7.4.2 Batch normalization

We have seen the importance of normalizing the input data, and we can apply similar reasoning to the variables in each hidden layer of a deep network. If there is wide variation in the range of activation values in a particular hidden layer, then normalizing those values to have zero mean and unit variance should make the learning problem easier for the next layer. However, unlike normalization of the input values, which can be done once prior to the start of training, normalization of the hidden-unit values will need to be repeated during training every time the weight values are updated. This is called *batch normalization* (Ioffe and Szegedy, 2015).

A further motivation for batch normalization arises from the phenomena of *vanishing gradients* and *exploding gradients*, which occur when we try to train very deep neural networks. From the chain rule of calculus, the gradient of an error function E with respect to a parameter in the first layer of the network is given by

Section 8.1.5

$$\frac{\partial E}{\partial w_i} = \sum_m \cdots \sum_l \sum_j \frac{\partial z_m^{(1)}}{\partial w_i} \cdots \frac{\partial z_j^{(K)}}{\partial z_l^{(K-1)}} \frac{\partial E}{\partial z_j^{(K)}} \tag{7.51}$$

where $z_j^{(k)}$ denotes the activation of node j in layer k, and each of the partial derivatives on the right-hand side of (7.51) represents the elements of the Jacobian matrix for that layer. The product of a large number of such terms will tend towards 0 if most of them have a magnitude < 1 and will tend towards ∞ if most of them have a magnitude > 1. Consequently, as the depth of a network increases, error function gradients can tend to become either very large or very small. Batch normalization largely resolves this issue.

Section 8.1.5

To see how batch normalization is defined, consider a specific layer within a multi-layer network. Each hidden unit in that layer computes a nonlinear function of its input pre-activation $z_i = h(a_i)$, and so we have a choice of whether to normalize the pre-activation values a_i or the activation values z_i. In practice, either approach may be used, and here we illustrate the procedure by normalizing the pre-activations. Because weight values are updated after each mini-batch of examples, we apply the normalization to each mini-batch. Specifically, for a mini-batch of size K, we define

$$\mu_i = \frac{1}{K}\sum_{n=1}^{K} a_{ni} \tag{7.52}$$

$$\sigma_i^2 = \frac{1}{K}\sum_{n=1}^{K} (a_{ni} - \mu_i)^2 \tag{7.53}$$

$$\widehat{a}_{ni} = \frac{a_{ni} - \mu_i}{\sqrt{\sigma_i^2 + \delta}} \tag{7.54}$$

where the summations over $n = 1, \ldots, K$ are taken over the elements of the mini-batch. Here δ is a small constant, introduced to avoid numerical issues in situations where σ_i^2 is small.

By normalizing the pre-activations in a given layer of the network, we reduce the number of degrees of freedom in the parameters of that layer and hence we