To compute column j of the Jacobian, we need to initialize the forward pass of the tangent equations by setting $\dot{x}_j = 1$ and $\dot{x}_i = 0$ for $i \neq j$. We can write this in vector form as $\dot{\mathbf{x}} = \mathbf{e}_i$ where $\mathbf{e}_i$ is the ith unit vector. To compute the full Jacobian matrix we need D forward-mode passes. However, if we wish to evaluate the product of the Jacobian with a vector $\mathbf{r} = (r_1, \ldots, r_D)^{\mathrm{T}}$:

$$\mathbf{J} = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \cdots & \dfrac{\partial f_1}{\partial x_D} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial f_K}{\partial x_1} & \cdots & \dfrac{\partial f_K}{\partial x_D} \end{bmatrix} \begin{bmatrix} r_1 \\ \vdots \\ r_D \end{bmatrix} \tag{8.67}$$

Exercise 8.18 then this can be done in single forward pass by setting $\dot{\mathbf{x}} = \mathbf{r}$.

We see that forward-mode automatic differentiation can evaluate the full $K \times D$ Jacobian matrix of derivatives using D forward passes. This is very efficient for networks with a few inputs and many outputs, such that $K \gg D$. However, we often operate in a regime where we often have just one function, namely the error function used for training, and large numbers of variables that we want to differentiate with respect to, comprising the weights and biases in the network, of which there may be millions or billions. In such situations, forward-mode automatic differentiation is extremely inefficient. We therefore turn to an alternative version of automatic differentiation based on the a backwards flow of derivative data through the evaluation trace graph.

8.2.2 Reverse-mode automatic differentiation

We can think of reverse-mode automatic differentiation as a generalization of the error backpropagation procedure. As with forward mode, we augment each intermediate variable v_i with additional variables, in this case called *adjoint* variables, denoted $\overline{v}_i$. Consider again a situation with a single output function f for which the adjoint variables are defined by

$$\overline{v}_i = \frac{\partial f}{\partial v_i}. \tag{8.68}$$

These can be evaluated sequentially starting with the output and working backwards by using the chain rule of calculus:

$$\overline{v}_i = \frac{\partial f}{\partial v_i} = \sum_{j \in \mathrm{ch}(i)} \frac{\partial f}{\partial v_j} \frac{\partial v_j}{\partial v_i} = \sum_{j \in \mathrm{ch}(i)} \overline{v}_j \frac{\partial v_j}{\partial v_i}. \tag{8.69}$$

Here $\mathrm{ch}(i)$ denotes the *children* of node i in the evaluation trace graph, in other words the set of nodes that have arrows pointing into them from node i. The successive evaluation of the adjoint variables represents a flow of information backwards through the graph, as we saw previously. *Figure 8.1*

If we again consider the specific example function given by (8.50) to (8.56), we obtain the following evaluation equations for the evaluation of the adjoint variables *Exercise 8.16*