Algorithm 8.1: Backpropagation

Input: Input vector $\mathbf{x}_n$
Network parameters $\mathbf{w}$
Error function $E_n(\mathbf{w})$ for input x_n
Activation function $h(a)$

Output: Error function derivatives $\{\partial E_n/\partial w_{ji}\}$

`// Forward propagation`

for $j \in$ all hidden and output units **do**

$a_j \leftarrow \sum_i w_{ji} z_i$ `// {`z_i`} includes inputs {`x_i`}`

$z_j \leftarrow h(a_j)$ `// activation function`

end for

`// Error evaluation`

for $k \in$ all output units **do**

$\delta_k \leftarrow \dfrac{\partial E_n}{\partial a_k}$ `// compute errors`

end for

`// Backward propagation, in reverse order`

for $j \in$ all hidden units **do**

$\delta_j \leftarrow h'(a_j) \sum_k w_{kj} \delta_k$ `// recursive backward evaluation`

$\dfrac{\partial E_n}{\partial w_{ji}} \leftarrow \delta_j z_i$ `// evaluate derivatives`

end for

return $\left\{ \dfrac{\partial E_n}{\partial w_{ji}} \right\}$

in Figure 8.1. Note that the summation in (8.13) is taken over the first index on w_{kj} (corresponding to backward propagation of information through the network), whereas in the forward propagation equation (8.5), it is taken over the second index. Because we already know the values of the δ's for the output units, it follows that by recursively applying (8.13), we can evaluate the δ's for all the hidden units in a feed-forward network, regardless of its topology. The backpropagation procedure is summarized in Algorithm 8.1.

Exercise 8.2

For batch methods, the derivative of the total error E can then be obtained by repeating the above steps for each data point in the training set and then summing over all data points in the batch or mini-batch:

$$\frac{\partial E}{\partial w_{ji}} = \sum_n \frac{\partial E_n}{\partial w_{ji}}. \tag{8.14}$$