which follows from (8.13) and (8.16). Finally, the derivatives with respect to the first-layer and second-layer weights are given by

$$\frac{\partial E_n}{\partial w_{ji}^{(1)}} = \delta_j x_i, \qquad \frac{\partial E_n}{\partial w_{kj}^{(2)}} = \delta_k z_j. \tag{8.23}$$

8.1.4 Numerical differentiation

One of the most important aspects of backpropagation is its computational efficiency. To understand this, let us examine how the number of compute operations required to evaluate the derivatives of the error function scales with the total number W of weights and biases in the network.

A single evaluation of the error function (for a given input data point) would require $\mathcal{O}(W)$ operations, for sufficiently large W. This follows because, except for a network with very sparse connections, the number of weights is typically much greater than the number of units, and so the bulk of the computational effort in forward propagation arises from evaluation of the sums in (8.5), with the evaluation of the activation functions representing a small overhead. Each term in the sum in (8.5) requires one multiplication and one addition, leading to an overall computational cost that is $\mathcal{O}(W)$.

An alternative approach to backpropagation for computing the derivatives of the error function is to use finite differences. This can be done by perturbing each weight in turn and approximating the derivatives by using the expression

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{E_n(w_{ji}+\epsilon) - E_n(w_{ji})}{\epsilon} + \mathcal{O}(\epsilon) \tag{8.24}$$

where $\epsilon \ll 1$. In a software simulation, the accuracy of the approximation to the derivatives can be improved by making ϵ smaller, until numerical round-off problems arise. The accuracy of the finite differences method can be improved significantly by using symmetrical *central differences* of the form

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{E_n(w_{ji}+\epsilon) - E_n(w_{ji}-\epsilon)}{2\epsilon} + \mathcal{O}(\epsilon^2). \tag{8.25}$$

Exercise 8.3

In this case, the $\mathcal{O}(\epsilon)$ corrections cancel, as can be verified by a Taylor expansion of the right-hand side of (8.25), and so the residual corrections are $\mathcal{O}(\epsilon^2)$. Note, however, that the number of computational steps is roughly doubled compared with (8.24). Figure 8.2 shows a plot of the error between a numerical evaluation of a gradient using both finite differences (8.24) and central differences (8.25) versus the analytical result, as a function of the value of the step size ϵ.

The main problem with numerical differentiation is that the highly desirable $\mathcal{O}(W)$ scaling has been lost. Each forward propagation requires $\mathcal{O}(W)$ steps, and there are W weights in the network each of which must be perturbed individually, so that the overall computational cost is $\mathcal{O}(W^2)$.

However, numerical differentiation can play a useful role in practice, because a comparison of the derivatives calculated from a direct implementation of backpropagation, or from automatic differentiation, with those obtained using central differences provides a powerful check on the correctness of the software.