights are unchanged and the next pattern is tested. Each *iteration* consists of a complete sweep over the set of training patterns. The process stops if throughout an iteration no adjusting of w_i, $1 \le i \le n$ occurs, i.e. if all the training patterns provide the desired targets.

Algorithm 8.6.1.

(An algorithm for a basic perceptron: PERC)

Input:

m – the number of training patterns.
n – the number of associative units.
θ – the perceptron threshold.
α – the perceptron learning rate.

$\{x_{ij}\}_{j=1}^{n}$ – The activations of the i - th pattern, $1 \le i \le m$.

$t_i, 1 \le i \le m$ – The correct targets of the training patterns.

$w_{j0}, 1 \le j \le n$ – The initial weights.

b_0 – The initial bias.

Output:

$w_j, 1 \le j \le n$ – the final weights.

b – the final bias.

Step 1. Set $it = 0$ and $w_{j0}^{*} = w_{j0}$, $1 \le j \le n$, $b_0^{*} = b_0$

Step 2. Set $ichange = 0$ and for $1 \le i \le m$ do Steps 3-5.

Step 3. Calculate

$$y\_in = \sum_{j=1}^{n} w_{j0} x_{ij} + b_0$$

Step 4. Set

$$y = \begin{cases} 1 & , \quad y\_in > \theta \\ 0 & , \quad -\theta \le y\_in \le \theta \\ -1 & , \quad y\_in < -\theta \end{cases}$$

Step 5. Updating the weights and bias:

If $y \neq t_i$ set

$$ichange = 1$$
$$w_j = w_{j0} + \alpha\, t_i x_{ij}, 1 \leq j \leq n$$
$$b = b_0 + \alpha\, t$$

and then $w_{j0} \leftarrow w_j, 1 \leq j \leq n$ and $b_0 \leftarrow b$.
Otherwise continue.

Step 6. If $w_j = w_{j0}^*$, $1 \leq j \leq n$; $b = b_0^*$ and $ichange = 0$, output 'learning is successfully completed', it (no. of iterations) and stop. Otherwise, if $w_j = w_{j0}^*$, $1 \leq j \leq n$; $b = b_0^*$ and $ichange = 1$, output 'learning cannot be completed for all training patterns' and stop; otherwise set $it \leftarrow it + 1$;
b_0^*, $b_0 \leftarrow b$; w_{j0}^*, $w_{j0} \leftarrow w_j$, $1 \leq j \leq n$ and go to Step 2.

■ **Example 8.6.1** Consider the logic function OR where the input is binary and the targets are bipolar, i.e.

x_1	x_2	$\rightarrow$ t
1	1	1
1	0	1
0	1	1
0	0	−1

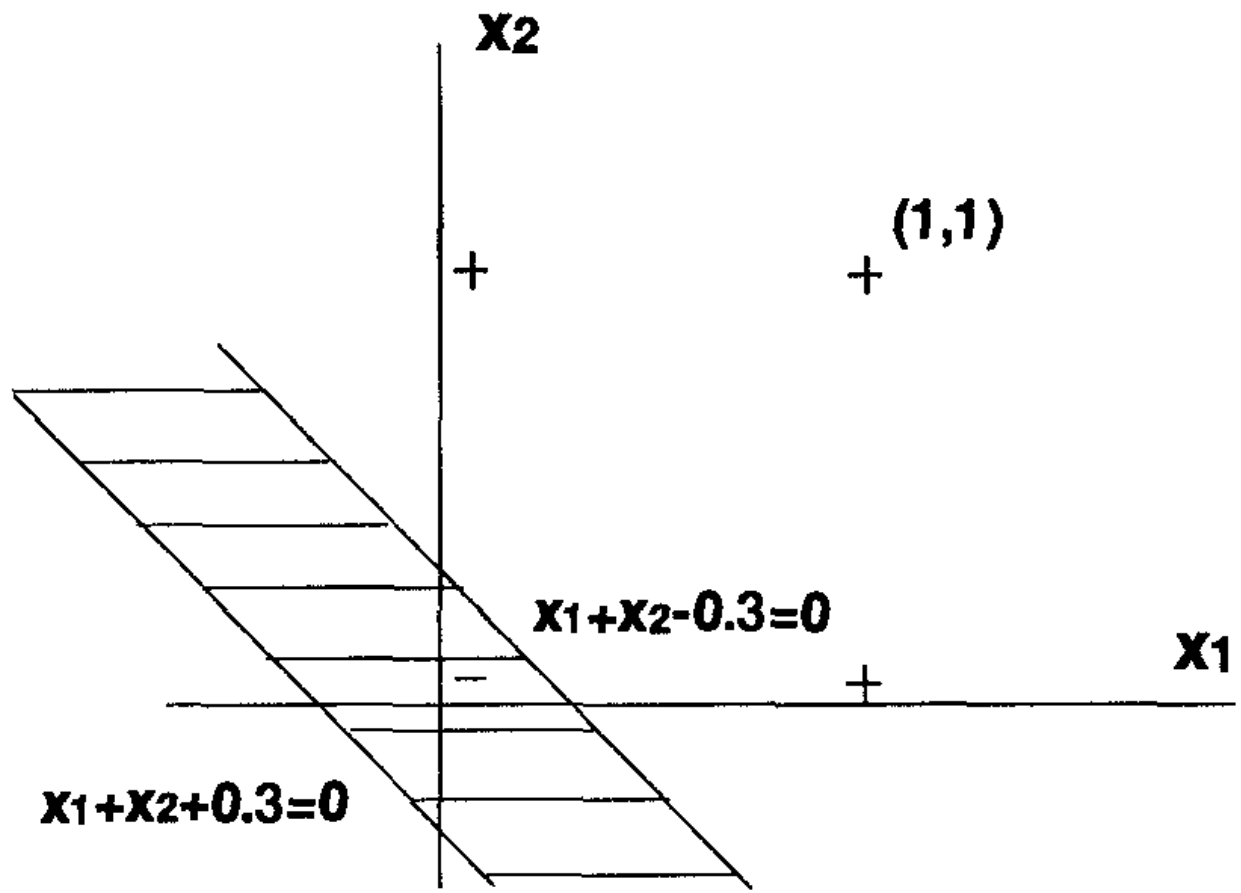
Choose $\theta = 0.3$, $\alpha = 1$ and $w_{10} = w_{20} = b_0 = 0$. The first iteration provides Table 8.6.1 which has a 'Net' column for y_in and an 'Output' column for $f(y\_in)$.

- **Table 8.6.1** The first perceptron iteration for OR.

Input			Net	Output	Target	Weight Changes			Weights		
x_1	x_2	1				Δw_1	Δw_2	Δb	w_1	w_2	b
1	1	1	0	0	1	1	1	1	1	1	1
1	0	1	2	1	1	0	0	0	1	1	1
0	1	1	2	1	1	0	0	0	1	1	1
0	0	1	1	1	−1	0	0	−1	1	1	0

At the end of this iteration or cycle we get the weights $w_1 = w_2 = 1$ and the bias $b = 0$ which yield a temporary *decision band* between the straight lines $x_1 + x_2 - 0.3 = 0$ and $x_1 + x_2 + 0.3 = 0$ (Fig. 8.6.3). In order for the decision band to provide correct results for all of the training patterns, the patterns with positive targets must fall on the positive side of the band, i.e. must satisfy $x_1 + x_2 - 0.3 > 0$, while the patterns with negative targets should fall on the negative side, i.e. satisfy $x_1 + x_2 + 0.3 < 0$. The threshold is therefore a parameter which indicates the *desired extent* of the decision band as a *separating zone* between the two classes of patterns. In this example the training input pattern (0,0) falls within the decision band and the process is not terminated. This can be also concluded by comparing the initial and final weights and bias:

$$(w_1, w_2, b) = (1,1,0) \neq (0,0,0) = (w_{10}^*, w_{20}^*, b_0^*)$$



The results of the next cycles are given below.

Cycle 2:

x_1	x_2	1	Net	Output	Target	Δw_1	Δw_2	Δb	w_1	w_2	b
1	1	1	2	1	1	0	0	0	1	1	0
1	0	1	2	1	1	0	0	0	1	1	0
0	1	1	2	1	1	0	0	0	1	1	0
0	0	1	0	0	−1	0	0	−1	1	1	−1

Cycle 3:

x_1	x_2	1	Net	Output	Target	Δw_1	Δw_2	Δb	w_1	w_2	b
1	1	1	1	1	1	0	0	0	1	1	−1
1	0	1	0	0	1	1	0	1	2	1	0
0	1	1	1	1	1	0	0	0	2	1	0
0	0	1	0	0	−1	0	0	−1	2	1	−1

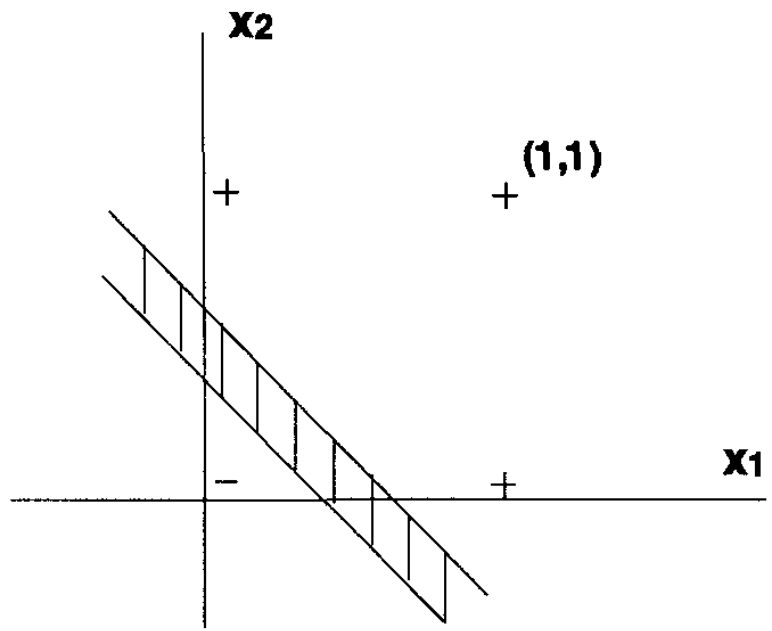
Cycle 4:

x_1	x_2	1	Net	Output	Target	Δw_1	Δw_2	Δb	w_1	w_2	b
1	1	1	2	1	1	0	0	0	2	1	−1
1	0	1	1	1	1	0	0	0	2	1	−1
0	1	1	0	0	1	0	1	1	2	2	0
0	0	1	0	0	−1	0	0	−1	2	2	−1

Cycle 5:

x_1	x_2	1	Net	Output	Target	Δw_1	Δw_2	Δb	w_1	w_2	b
1	1	1	3	1	1	0	0	0	2	2	−1
1	0	1	1	1	1	0	0	0	2	2	−1
0	1	1	1	1	1	0	0	0	2	2	−1
0	0	1	−1	−1	−1	0	0	0	2	2	−1

Since all the training patterns produce outputs which are identical to their corresponding targets the process ends successfully after five cycles. The final weights are $w_1 = w_2 = 2$ and the final bias is $b = -1$. The training patterns and the final separation band are illustrated in Fig. 8.6.4.



■ **Figure 8.6.4** A final separation band for OR.

We will next prove a convergence theorem for the perceptron learning rule.

Consider a set X of input training vectors $\boldsymbol{x}_i$, $1 \le i \le m$ with associated target values t_i, $1 \le i \le m$ respectively, such that t_i is either 1 or -1, and with an activation function $y = f(y\_in)$ such that

$$y = \begin{cases} 1 & , \quad y\_in > \theta \\ 0 & , \quad -\theta \le y\_in \le \theta \\ -1 & , \quad y\_in < -\theta \end{cases}$$

Let the new weights be updated (if $y \neq t$) by

$$\boldsymbol{w}(new) = \boldsymbol{w}(old) + \alpha t \boldsymbol{x}$$

If $y = t$ the weights remain the same.