

applying the back-propagation algorithm to this graph.

Algorithms 6.3 and 6.4 are demonstrations that are chosen to be simple and straightforward to understand. However, they are specialized to one specific problem.

Modern software implementations are based on the generalized form of back-propagation described in section 6.5.6 below, which can accommodate any computational graph by explicitly manipulating a data structure for representing symbolic computation.

Algorithm 6.3 Forward propagation through a typical deep neural network and the computation of the cost function. The loss $L(\hat{\mathbf{y}}, \mathbf{y})$ depends on the output $\hat{\mathbf{y}}$ and on the target \mathbf{y} (see section 6.2.1.1 for examples of loss functions). To obtain the total cost J , the loss may be added to a regularizer $\Omega(\theta)$, where θ contains all the parameters (weights and biases). Algorithm 6.4 shows how to compute gradients of J with respect to parameters \mathbf{W} and \mathbf{b} . For simplicity, this demonstration uses only a single input example \mathbf{x} . Practical applications should use a minibatch. See section 6.5.7 for a more realistic demonstration.

Require: Network depth, l

Require: $\mathbf{W}^{(i)}, i \in \{1, \dots, l\}$, the weight matrices of the model

Require: $\mathbf{b}^{(i)}, i \in \{1, \dots, l\}$, the bias parameters of the model

Require: \mathbf{x} , the input to process

Require: \mathbf{y} , the target output

$\mathbf{h}^{(0)} = \mathbf{x}$

for $k = 1, \dots, l$ **do**

$\mathbf{a}^{(k)} = \mathbf{b}^{(k)} + \mathbf{W}^{(k)} \mathbf{h}^{(k-1)}$

$\mathbf{h}^{(k)} = f(\mathbf{a}^{(k)})$

end for

$\hat{\mathbf{y}} = \mathbf{h}^{(l)}$

$J = L(\hat{\mathbf{y}}, \mathbf{y}) + \lambda \Omega(\theta)$

6.5.5 Symbol-to-Symbol Derivatives

Algebraic expressions and computational graphs both operate on **symbols**, or variables that do not have specific values. These algebraic and graph-based representations are called **symbolic** representations. When we actually use or train a neural network, we must assign specific values to these symbols. We replace a symbolic input to the network \mathbf{x} with a specific **numeric** value, such as $[1.2, 3.765, -1.8]^\top$.