CS4811 Neural Network Learning Algorithms

From: Stuart Russell and Peter Norvig
Artificial Intelligence a Modern Approach
Prentice Hall Series in Artificial Intelligence, 2003.

The following is a gradient descent learning algorithm for perceptrons, assuming a differentiable activation function g. For threshold perceptrons, the factor $g'(in)$ is omitted from the weight update. NEURAL-NET-HYPOTHESIS returns a hypothesis that computes the network output for any given example.

function PERCEPTRON-LEARNING(*examples, network*)
returns a perceptron hypothesis

inputs:
  examples, a set of examples, each with input $\mathbf{x} = x_1 \ldots, x_n$ and output y
  network, a perceptron with weights $W_j,\ j = 0, \ldots n$ and activation function g
repeat
  for each e **in** *examples* **do**
    $in \leftarrow \sum_{j=0}^{n} W_j x_j[e]$
    $err \leftarrow y[e] - g(in)$
    $W_j \leftarrow W_j + c \times Err \times g'(in) \times x_j[e]$
until some stopping criterion is satisfied
return NEURAL-NET-HYPOTHESIS(*Network*)

The following is the back propogation algorithm for learning in multilayer networks.

function BACK-PROP-LEARNING(*examples*, *network*)
returns a neural network

inputs:
    examples, a set of examples, each with input vector **x** and output vector **y**.
    network, a multilayer network with L layers, weights $W_{j,i}$, activation function g
repeat
    for each e **in** *examples* **do**
        for each node j in the input layer **do** $a_j \leftarrow x_j[e]$
        for $l = 2$ to L **do**
            $in_i \leftarrow \sum_j W_{j,i} a_j$
            $a_i \leftarrow g(in_i)$
        for each node i in the output layer **do**
            $\Delta_i \leftarrow g'(in_i) \times (y_i[e] - a_i)$
        for $l = L - 1$ **to** 1 **do**
            for each node j in layer l **do**
                $\Delta_j \leftarrow g'(in_j) \sum_i W_{j,i} \Delta_i$
                for each node i in layer $l + 1$ **do**
                    $W_{j,i} \leftarrow W_{j,i} + c \times a_j \times \Delta_i$
until some stopping criterion is satisfied
return NEURAL-NET-HYPOTHESIS(*network*)

For g, use the hyperbolic tangent: $tanh(x)$. The derivative of $tanh$ is $sech^2$.

$$tanh(x) = \frac{sinh(x)}{cosh(x)}$$

$$sech(x) = \frac{1}{sinh(x)}$$

.