Algorithm 5.4 Perceptron learning algorithm.

```
1: Let $D = \{(\mathbf{x}_i, y_i) \mid i = 1, 2, \ldots, N\}$ be the set of training examples.
2: Initialize the weight vector with random values, $\mathbf{w}^{(0)}$
3: repeat
4:   for each training example $(\mathbf{x}_i, y_i) \in D$ do
5:     Compute the predicted output $\hat{y}_i^{(k)}$
6:     for each weight $w_j$ do
7:       Update the weight, $w_j^{(k+1)} = w_j^{(k)} + \lambda(y_i - \hat{y}_i^{(k)})x_{ij}$.
8:     end for
9:   end for
10: until stopping condition is met
```

to the prediction error, $(y - \hat{y})$. If the prediction is correct, then the weight remains unchanged. Otherwise, it is modified in the following ways:

- If $y = +1$ and $\hat{y} = -1$, then the prediction error is $(y - \hat{y}) = 2$. To compensate for the error, we need to increase the value of the predicted output by increasing the weights of all links with positive inputs and decreasing the weights of all links with negative inputs.

- If $y_i = -1$ and $\hat{y} = +1$, then $(y - \hat{y}) = -2$. To compensate for the error, we need to decrease the value of the predicted output by decreasing the weights of all links with positive inputs and increasing the weights of all links with negative inputs.

In the weight update formula, links that contribute the most to the error term are the ones that require the largest adjustment. However, the weights should not be changed too drastically because the error term is computed only for the current training example. Otherwise, the adjustments made in earlier iterations will be undone. The learning rate λ, a parameter whose value is between 0 and 1, can be used to control the amount of adjustments made in each iteration. If λ is close to 0, then the new weight is mostly influenced by the value of the old weight. On the other hand, if λ is close to 1, then the new weight is sensitive to the amount of adjustment performed in the current iteration. In some cases, an adaptive λ value can be used; initially, λ is moderately large during the first few iterations and then gradually decreases in subsequent iterations.

The perceptron model shown in Equation 5.23 is linear in its parameters $\mathbf{w}$ and attributes $\mathbf{x}$. Because of this, the decision boundary of a perceptron, which is obtained by setting $\hat{y} = 0$, is a linear hyperplane that separates the data into two classes, -1 and $+1$. Figure 5.15 shows the decision boundary

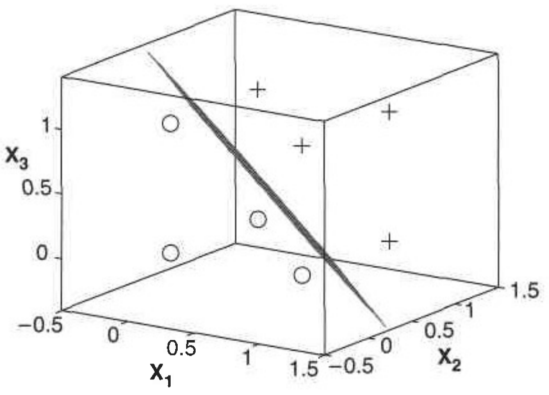

Figure 5.15. Perceptron decision boundary for the data given in Figure 5.14.

obtained by applying the perceptron learning algorithm to the data set given in Figure 5.14. The perceptron learning algorithm is guaranteed to converge to an optimal solution (as long as the learning rate is sufficiently small) for linearly separable classification problems. If the problem is not linearly separable, the algorithm fails to converge. Figure 5.16 shows an example of nonlinearly separable data given by the XOR function. Perceptron cannot find the right solution for this data because there is no linear hyperplane that can perfectly separate the training instances.

X_1	X_2	y
0	0	−1
1	0	1
0	1	1
1	1	−1

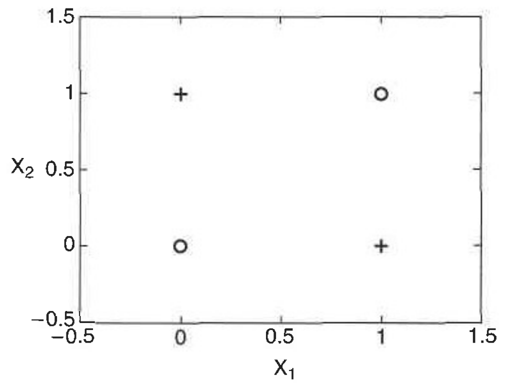

Figure 5.16. XOR classification problem. No linear hyperplane can separate the two classes.