where $y_{nk} = y_k(\mathbf{x}_n, \mathbf{w})$, and t_{nk} is the associated target value. The gradient of this error function with respect to a weight w_{ji} is given by

$$\frac{\partial E_n}{\partial w_{ji}} = (y_{nj} - t_{nj})x_{ni}. \tag{8.4}$$

This can be interpreted as a 'local' computation involving the product of an 'error signal' $y_{nj} - t_{nj}$ associated with the output end of the link w_{ji} and the variable x_{ni} associated with the input end of the link. In Section 5.4.3, we saw how a similar formula arises with the logistic-sigmoid activation function together with the cross-entropy error function and similarly for the softmax activation function together with its matching multivariate cross-entropy error function. We will now see how this simple result extends to the more complex setting of multilayer feed-forward networks.

8.1.2 General feed-forward networks

In general, a feed-forward network consists of a set of units each of which computes a weighted sum of its inputs:

$$a_j = \sum_i w_{ji} z_i \tag{8.5}$$

where z_i is either the activation of another unit or an input unit that sends a connection to unit j, and w_{ji} is the weight associated with that connection. Biases can be included in this sum by introducing an extra unit, or input, with activation fixed at $+1$, and so we do not need to deal with biases explicitly. The sum in (8.5), known as a pre-activation, is transformed by a nonlinear activation function $h(\cdot)$ to give the activation z_j of unit j in the form

Section 6.2

$$z_j = h(a_j). \tag{8.6}$$

Note that one or more of the variables z_i in the sum in (8.5) could be an input, and similarly, the unit j in (8.6) could be an output.

For each data point in the training set, we will suppose that we have supplied the corresponding input vector to the network and calculated the activations of all the hidden and output units in the network by successive application of (8.5) and (8.6). This process is called *forward propagation* because it can be regarded as a forward flow of information through the network.

Now consider the evaluation of the derivative of E_n with respect to a weight w_{ji}. The outputs of the various units will depend on the particular input data point n. However, to keep the notation uncluttered, we will omit the subscript n from the network variables. First note that E_n depends on the weight w_{ji} only via the summed input a_j to unit j. We can therefore apply the chain rule for partial derivatives to give

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j}\frac{\partial a_j}{\partial w_{ji}}. \tag{8.7}$$

We now introduce a useful notation:

$$\delta_j \equiv \frac{\partial E_n}{\partial a_j} \tag{8.8}$$