be necessary to run a gradient-based algorithm multiple times, each time using a different randomly chosen starting point, and comparing the resulting performance on an independent validation set.

7.2.1 Use of gradient information

Chapter 8

The gradient of an error function for a deep neural network can be evaluated efficiently using the technique of error backpropagation, and applying this gradient information can lead to significant improvements in the speed of network training. We can see why this is so, as follows.

Exercise 7.7

In the quadratic approximation to the error function given by (7.3), the error surface is specified by the quantities $\mathbf{b}$ and $\mathbf{H}$, which contain a total of $W(W+3)/2$ independent elements (because the matrix $\mathbf{H}$ is symmetric), where W is the dimensionality of $\mathbf{w}$ (i.e., the total number of learnable parameters in the network). The location of the minimum of this quadratic approximation therefore depends on $\mathcal{O}(W^2)$ parameters, and we should not expect to be able to locate the minimum until we have gathered $\mathcal{O}(W^2)$ independent pieces of information. If we do not make use of gradient information, we would expect to have to perform $\mathcal{O}(W^2)$ function evaluations, each of which would require $\mathcal{O}(W)$ steps. Thus, the computational effort needed to find the minimum using such an approach would be $\mathcal{O}(W^3)$.

Chapter 8

Now compare this with an algorithm that makes use of the gradient information. Because ∇E is a vector of length W, each evaluation of ∇E brings W pieces of information, and so we might hope to find the minimum of the function in $\mathcal{O}(W)$ gradient evaluations. As we shall see, by using error backpropagation, each such evaluation takes only $\mathcal{O}(W)$ steps and so the minimum can now be found in $\mathcal{O}(W^2)$ steps. Although the quadratic approximation only holds in the neighbourhood of a minimum, the efficiency gains are generic. For this reason, the use of gradient information forms the basis of all practical algorithms for training neural networks.

7.2.2 Batch gradient descent

The simplest approach to using gradient information is to choose the weight update in (7.15) such that there is a small step in the direction of the negative gradient, so that

$$\mathbf{w}^{(\tau)} = \mathbf{w}^{(\tau-1)} - \eta \nabla E(\mathbf{w}^{(\tau-1)}) \tag{7.16}$$

where the parameter $\eta > 0$ is known as the *learning rate*. After each such update, the gradient is re-evaluated for the new weight vector $\mathbf{w}^{(\tau+1)}$ and the process repeated. At each step, the weight vector is moved in the direction of the greatest rate of decrease of the error function, and so this approach is known as *gradient descent* or *steepest descent*. Note that the error function is defined with respect to a training set, and so to evaluate ∇E, each step requires that the entire training set be processed. Techniques that use the whole data set at once are called *batch* methods.

7.2.3 Stochastic gradient descent

Deep learning methods benefit greatly from very large data sets. However, batch methods can become extremely inefficient if there are many data points in the training set because each error function or gradient evaluation requires the entire data set