5.9 Hand-worked Example and MATLAB Simulation

In this section we cement our understanding developed hitherto, with the help of a worked example in conjunction with a small MATLAB simulation.

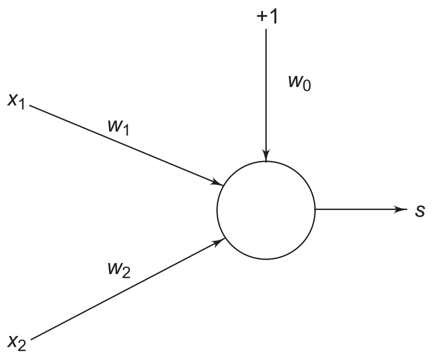


Fig. 5.11 Binary threshold neuron

Consider the familiar binary threshold neuron in two dimensions shown in Fig. 5.11. Assume that we wish to train the neuron of Fig. 5.11 to learn the AND pattern classification problem using Perceptron learning. For this we will use the four augmented patterns shown in Table 5.1. Here, $x_0 = 1$ in the first column of the table models the constant input to the threshold weight w_0.

Table 5.1 Augmented pattern set for logical AND problem in two dimensions

Pattern	x_0	x_1	x_2	d
1	1	0	0	0
2	1	0	1	0
3	1	1	0	0
4	1	1	1	1

Assume the following simulation parameters: $W_1 = (0\ 0\ 0)^T$, $\eta = 1$. Table 5.2 shows the iterations for this training set. A $\star$ denotes an ambiguous output. This occurs whenever the activation $y = X^T W = 0$, that is $\mathcal{S}(0) = \star$. Table 5.2 shows that it takes 32 iterations for the system to settle down. Iterations 33–36 simply check that no misclassification occurs for the four patterns in question. It is a useful exercise to work through some iterations (say the first eight) out by hand.

Recall that each update of weights corresponds to a shift in the separating hyperplane represented by:

$$w_1x_1 + w_2x_2 + w_0 = 0 \tag{5.4}$$

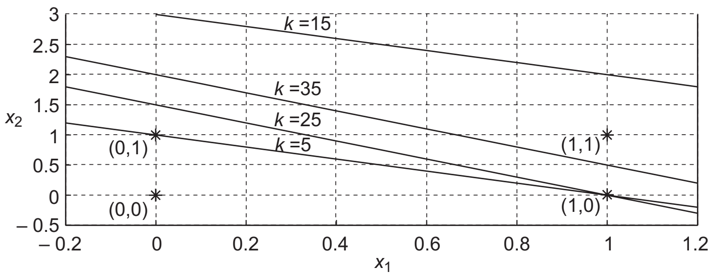
It is important to understand the correspondence between a weight update and the movement of the separating hyperplane which in this case is a straight line. Each instance of weights W_k defines a separating line in x_1–x_2 space; each weight perturbation causes a corresponding perturbation in the orientation and position of the line. This is made clear in Fig. 5.12(a) which plots hyperplane positions after 5, 15, 25 and 35 iterations. The correspondence between learning and hyperplane movement should now be clear from Fig. 5.14.

It is also a good idea to keep both pattern space and weight space views in one's mind while understanding the working of the Perceptron algorithm. For example, see Fig. 5.12(b) which plots weight perturbations corresponding to Table 5.2 in weight space from an initial weight (0, 0, 0) to a final weight of $(-4, 3, 2)$. In weight space, each perturbation causes a movement in a direction perpendicular to the pattern hyperplane of the corresponding pattern that causes the perturbation. That is why if you observe carefully, the weight space perturbations in Fig. 5.12(b) take place only in four directions—perpendicular to each of the four corresponding pattern hyperplanes.

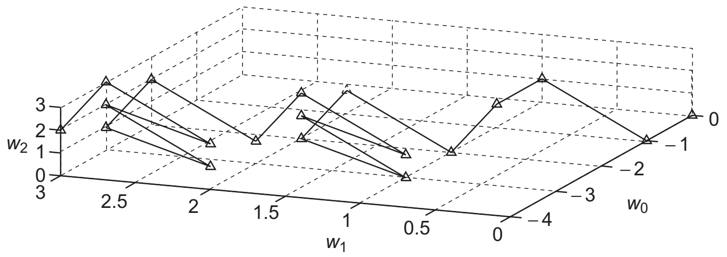
Figure 5.12 was generated with the help of the MATLAB code of Table **??**. Study the code carefully. Notice that this code implementation works with the non-adjusted training set, that is $X_k \in \mathcal{X}$. It therefore uses Perceptron learning in the form specified in Eq. **??**. The code is straightforward to understand from the comments. An entry in the binary `update` flag vector is set whenever the weights are updated on the pattern corresponding to that entry. In any epoch, the number of updates is the inner product of the `update` flag with itself. The `update` flag is reset on every epoch, and the algorithm terminates when no weight updates are made in a complete epoch.

Table 5.2 Weight updates through 36 iterations for the binary threshold neuron of Fig. 5.11 using Perceptron learning Eq. **??**

Iteration No.	X_k			W_k			y	s	W_{k+1}		
	x_0	x_1	x_2	w_0	w_1	w_2					
1	1	0	0	0	0	0	0	⋆	–1	0	0
2	1	0	1	–1	0	0	–1	–1	–1	0	0
3	1	1	0	–1	0	0	–1	–1	–1	0	0
4	1	1	1	–1	0	0	–1	–1	0	1	1
5	1	0	0	0	1	1	0	⋆	–1	1	1
6	1	0	1	–1	1	1	0	⋆	–2	1	0
7	1	1	0	–2	1	0	–1	–1	–2	1	0
8	1	1	1	–2	1	0	–1	–1	–1	2	1
9	1	0	0	–1	2	1	–1	–1	–1	2	1
10	1	0	1	–1	2	1	0	⋆	–2	2	0
11	1	1	0	–2	2	0	0	⋆	–3	1	0
12	1	1	1	–3	1	0	–2	–1	–2	2	1
13	1	0	0	–2	2	1	–2	–1	–2	2	1
14	1	0	1	–2	2	1	–1	–1	–2	2	1
15	1	1	0	–2	2	1	0	⋆	–3	1	1
16	1	1	1	–3	1	1	–1	–1	–2	2	2
17	1	0	0	–2	2	2	–2	–1	–2	2	2
18	1	0	1	–2	2	2	0	⋆	–3	2	1
19	1	1	0	–3	2	1	–1	–1	–3	2	1
20	1	1	1	–3	2	1	0	⋆	–2	3	2
21	1	0	0	–2	3	2	–2	–1	–2	3	2
22	1	0	1	–2	3	2	0	⋆	–3	3	1
23	1	1	0	–3	3	1	0	⋆	–4	2	1
24	1	1	1	–4	2	1	–1	–1	–3	3	2
25	1	0	0	–3	3	2	–3	–1	–3	3	2
26	1	0	1	–3	3	2	–1	–1	–3	3	2
27	1	1	0	–3	3	2	0	⋆	–4	2	2
28	1	1	1	–4	2	2	0	⋆	–3	3	3
29	1	0	0	–3	3	3	–3	–1	–3	3	3
30	1	0	1	–3	3	3	0	⋆	–4	3	2
31	1	1	0	–4	3	2	–1	–1	–4	3	2
32	1	1	1	–4	3	2	1	1	–4	3	2
33	1	0	0	–4	3	2	–4	–1	–4	3	2
34	1	0	1	–4	3	2	–2	–1	–4	3	2
35	1	1	0	–4	3	2	–1	–1	–4	3	2
36	1	1	1	–4	3	2	1	1	–4	3	2



(a) Hyperplane movement depicted during Perceptron learning



(b) Weight space trajectories: w_0 = (0,0,0,) w_s = (– 4 3 2)

Fig. 5.12 Computer simulations of (a) hyperplane movement in pattern space; and (b) weight vector movement in weight space