Backpropagation Algorithm

// In a multilayer network, there is no given target value for the perceptrons in the
// hidden layer; hence their error values cannot be directly calculated.
// Rather, the error must be propagated backwards from the output layer and used to
// appropriately adjust the weights (to minimize error and produce correct output).

Initialize all weights to small random numbers

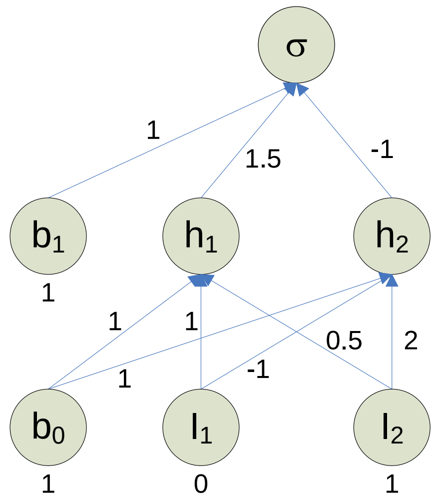
Repeat until termination condition is met:

For each training example, do

1. Feed the input forward through the network
   a. Input instance and calculate output of each unit
      // use the output function (e.g. sigmoid)
2. Propagate the errors backward through the network
   a. For each output unit *j*, calculate the error
      $E_j = (t_j - y_j)y_j(1 - y_j)$ *// note: derivative of sigmoid func*
   b. For each hidden unit *i*, calculate the error
      $E_i = h_i(1 - h_i)\sum_k w_{ik}E_k$
   c. Update each network weight
      $w_j = w_j + \eta E_j z_j$ *// connecting hidden to output*
      $w_i = w_i + \eta E_i x_i$ *// connecting input to hidden*

Multi-layer Perceptron Learning: one Epoch

Below is a snapshot of a neural network during training. There are two input units, two hidden layer perceptrons, and a single output unit. Input I_1 has a value of 0; input I_2 has a value of 1; all bias have value 1. Edges are labeled with their corresponding weights. Learning factor η = 0.5. Target value is 1.



Step 1: Feed the inputs forward

Use the formula, output = $\dfrac{1}{1+e^{-\sigma}}$ where $\sigma = \sum_i w_i x_i + bias$

h_1 = (1 • 0) + (0.5 • 1) + (1 • 1) = 1.5 ⇨ 1/(1+$e^{-1.5}$) = 0.818

h_2 = (-1 • 0) + (2 • 1) + (1 • 1) = 3 ⇨ 1/(1+e^{-3}) = 0.953

y = (1.5 • 0.818) + (-1 • 0.953) + (1 • 1) = 1.274 ⇨ 1/(1+$e^{-1.274}$) = 0.781

Calculate total error in network, $E = \frac{1}{2}(t-y)^2$

E = ½ (1 – 0.781)^2 = 0.024

Step 2: <u>Backpropagate the errors</u>

a) Calculate the error for the output unit *y*,
Use the formula, $E_y = y(1-y)(t-y)$

$E_y = (0.781)(1 – 0.781)(1 – 0.781) = 0.037$

b) Calculate the error for each hidden unit h_i
Use the formula, $E_{h_i} = h_i(1-h_i)(w_{h_i,y} \cdot E_y)$

$E_{h1} = (0.818)(1 – 0.818)(1.5 \bullet 0.037) = 0.008$
$E_{h2} = (0.953)(1 – 0.953)(-1 \bullet 0.037) = -0.002$

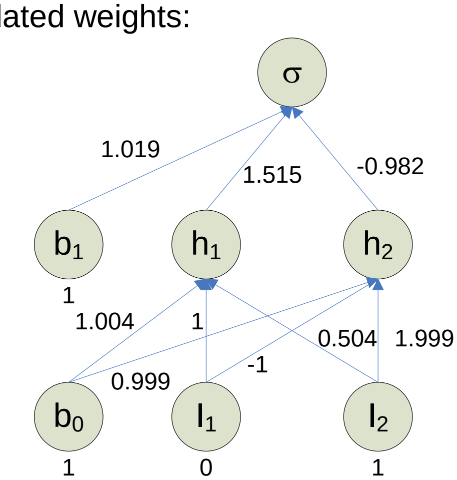
c) Update network weights proportionately
Use the formula, $w_{i,j} = w_{i,j} + \eta E_j z_i$ where z_i is value of *i*

$w_{h1,y} = 1.5 + (0.5)(0.037)(0.818) = 1.515$
$w_{h2,y} = -1 + (0.5)(0.037)(0.953) = -0.982$
$w_{b1,y} = 1 + (0.5)(0.037)(1) = 1.019$

$w_{I1, h1} = 1 + (0.5)(0.008)(0) = 1$
$w_{I2, h1} = 0.5 + (0.5)(0.008)(1) = 0.504$
$w_{b0, h1} = 1 + (0.5)(0.008)(1) = 1.004$

$w_{I1, h2} = -1 + (0.5)(-0.002)(0) = -1$
$w_{I2, h2} = 2 + (0.5)(-0.002)(1) = 1.999$
$w_{b0, h2} = 1 + (0.5)(-0.002)(1) = 0.999$

Label network with updated weights:



Repeat for all training samples ⇨ one epoch.