2.4 The Perceptron

After the linear networks, the *perceptron* is the simplest type of neural network and it is typically used for classification. In the one-output case it consists of a neuron with a step function. Figure 2.4 is a graphical illustration of a perceptron with inputs $x_1, \ldots, x_n$ and output $\hat{y}$.

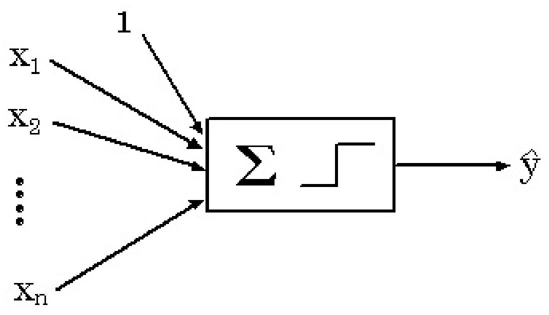

Figure 2.4. A perceptron classifier.

As indicated, the weighted sum of the inputs and the unity bias are first summed and then processed by a step function to yield the output

$$\hat{y}(x, w, b) = \text{UnitStep}[w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b] \qquad (4)$$

where $\{w_1, \ldots, w_n\}$ are the weights applied to the input vector and b is the *bias weight*. Each weight is indicated with an arrow in Figure 2.4. Also, the `UnitStep` function is 0 for arguments less than 0 and 1 elsewhere. So, the output $\hat{y}$ can take values of 0 or 1, depending on the value of the weighted sum. Consequently, the perceptron can indicate two classes corresponding to these two output values. In the training process, the weights (input and bias) are adjusted so that input data is mapped correctly to one of the two classes. An example can be found in Section 4.2.1, Two Classes in Two Dimensions.

The perceptron can be trained to solve any two-class classification problem where the classes are *linearly separable*. In two-dimensional problems (where x is a two-component row vector), the classes may be separated by a straight line, and in higher-dimensional problems it means that the classes are separable by a hyperplane.

If the classification problem is not linearly separable, then it is impossible to obtain a perceptron that correctly classifies all training data. If some misclassifications can be accepted, then a perceptron could still constitute a good classifier.

Because of its simplicity, the perceptron is often inadequate as a model for many problems. Nevertheless, many classification problems have simple solutions for which it may apply. Also, important insights may be gained from using the perceptron, which may shed some light in considering more complicated neural network models.

Perceptron classifiers are trained with a supervised training algorithm. This presupposes that the true classes of the training data are available and incorporated in the training process. More specifically, as individual inputs are presented to the perceptron, its weights are adjusted iteratively by the training algorithm so as to produce the correct class mapping at the output. This training process continues until the perceptron correctly classifies all the training data or when a maximum number of iterations has been reached. It is possible to choose a judicious initialization of the weight values, which in many cases makes the iterative learning unnecessary. This is described in Section 4.1.1, InitializePerceptron.

Classification problems involving a number of classes greater than two can be handled by a multi-output perceptron that is defined as a number of perceptrons in parallel. It contains one perceptron, as shown in Figure

2.4, for each output, and each output corresponds to a class.

The training process of such a multi-output perceptron structure attempts to map each input of the training data to the correct class by iteratively adjusting the weights to produce 1 at the output of the corresponding perceptron and 0 at the outputs of all the remaining outputs. However, it is quite possible that a number of input vectors may map to multiple classes, indicating that these vectors could belong to several classes. Such cases may require special handling. It may also be that the perceptron classifier cannot make a decision for a subset of input vectors because of the nature of the data or insufficient complexity of the network structure itself. An example with several classes can be found in 4.2.2 Several Classes in Two Dimensions.

Training Algorithm

The training of a one-output perceptron will be described in the following section. In the case of a multi-output perceptron, each of the outputs may be described similarly.

A perceptron is defined parametrically by its weights $\{w,b\}$, where w is a column vector of length equal to the dimension of the input vector x and b is a scalar. Given the input, a row vector, $x=\{x_1,\ldots,x_n\}$, the output of a perceptron is described in compact form by

$$\hat{y}(x, w, b) = \text{UnitStep}[x\, w + b] \tag{5}$$

This description can be used also when a set of input vectors is considered. Let x be a matrix with one input vector in each row. Then $\hat{y}$ in Eq. (2.5) becomes a column vector with the corresponding output in its rows.

The weights $\{w,b\}$ are obtained by iteratively training the perceptron with a known data set containing input-output pairs, one input vector in each row of a matrix x, and one output in each row of a matrix y, as described in Section 3.2.1, Data Format. Given N such pairs in the data set, the training algorithm is defined by

$$\begin{aligned} w_{i+1} &= w_i + \eta\, x^T \epsilon_i \\ b_{i+1} &= b_i + \eta \sum_{j=1}^{N} \epsilon_i[[j]] \end{aligned} \tag{6}$$

where i is the iteration number, η is a scalar step size, and $\epsilon_i = y - \hat{y}(x, w_i, b_i)$ is a column vector with N-components of classification errors corresponding to the N data samples of the training set. The components of the error vector can only take three values, namely, 0, 1, and -1. At any iteration i, values of 0 indicate that the corresponding data samples have been classified correctly, while all the others have been classified incorrectly.

The training algorithm Eq. (2.5) begins with initial values for the weights $\{w,b\}$ and $i=0$, and iteratively updates these weights until all data samples have been classified correctly or the iteration number has reached a maximum value, i_{max}.

The step size η, or *learning rate* as it is often called, has the following default value

$$\eta = \frac{(\text{Max}[x] - \text{Min}[x])}{N} \tag{7}$$

By compensating for the range of the input data, x, and for the number of data samples, N, this default value of η should be good for many classification problems independent of the number of data samples and their numerical range. It is also possible to use a step size of choice rather than using the default value. However, although larger values of η might accelerate the training process, they may induce oscillations that may slow down the convergence.