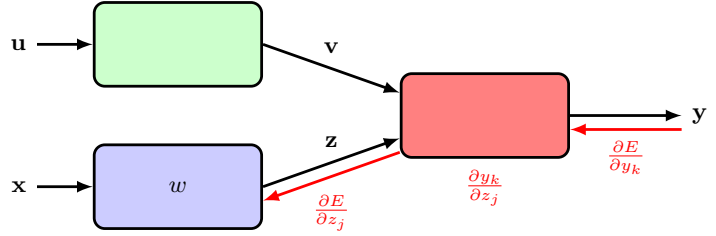


Figure 8.3 Illustration of a modular deep learning architecture in which the Jacobian matrix can be used to back-propagate error signals from the outputs through to earlier modules in the system.



associated with the inputs to be propagated through the trained network to estimate their contribution Δy_k to the errors at the outputs, through the relation

$$\Delta y_k \simeq \sum_i \frac{\partial y_k}{\partial x_i} \Delta x_i, \quad (8.28)$$

which assumes that the $|\Delta x_i|$ are small. In general, the network mapping represented by a trained neural network will be nonlinear, and so the elements of the Jacobian matrix will not be constants but will depend on the particular input vector used. Thus, (8.28) is valid only for small perturbations of the inputs, and the Jacobian itself must be re-evaluated for each new input vector.

The Jacobian matrix can be evaluated using a backpropagation procedure that is like the one derived earlier for evaluating the derivatives of an error function with respect to the weights. We start by writing the element J_{ki} in the form

$$\begin{aligned} J_{ki} = \frac{\partial y_k}{\partial x_i} &= \sum_j \frac{\partial y_k}{\partial a_j} \frac{\partial a_j}{\partial x_i} \\ &= \sum_j w_{ji} \frac{\partial y_k}{\partial a_j} \end{aligned} \quad (8.29)$$

where we have made use of (8.5). The sum in (8.29) runs over all units j to which the input unit i sends connections (for example, over all units in the first hidden layer in the layered topology considered earlier). We now write down a recursive backpropagation formula for the derivatives $\partial y_k / \partial a_j$:

$$\begin{aligned} \frac{\partial y_k}{\partial a_j} &= \sum_l \frac{\partial y_k}{\partial a_l} \frac{\partial a_l}{\partial a_j} \\ &= h'(a_j) \sum_l w_{lj} \frac{\partial y_k}{\partial a_l} \end{aligned} \quad (8.30)$$

where the sum runs over all units l to which unit j sends connections (corresponding to the first index of w_{lj}). Again, we have made use of (8.5) and (8.6). This backpropagation starts at the output units, for which the required derivatives can be found directly from the functional form of the output-unit activation function. For linear output units, we have

$$\frac{\partial y_k}{\partial a_l} = \delta_{kl} \quad (8.31)$$