$$\overline{v}_7 = 1 \tag{8.70}$$
$$\overline{v}_6 = \overline{v}_7 \tag{8.71}$$
$$\overline{v}_5 = \overline{v}_7 \tag{8.72}$$
$$\overline{v}_4 = -\overline{v}_6 \tag{8.73}$$
$$\overline{v}_3 = \overline{v}_5 v_5 + \overline{v}_6 \tag{8.74}$$
$$\overline{v}_2 = \overline{v}_2 v_1 + \overline{v}_4 \cos(v_2) \tag{8.75}$$
$$\overline{v}_1 = \overline{v}_3 v_2. \tag{8.76}$$

Note that these start at the output and then flow backwards through the graph to the inputs. Even with multiple inputs, only a single backward pass is required to evaluate the derivatives. For a neural network error function, the derivatives of E with respect to the weight and biases are obtained as the corresponding adjoint variables. However, if we now have more than one output then we need to run a separate backward pass for each output variable.

Figure 8.5

Reverse mode is often more memory intensive than forward mode because all of the intermediate primal variables must be stored so that they will be available as needed when evaluating the adjoint variables during the backward pass. By contrast, with forward mode, the primal and tangent variables are computed together during the forward pass, and therefore variables can be discarded once they have been used. It is therefore also generally easier to implement forward mode compared to reverse mode.

For both forward-mode and reverse-mode automatic differentiation, a single pass through the network is guaranteed to take no more than 6 times the computational cost of a single function evaluation. In practice, the overhead is typically closer to a factor of 2 or 3 (Griewank and Walther, 2008). Hybrids of forward and reverse modes are also of interest. One situation in which this arises is in the evaluation of the product of a Hessian matrix with a vector, which can be calculated without explicit evaluation of the full Hessian (Pearlmutter, 1994). Here we can use reverse mode to calculate the gradient of code, which itself has been generated by the forward model. We start from a vector $\mathbf{b}$ and a point $\mathbf{x}$ at which the Hessian–vector product is to be evaluated. By setting $\dot{\mathbf{x}} = \mathbf{v}$ and using forward mode, we obtain the directional derivative $\mathbf{v}^{\mathrm{T}}\nabla f$. This is then differentiated using reverse mode to obtain $\nabla^2 f\mathbf{v} = \mathbf{H}\mathbf{v}$. If W is the number of parameters in the neural network then this evaluation has $\mathcal{O}(W)$ complexity even though the Hessian is of size $W \times W$. The Hessian itself can also be evaluated explicitly using automatic differentiation but this has $\mathcal{O}(W^2)$ complexity.

Exercises

8.1 ($\star$) By making use of (8.5), (8.6), (8.8), and (8.12), verify the backpropagation formula (8.13) for evaluating the derivatives of an error function.

8.2 ($\star\star$) Consider a network that consists of layers and rewrite the backpropagation formula (8.13) in matrix notation by starting with the forward propagation equation (6.19). Note that the result involves multiplication by the transposes of the matrices.