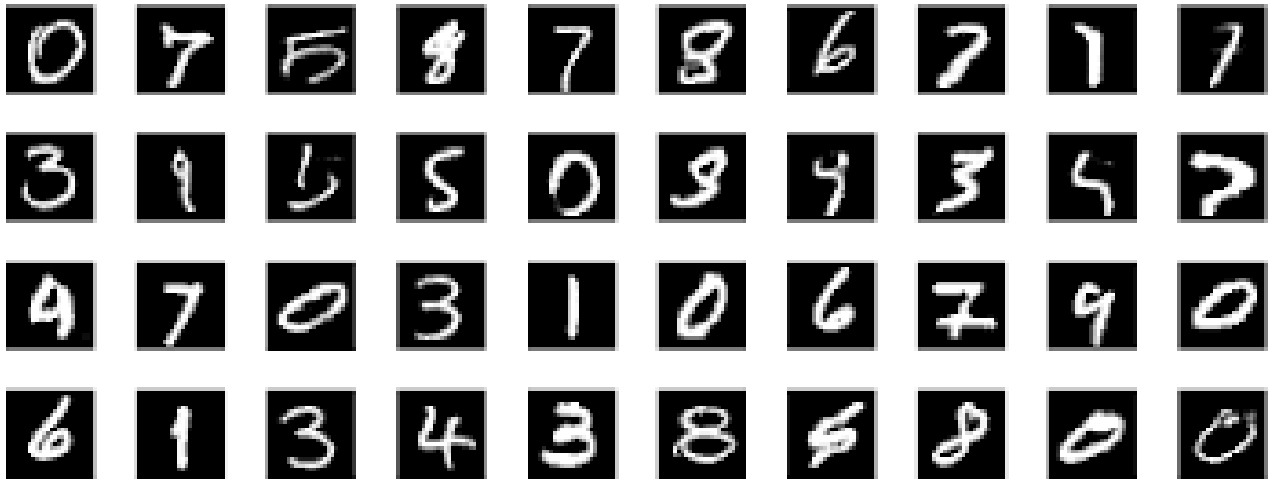
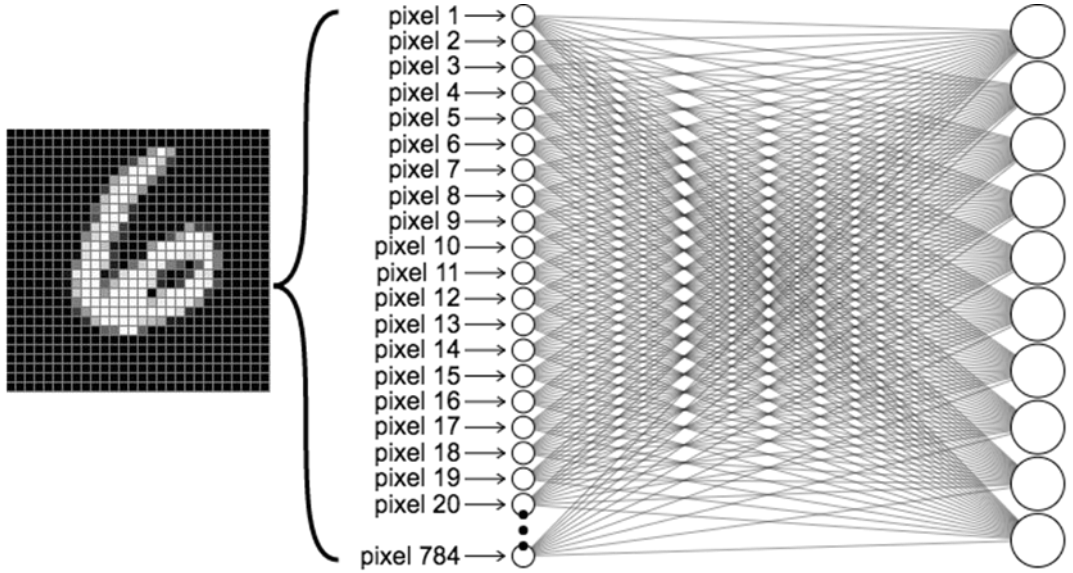
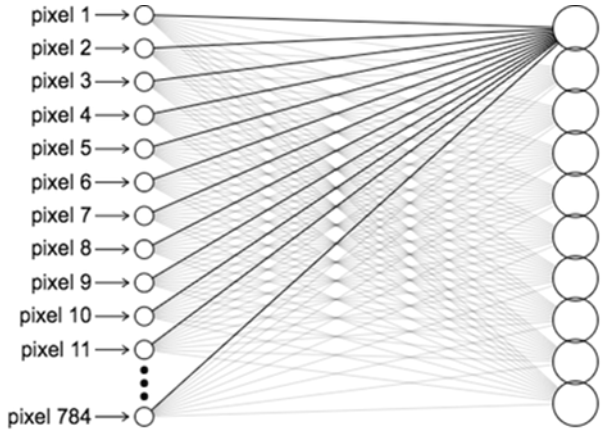
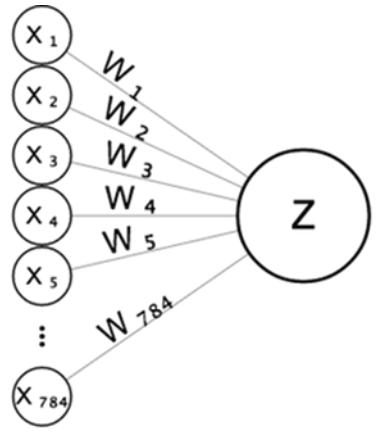
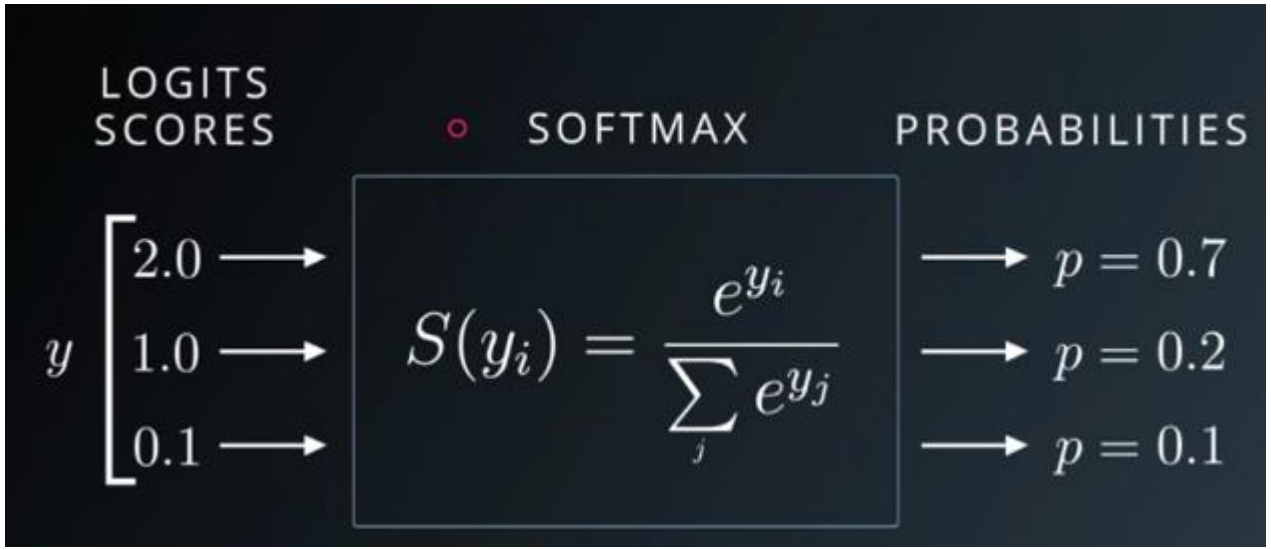
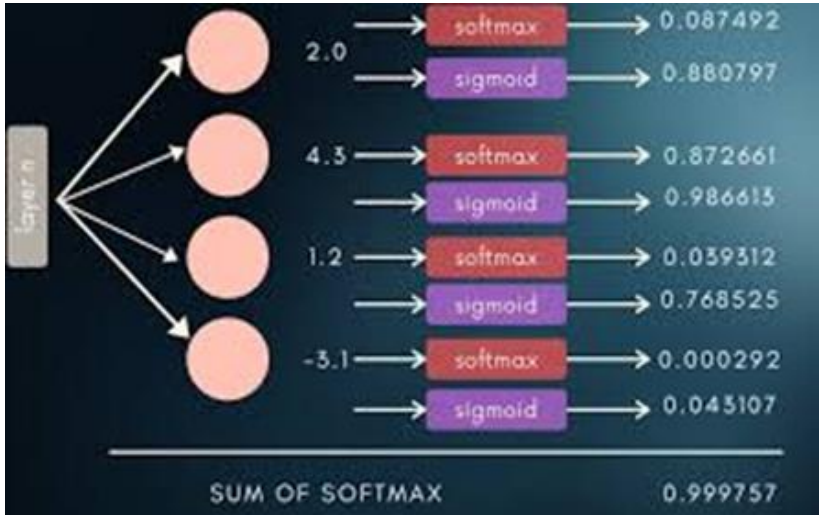


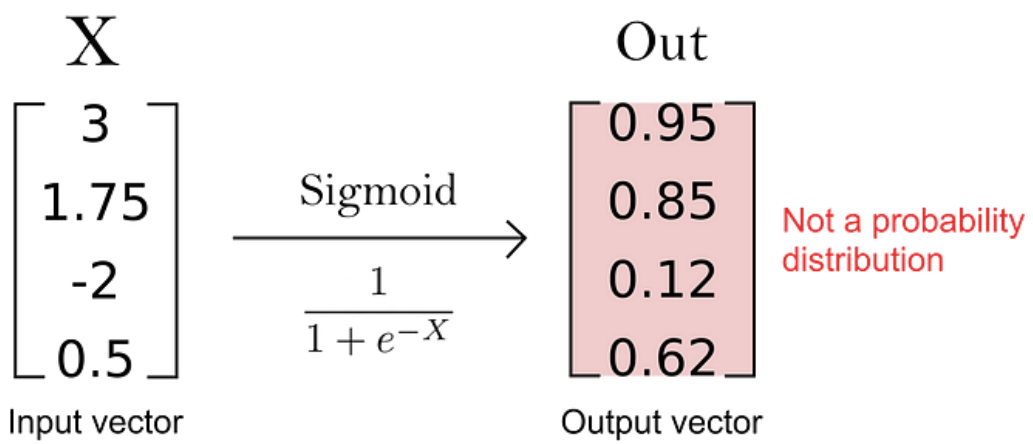

pixel 1
pixel 2
pixel 3
pixel 4
pixel 5
pixel 6
pixel 7
pixel 8
pixel 9
pixel 10
pixel 11
pixel 12
pixel 13
pixel 14
pixel 15
pixel 16
pixel 17
pixel 18
pixel 19
pixel 20
pixel 784

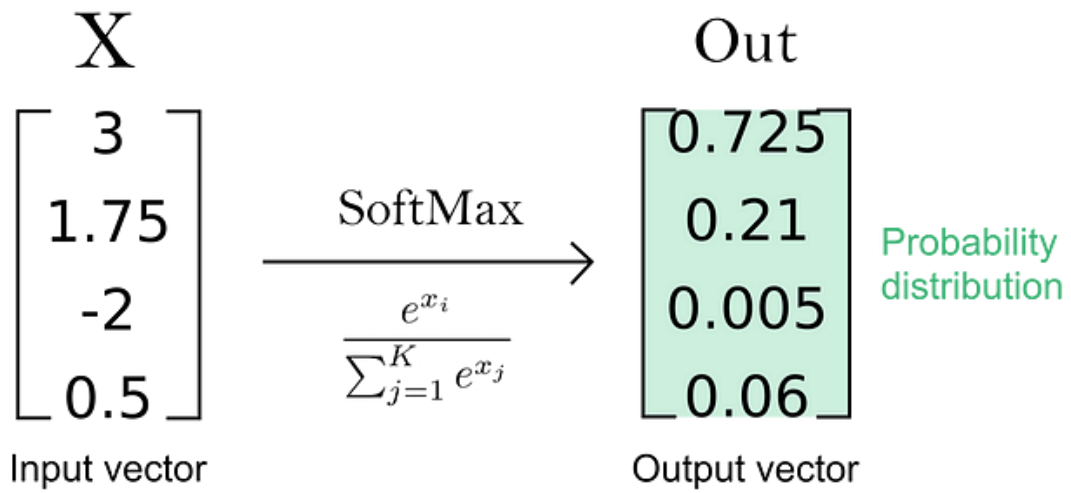

pixel 1
pixel 2
pixel 3
pixel 4
pixel 5
pixel 6
pixel 7
pixel 8
pixel 9
pixel 10
pixel 11
pixel 784


x 1
x 2
x 3
x 4
x 5
x 784
w 1
w 2
w 3
w 4
w 5
w 784
z



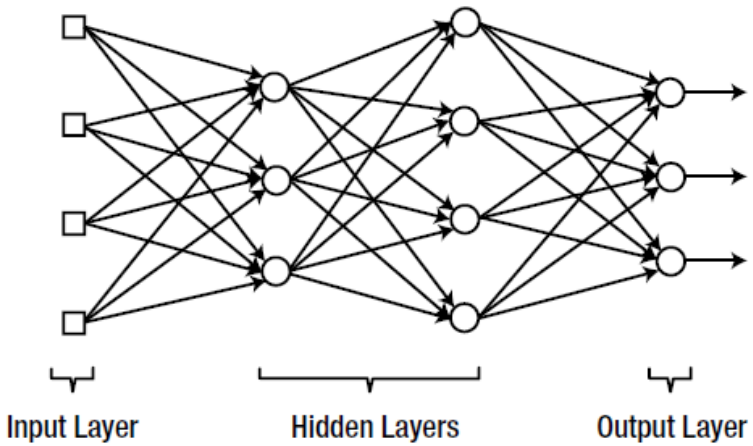


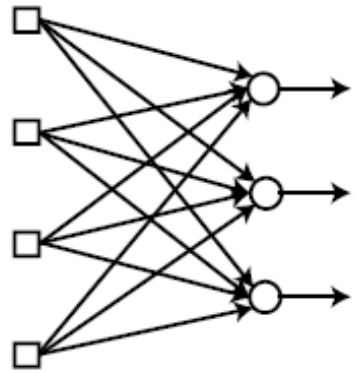




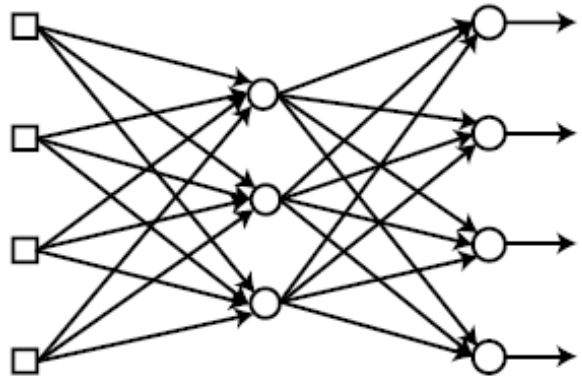
Sigmoid is used for **binary classification** methods where we only have 2 classes, while SoftMax applies to **multiclass problems**.

ANN with multiple layers

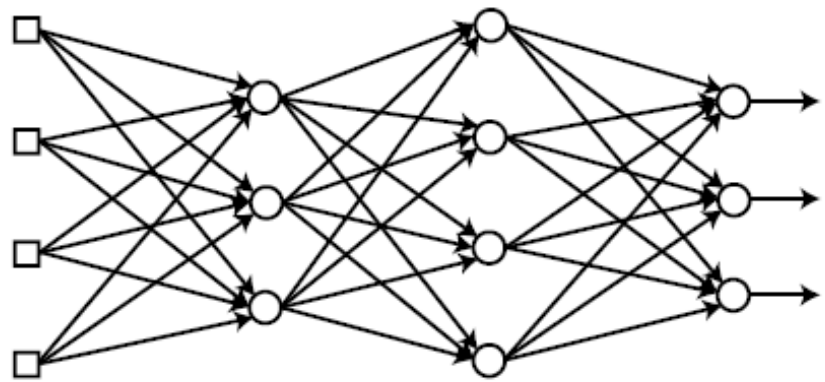




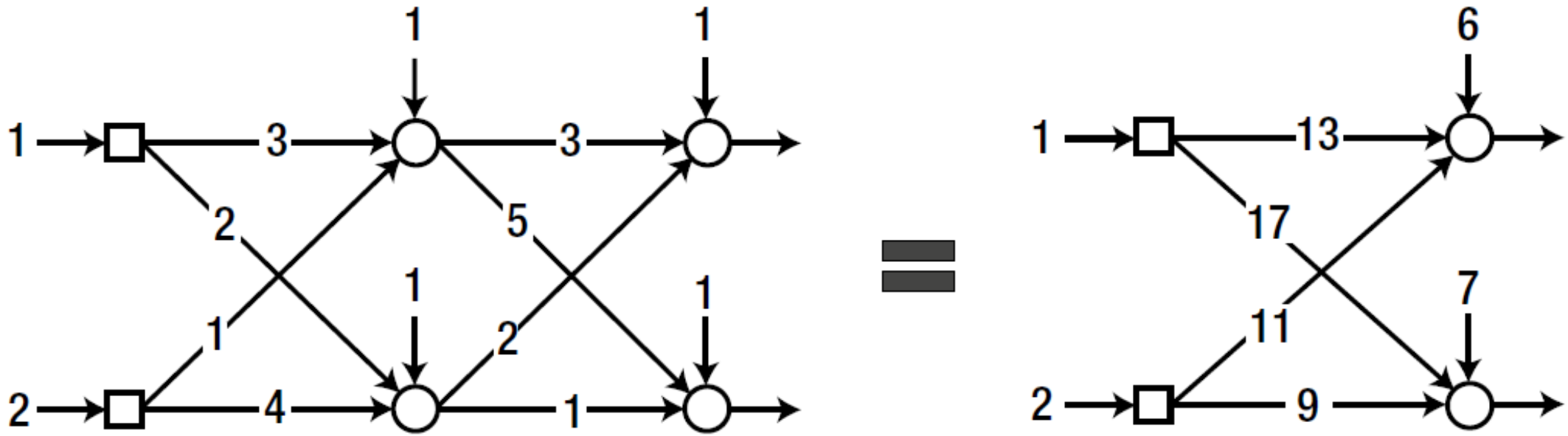
Single-layer Neural Network



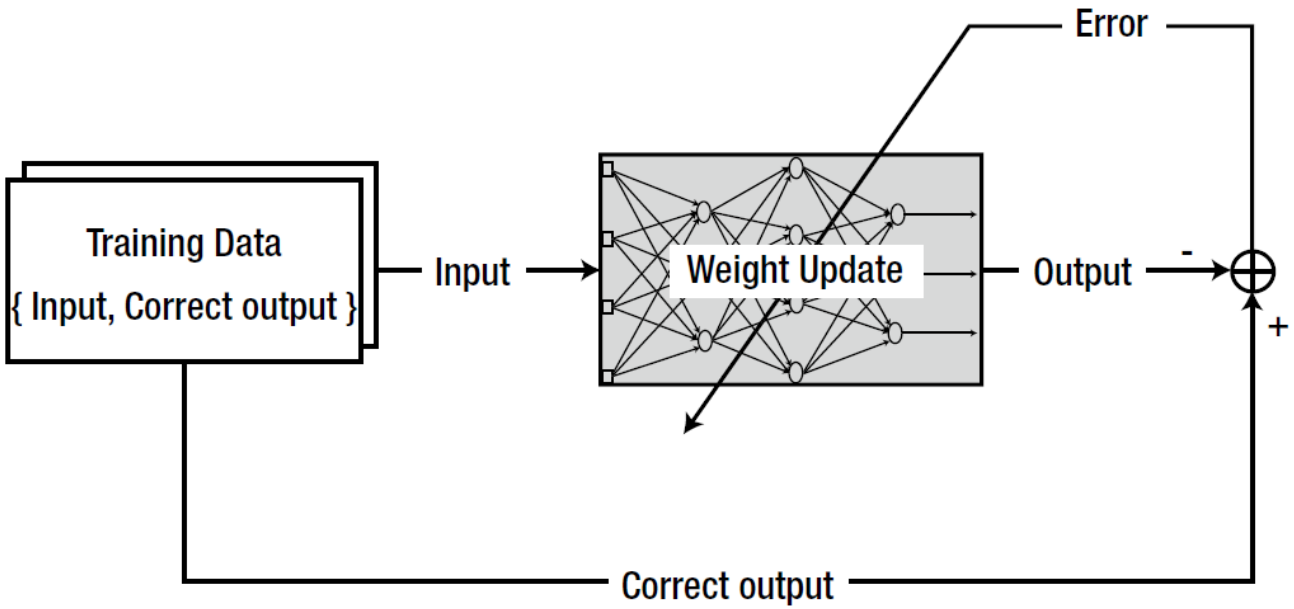
(Shallow) Multi-layer Neural Network



Deep Neural Network

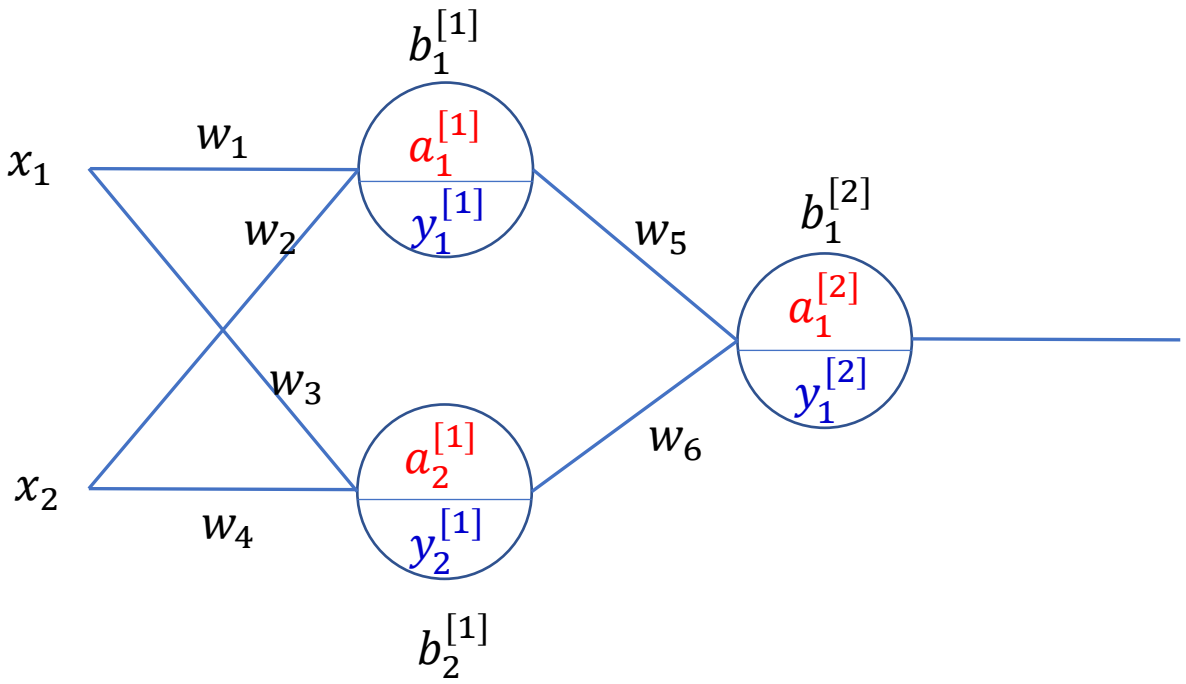


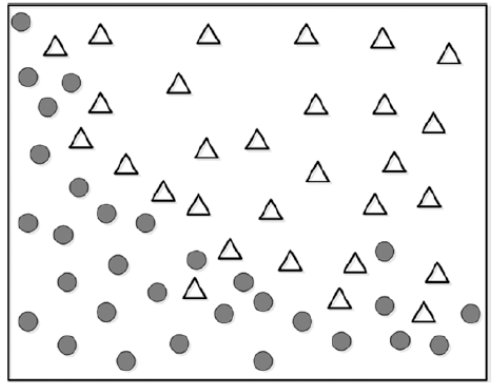
Keep in mind that the hidden layer becomes ineffective when the hidden nodes have linear activation functions.

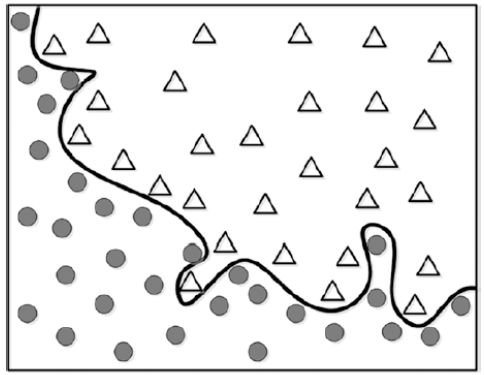


1. Initialize the weights with adequate values.
2. Take the "input" from the training data, which is formatted as `{ input, correct output }`, and enter it into the neural network. Obtain the output from the neural network and calculate the error from the correct output.
3. Adjust the weights to reduce the error.
4. Repeat Steps 2-3 for all training data

ANN with multiple layers







$a_1^{[1]} = x_1\, w_1 + x_2\, w_2 + b_1^{[1]}$

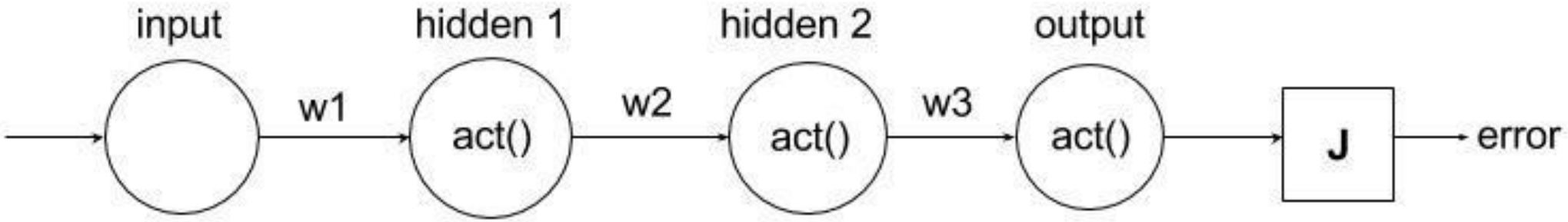
$y_1^{[1]} = \sigma(a_1^{[1]})$

$a_2^{[1]} = x_1\, w_3 + x_2\, w_4 + b_2^{[1]}$

$y_2^{[1]} = \sigma(a_2^{[1]})$

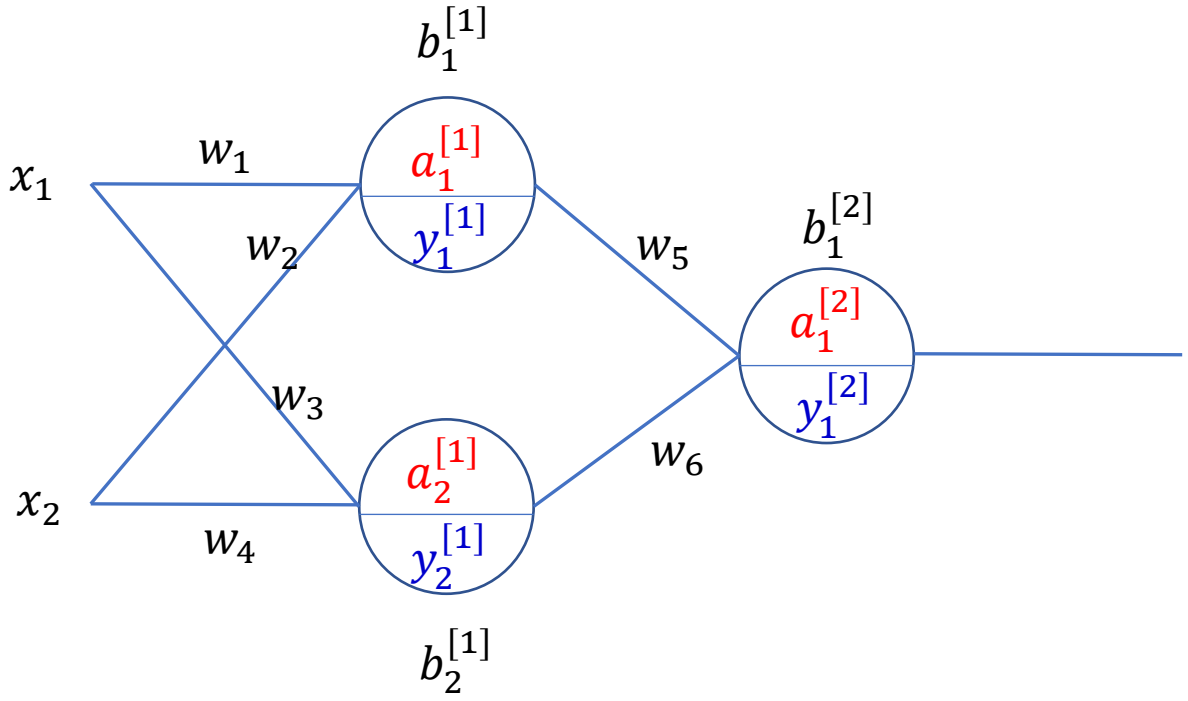
$a_1^{[2]} = y_1^{[1]}\, w_5 + y_2^{[1]}\, w_6 + b_1^{[2]}$

$y_1^{[2]} = \sigma(a_1^{[2]})$



$$\frac{\partial(error)}{\partial w_1} = \frac{\partial(error)}{\partial(output)} * \frac{\partial(output)}{\partial(hidden\ 2)} * \frac{\partial(hidden\ 2)}{\partial(hidden\ 1)} * \frac{\partial(hidden\ 1)}{\partial w_1}$$

Backpropagation (“backprop” for short) is a way of computing the partial derivatives of a loss function with respect to the parameters of a network.



$$w_5 = w_5 - \alpha \frac{\partial E}{\partial w5}$$

$$w_1 = w_1 - \alpha \frac{\partial E}{\partial w1}$$

$$E = \frac{1}{2}(t - y_1^{[2]})^2$$

$$y_1^{[2]} = \sigma(a_1^{[2]})$$

$$\frac{\partial E}{\partial w5} = \frac{\partial E}{\partial y_1^{[2]}} \frac{\partial y_1^{[2]}}{\partial a_1^{[2]}} \frac{\partial a_1^{[2]}}{\partial w5}$$

$$\frac{\partial E}{\partial y_1^{[2]}} = \frac{\partial (\frac{1}{2}(t - y_1^{[2]})^2)}{\partial y_1^{[2]}} = -(t - y_1^{[2]})$$

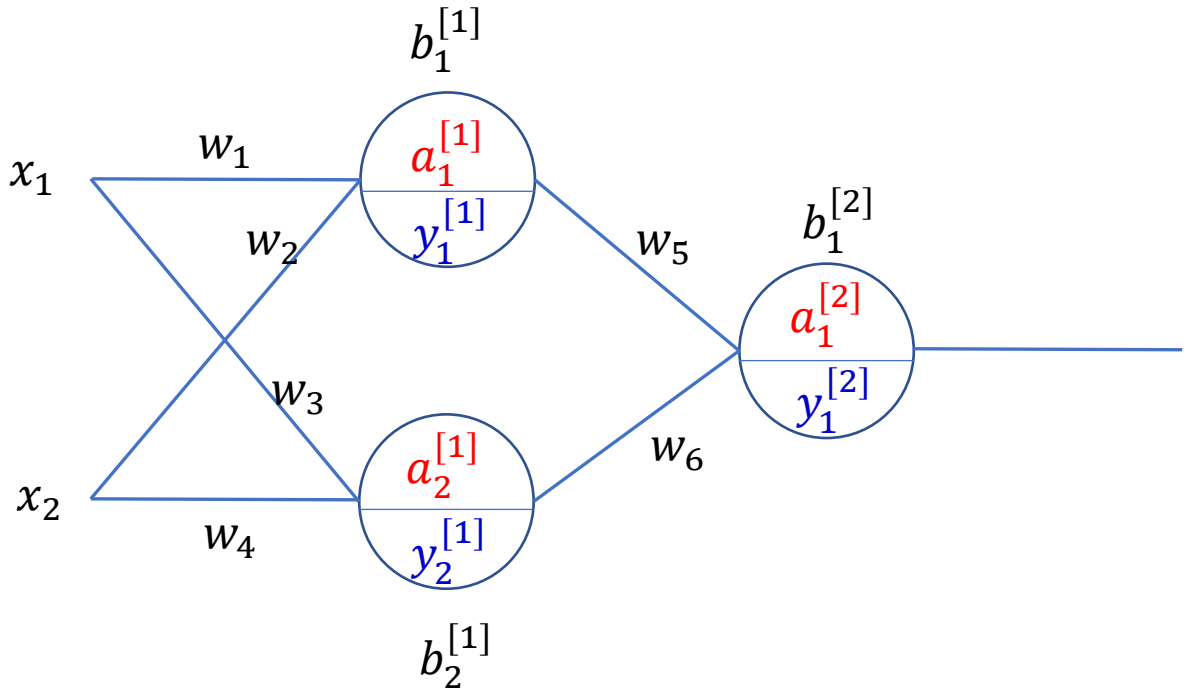
$$\frac{\partial y_1^{[2]}}{\partial a_1^{[2]}} = \frac{\partial \sigma(a_1^{[2]})}{\partial a_1^{[2]}} = \sigma(a_1^{[2]})\,(1 - \sigma(a_1^{[2]})) = y_1^{[2]}(1 - y_1^{[2]})$$

$$\frac{\partial a_1^{[2]}}{\partial w5} = \frac{\partial (y_1^{[1]} w_5 + y_2^{[1]} w_6 + b_1^{[2]})}{\partial w5} = y_1^{[1]}$$

$$\frac{\partial E}{\partial w5} = -(t - y_1^{[2]}) y_1^{[2]} (1 - y_1^{[2]})\, y_1^{[1]}$$

$$w_5 = w_5 + \alpha\,(t - y_1^{[2]}) y_1^{[2]} (1 - y_1^{[2]})\, y_1^{[1]}$$

$$w_5 = w_5 + \alpha\, \delta_1^{[2]} y_1^{[1]}$$



$$w_1 = w_1 - \alpha \frac{\partial E}{\partial w1}$$

$$E = \frac{1}{2}(t - y_1^{[2]})^2$$

$$y_1^{[2]} = \sigma(a_1^{[2]})$$

$$\frac{\partial E}{\partial w1} = \frac{\partial E}{\partial y_1^{[2]}} \frac{\partial y_1^{[2]}}{\partial y_1^{[1]}} \frac{\partial y_1^{[1]}}{\partial w1}$$

$$\frac{\partial E}{\partial y_1^{[2]}} = \frac{\partial (\frac{1}{2}(t - y_1^{[2]})^2)}{\partial y_1^{[2]}} = -(t - y_1^{[2]})$$

$$\frac{\partial y_1^{[2]}}{\partial y_1^{[1]}} = \frac{\partial y_1^{[2]}}{\partial a_1^{[2]}} \frac{\partial a_1^{[2]}}{\partial y_1^{[1]}}$$

$$\frac{\partial y_1^{[2]}}{\partial a_1^{[2]}} = y_1^{[2]}(1 - y_1^{[2]})$$

$$\frac{\partial a_1^{[2]}}{\partial y_1^{[1]}} = w_5$$

$$\frac{\partial y_1^{[2]}}{\partial y_1^{[1]}} = y_1^{[2]}(1 - y_1^{[2]})\, w_5$$

$$\frac{\partial y_1^{[1]}}{\partial w1} = \frac{\partial y_1^{[1]}}{\partial a_1^{[1]}} \frac{\partial a_1^{[1]}}{\partial w1}$$

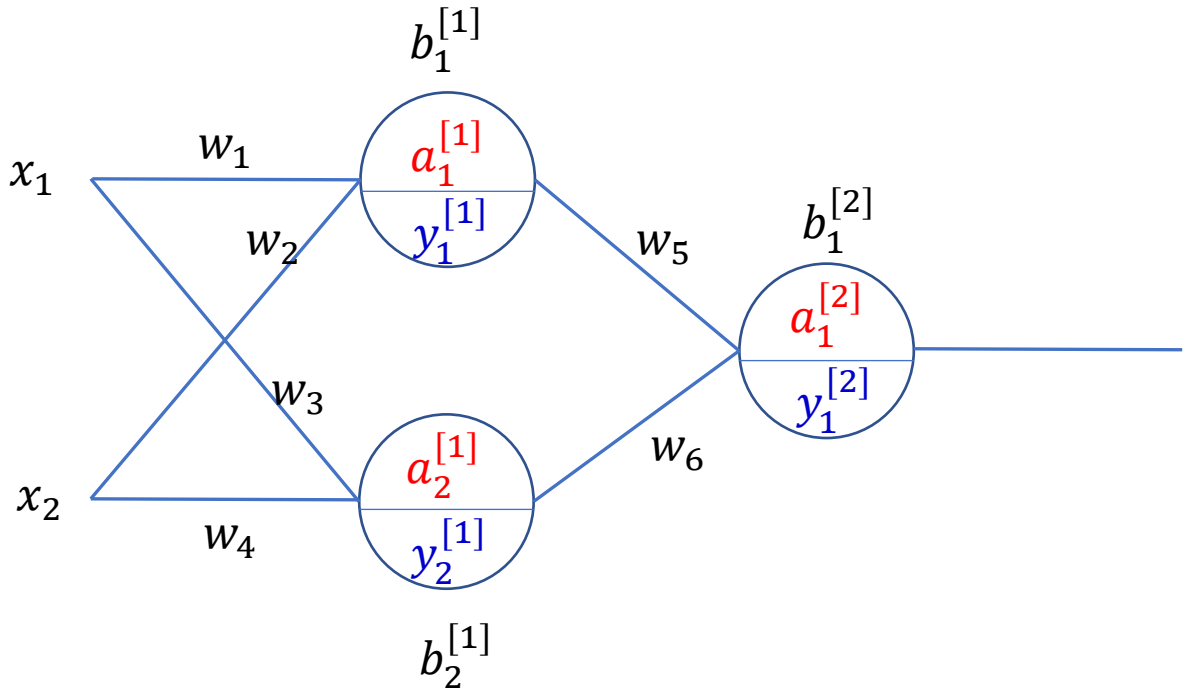
$$\frac{\partial y_1^{[1]}}{\partial a_1^{[1]}} = y_1^{[1]}(1 - y_1^{[1]})$$

$$\frac{\partial a_1^{[1]}}{\partial w1} = \frac{\partial (x_1 w_1 + x_2 w_2 + b_1^{[1]})}{\partial w1} = x_1$$

$$\frac{\partial y_1^{[1]}}{\partial w1} = y_1^{[1]}(1 - y_1^{[1]})\, x_1$$

$$\frac{\partial E}{\partial w1} = -(t - y_1^{[2]}) y_1^{[2]}(1 - y_1^{[2]}) w_5 y_1^{[1]}(1 - y_1^{[1]}) x_1$$

$$w_1 = w_1 + \alpha\,(t - y_1^{[2]}) y_1^{[2]}(1 - y_1^{[2]}) w_5 y_1^{[1]}(1 - y_1^{[1]}) x_1$$



$$w_5 = w_5 + \alpha\,\delta^{[2]} y_1^{[1]}$$

$$w_6 = w_6 + \alpha\,\delta^{[2]} y_2^{[1]}$$

$$\delta^{[2]} = (t - y_1^{[2]}) y_1^{[2]} (1 - y_1^{[2]})$$

$$w_1 = w_1 + \alpha\,\delta^{[1]} x_1$$

$$\delta^{[1]} = \delta^{[2]} w_5\, y_1^{[1]} (1 - y_1^{[1]})$$

$$w_{j,l} = w_{j,l} + \alpha\,\delta^{[l]} y_{l-1}$$

$$w_5 = w_5 + \alpha\,(t - y_1^{[2]}) y_1^{[2]} (1 - y_1^{[2]})\, y_1^{[1]}$$

$$w_6 = w_6 + \alpha\,(t - y_1^{[2]}) y_1^{[2]} (1 - y_1^{[2]})\, y_2^{[1]}$$

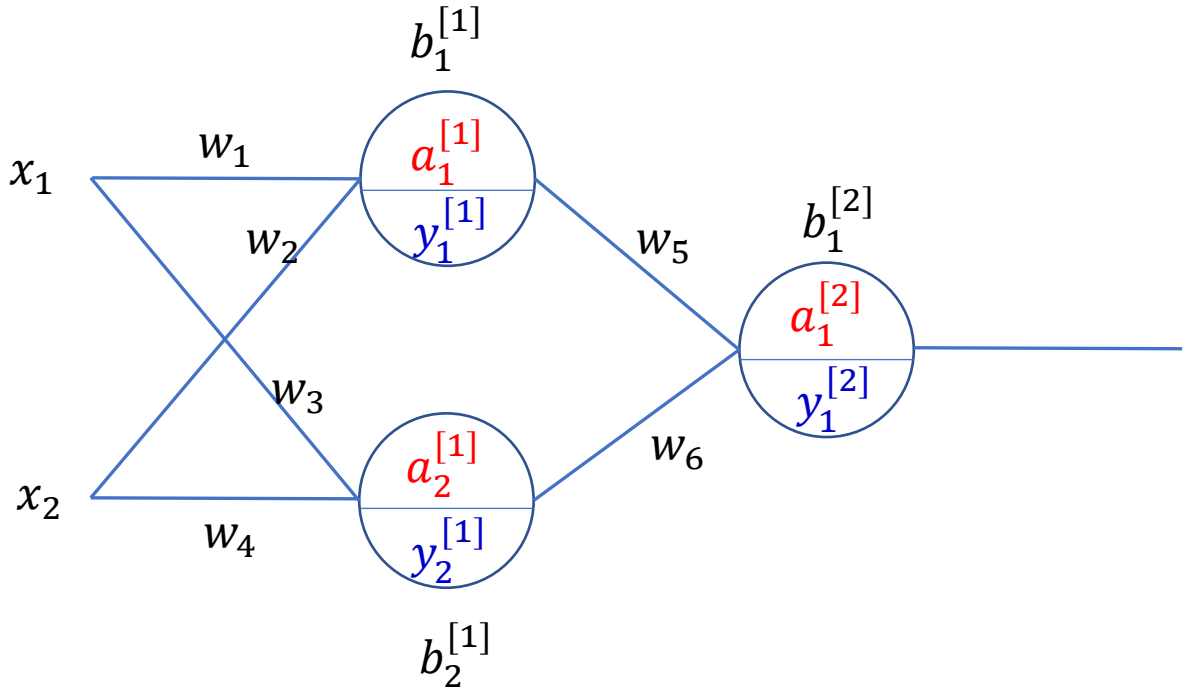
$$w_1 = w_1 + \alpha\,(t - y_1^{[2]}) y_1^{[2]} (1 - y_1^{[2]}) w_5 y_1^{[1]} (1 - y_1^{[1]}) x_1$$

$$w_2 = w_2 + \alpha\,(t - y_1^{[2]}) y_1^{[2]} (1 - y_1^{[2]}) w_5 y_1^{[1]} (1 - y_1^{[1]}) x_2$$

$$w_3 = w_3 + \alpha\,(t - y_1^{[2]}) y_1^{[2]} (1 - y_1^{[2]}) w_6 y_2^{[1]} (1 - y_2^{[1]}) x_1$$

$$w_4 = w_4 + \alpha\,(t - y_1^{[2]}) y_1^{[2]} (1 - y_1^{[2]}) w_6 y_2^{[1]} (1 - y_2^{[1]}) x_2$$

$$\delta^{[i]} = \begin{cases} (t - y_j^{[i]}) y_j^{[i]} (1 - y_j^{[i]}) & \text{if j is an output neuron} \\ \left(\sum_{l \in L} \delta^{[l]} w_{jl} \right) y_j^{[l]} (1 - y_j^{[l]}) & \text{if j is an inner neuron} \end{cases}$$



$$E = -(t\log(y) + (1-t)\log(1-y))$$

$$w_1 = w_1 - \alpha \frac{\partial E}{\partial w1}$$

$$y_1^{[2]} = \sigma(a_1^{[2]})$$

$$\frac{\partial E}{\partial w1} = \frac{\partial E}{\partial y_1^{[2]}} \frac{\partial y_1^{[2]}}{\partial y_1^{[1]}} \frac{\partial y_1^{[1]}}{\partial w1}$$

$$\frac{\partial E}{\partial y_1^{[2]}} = \frac{\partial(t\log y_1^{[2]} + (1-t)\log\left(1-y_1^{[2]}\right))}{\partial y_1^{[2]}}$$

$$\frac{\partial E}{\partial y} = -\left(\frac{t}{y_1^{[2]}} - \frac{(1-t)}{\left(1-y_1^{[2]}\right)}\right) = \frac{y_1^{[2]} - t}{y_1^{[2]}\left(1-y_1^{[2]}\right)}$$

$$\frac{\partial y_1^{[2]}}{\partial y_1^{[1]}} = \frac{\partial y_1^{[2]}}{\partial a_1^{[2]}} \frac{\partial a_1^{[2]}}{\partial y_1^{[1]}}$$

$$\frac{\partial y_1^{[2]}}{\partial a_1^{[2]}} = y_1^{[2]}(1 - y_1^{[2]})$$

$$\frac{\partial a_1^{[2]}}{\partial y_1^{[1]}} = w_5$$

$$\frac{\partial y_1^{[2]}}{\partial y_1^{[1]}} = y_1^{[2]}(1 - y_1^{[2]})\, w_5$$

$$\frac{\partial y_1^{[1]}}{\partial w1} = \frac{\partial y_1^{[1]}}{\partial a_1^{[1]}} \frac{\partial a_1^{[1]}}{\partial w1}$$

$$\frac{\partial y_1^{[1]}}{\partial a_1^{[1]}} = y_1^{[1]}(1 - y_1^{[1]})$$

$$\frac{\partial a_1^{[1]}}{\partial w1} = \frac{\partial(x_1\, w_1 + x_2\, w_2 + b_1^{[1]})}{\partial w1} = x_1$$

$$\frac{\partial y_1^{[1]}}{\partial w1} = y_1^{[1]}(1 - y_1^{[1]})\, x_1$$

$$\frac{\partial E}{\partial w1} = \frac{y_1^{[2]} - t}{y_1^{[2]}\left(1-y_1^{[2]}\right)} y_1^{[2]}(1 - y_1^{[2]}) w_5 y_1^{[1]}(1 - y_1^{[1]}) x_1$$

$$w_1 = w_1 + \alpha\,(t - y_1^{[2]})\, w_5 y_1^{[1]}(1 - y_1^{[1]}) x_1$$

Backpropogation algorithm

Phase 1: propagation

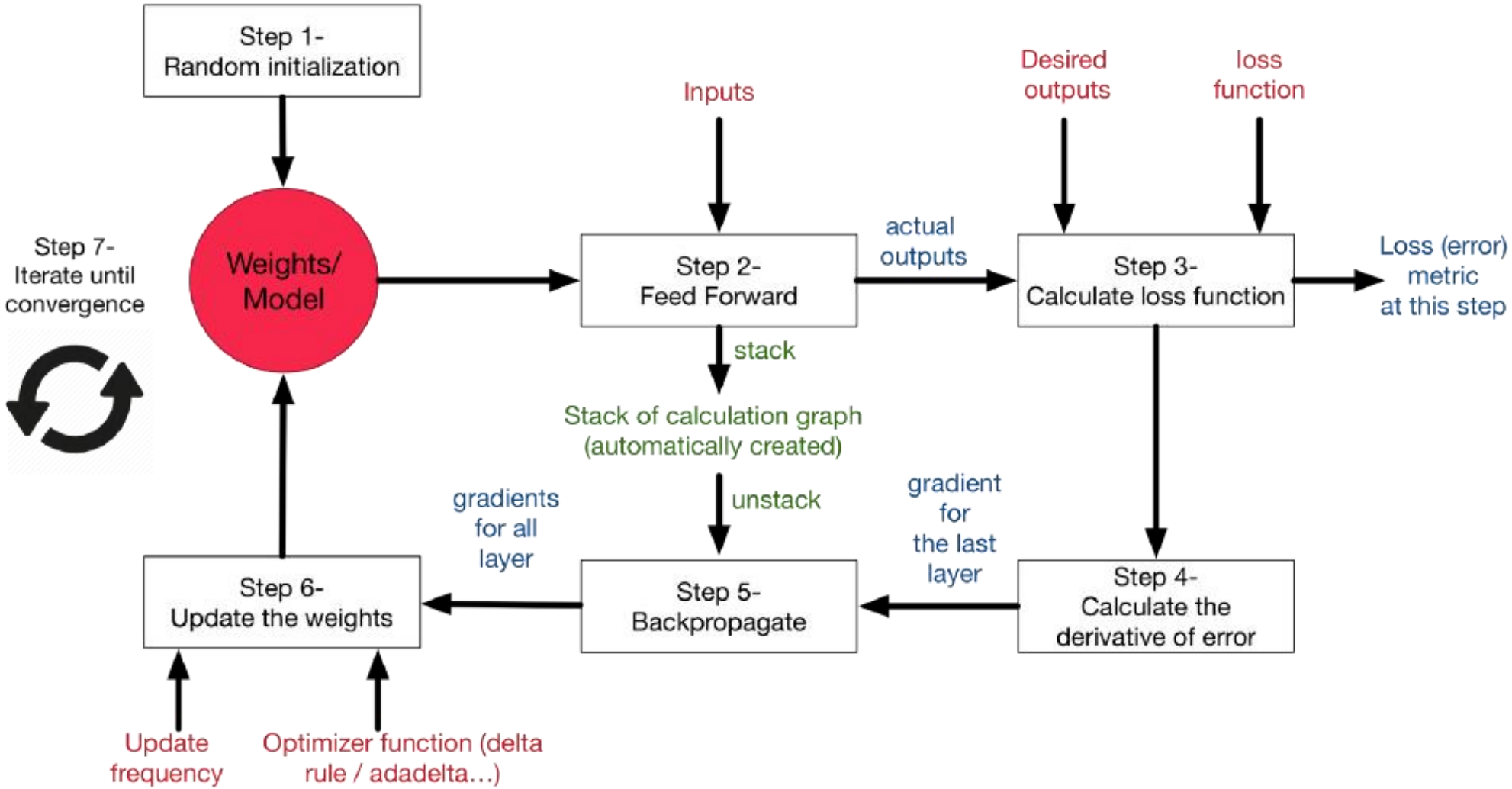
Each propagation involves the following steps:

1. Propagation forward through the network to generate the output value(s)
2. Calculation of the cost (error term)
3. Propagation of the output activations back through the network using the training pattern target in order to generate the deltas (the difference between the targeted and actual output values) of all output and hidden neurons.

Phase 2: weight update

For each weight, the following steps must be followed:

1. The weight's output delta and input activation are multiplied to find the gradient of the weight.
2. A ratio (percentage) of the weight's gradient is subtracted from the weight.



https://medium.com/datathings/neural-networks-and-backpropagation-explained-in-a-simple-way-f540a3611f5e

Historical and bibliographical remarks

Modern version of BP (also called automatic differentiation) was first published in 1970 by Finnish master student Seppo Linnainmaa.

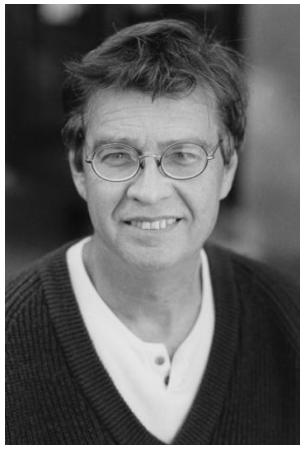
First NN-specific application of efficient BP was described by Werbos (1982).

Related work was published several years later (Parker, 1985; LeCun, 1985).

A paper of 1986 significantly contributed to the popularisation of BP for NNs (Rumelhart et al., 1986), experimentally demonstrating the emergence of useful internal representations in hidden layers.

Seppo Linnainmaa 

Paul Werbos 

David Rumelhart 

http://people.idsia.ch/~juergen/who-invented-backpropagation.html

Learning representations by back-propagating errors

David E. Rumelhart*, Geoffrey E. Hinton† & Ronald J. Williams*

* Institute for Cognitive Science, C-015, University of California, San Diego, La Jolla, California 92093, USA
† Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Philadelphia 15213, USA

We describe a new learning procedure, back-propagation, for networks of neurone-like units. The procedure repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector. As a result of the weight adjustments, internal 'hidden' units which are not part of the input or output come to represent important features of the task domain, and the regularities in the task are captured by the interactions of these units. The ability to create useful new features distinguishes back-propagation from earlier, simpler methods such as the perceptron-convergence procedure[1].

There have been many attempts to design self-organizing neural networks. The aim is to find a powerful synaptic modification rule that will allow an arbitrarily connected neural network to develop an internal structure that is appropriate for a particular task domain. The task is specified by giving the desired state vector of the output units for each state vector of the input units. If the input units are directly connected to the output units it is relatively easy to find learning rules that iteratively adjust the relative strengths of the connections so as to progressively reduce the difference between the actual and desired output vectors[2]. Learning becomes more interesting but more difficult when we introduce hidden units whose actual or desired states are not specified by the task. (In perceptrons, there are 'feature analysers' between the input and output that are not true hidden units because their input connections are fixed by hand, so their states are completely determined by the input vector: they do not learn representations.) The learning procedure must decide under what circumstances the hidden units should be active in order to help achieve the desired input–output behaviour. This amounts to deciding what these units should represent. We demonstrate that a general purpose and relatively simple procedure is powerful enough to construct appropriate internal representations.

The simplest form of the learning procedure is for layered networks which have a layer of input units at the bottom; any number of intermediate layers; and a layer of output units at the top. Connections within a layer or from higher to lower layers are forbidden, but connections can skip intermediate layers. An input vector is presented to the network by setting the states of the input units. Then the states of the units in each layer are determined by applying equations (1) and (2) to the connections coming from lower layers. All units within a layer have their states set in parallel, but different layers have their states set sequentially, starting at the bottom and working upwards until the states of the output units are determined.

The total input, x_j, to unit j is a linear function of the outputs, y_i, of the units that are connected to j and of the weights, w_{ji}, on these connections

$$x_j = \sum_i y_i w_{ji} \tag{1}$$

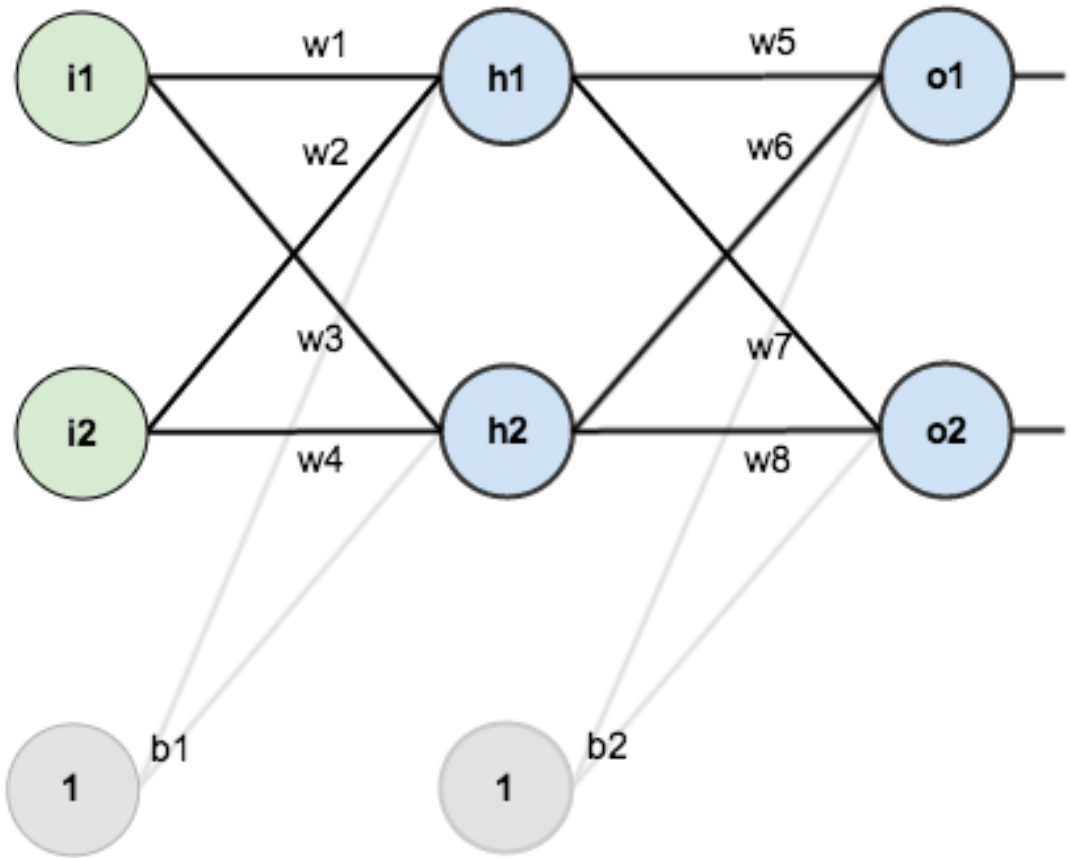
Units can be given biases by introducing an extra input to each unit which always has a value of 1. The weight on this extra input is called the bias and is equivalent to a threshold of the opposite sign. It can be treated just like the other weights.

A unit has a real-valued output, y_j, which is a non-linear function of its total input

$$y_j = \frac{1}{1+e^{-x_j}} \tag{2}$$

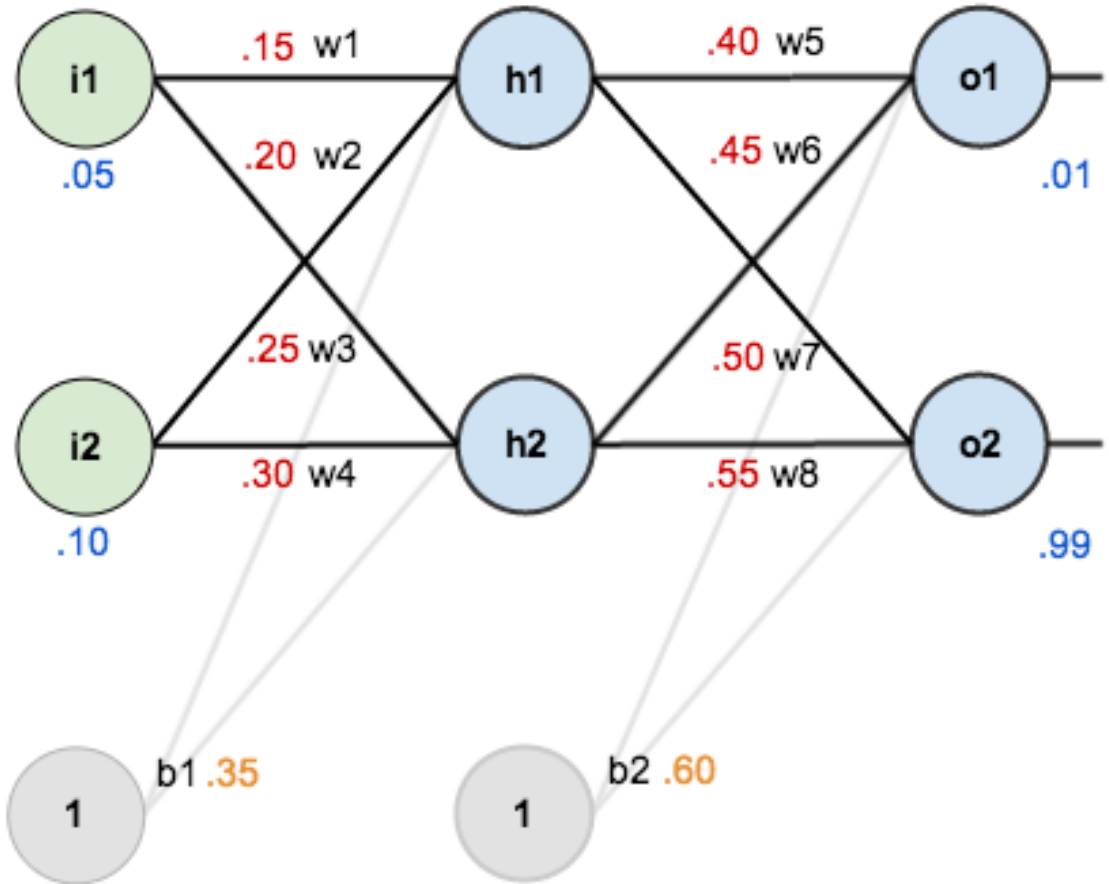
† To whom correspondence should be addressed.

Backpropagation Example



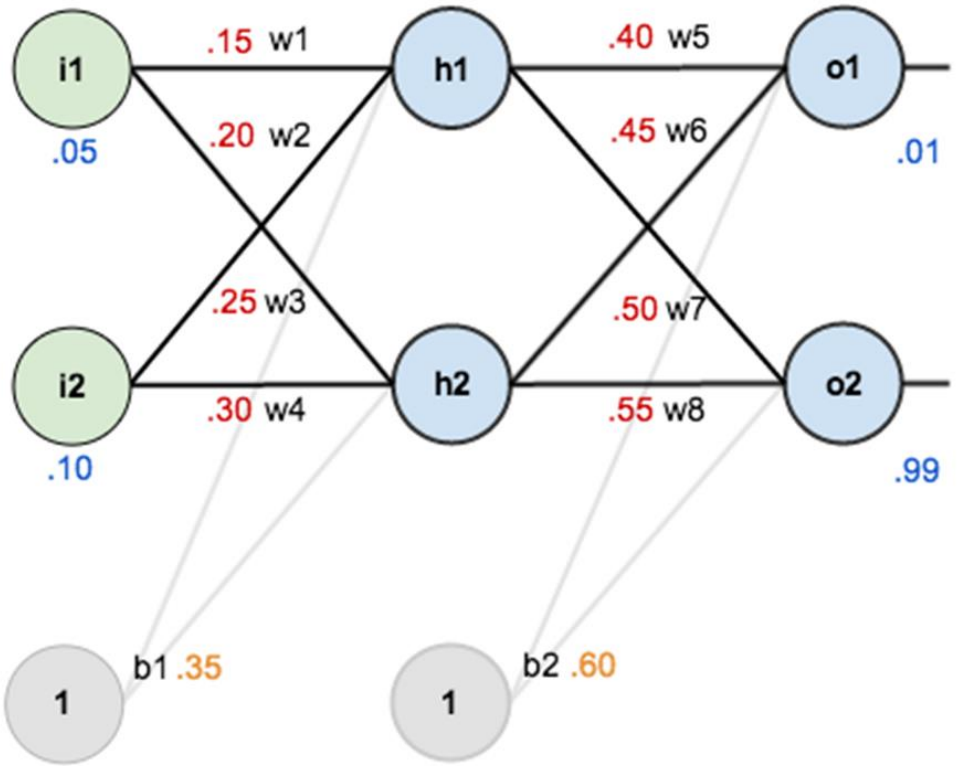
i – input layer
h – hidden layer
o – output layer
w – weights
b - bias
Activation function : sigmoid
Loss function : mse
Learning rate : 0.5

https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/



Here are the initial weights, the biases, and training inputs/outputs:

Given inputs are 0.05 and 0.10, we want the neural network to output 0.01 and 0.99.



The Forward Pass

Weighted sum in h1 $net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$

$net_{h1} = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775$

Output of h1 $out_{h1} = \frac{1}{1+e^{-net_{h1}}} = \frac{1}{1+e^{-0.3775}} = 0.593269992$

Output of o1 $net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$

$net_{o1} = 0.4 * 0.593269992 + 0.45 * 0.596884378 + 0.6 * 1 = 1.105905967$

Output of o1 $out_{o1} = \frac{1}{1+e^{-net_{o1}}} = \frac{1}{1+e^{-1.105905967}} = 0.75136507$

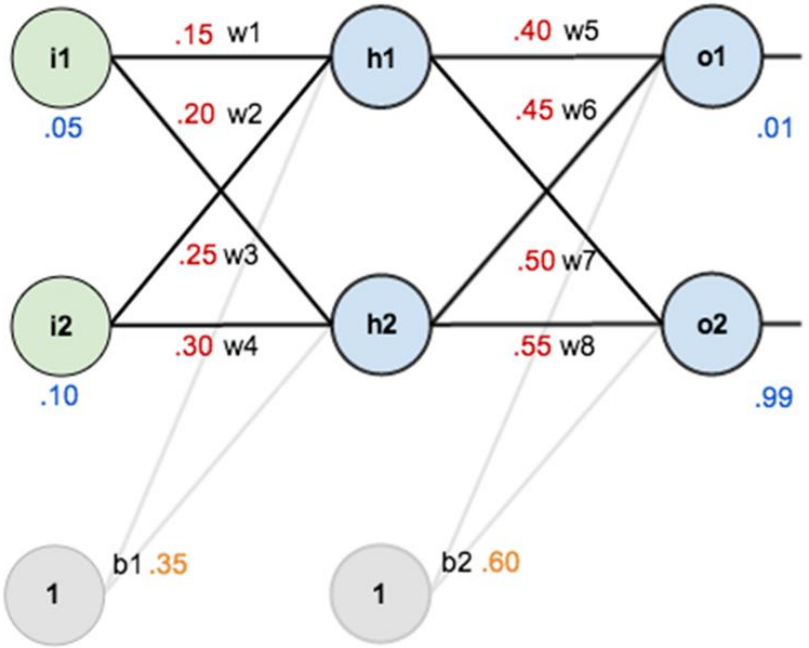
Output of o2 $out_{o2} = 0.772928465$

Error calculation : $E_{total} = \sum \frac{1}{2}(target - output)^2$

$E_{o1} = \frac{1}{2}(target_{o1} - out_{o1})^2 = \frac{1}{2}(0.01 - 0.75136507)^2 = 0.274811083$

$E_{o2} = 0.023560026$

$E_{total} = E_{o1} + E_{o2} = 0.274811083 + 0.023560026 = 0.298371109$



The Backward Pass

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

$$E_{total} = \frac{1}{2}(target_{o1} - out_{o1})^2 + \frac{1}{2}(target_{o2} - out_{o2})^2$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = 2 * \frac{1}{2}(target_{o1} - out_{o1})^{2-1} * -1 + 0$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = -(target_{o1} - out_{o1}) = -(0.01 - 0.75136507) = 0.74136507$$

$$out_{o1} = \frac{1}{1+e^{-net_{o1}}}$$

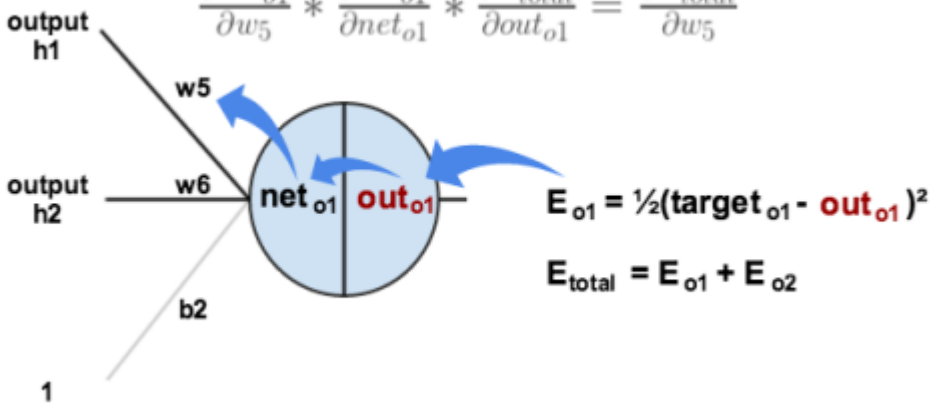
$$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1}) = 0.75136507(1 - 0.75136507) = 0.186815602$$

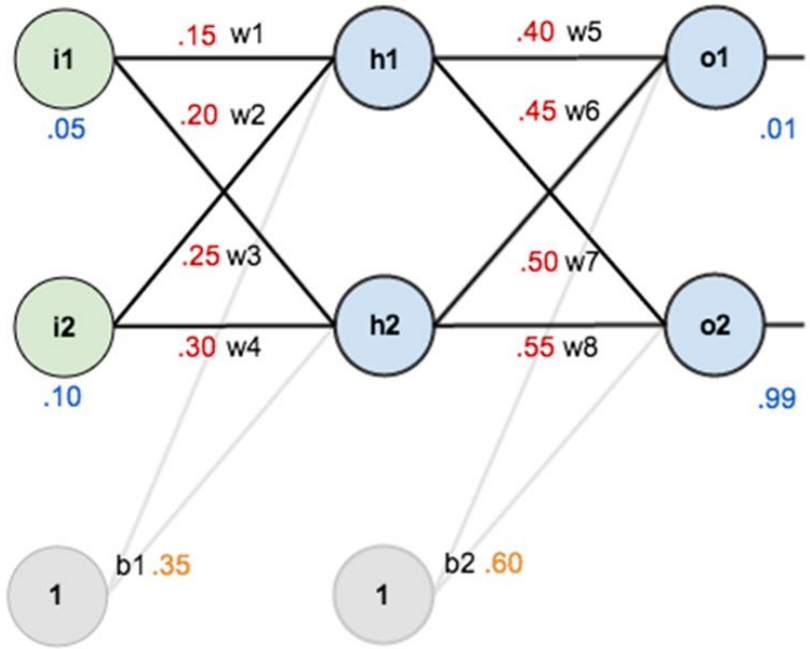
$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$\frac{\partial net_{o1}}{\partial w_5} = 1 * out_{h1} * w_5^{(1-1)} + 0 + 0 = out_{h1} = 0.593269992$$

$$\frac{\partial E_{total}}{\partial w_5} = 0.74136507 * 0.186815602 * 0.593269992 = 0.082167041$$

$$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$$



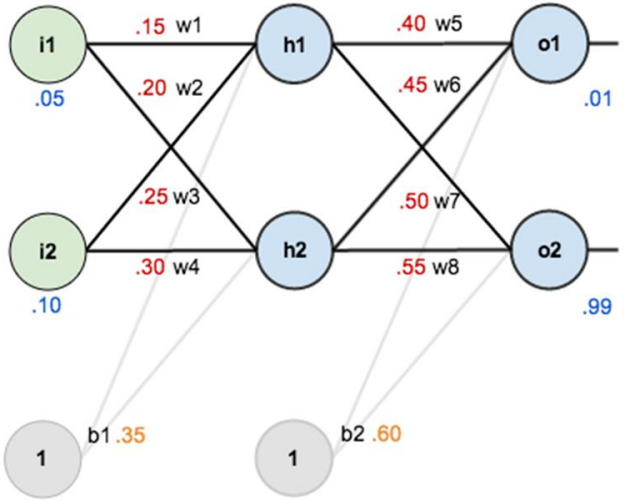


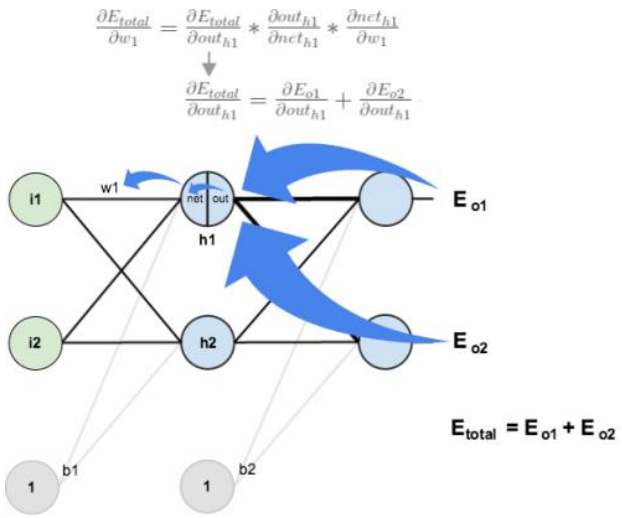
$$w_5^+ = w_5 - \eta * \frac{\partial E_{total}}{\partial w_5} = 0.4 - 0.5 * 0.082167041 = 0.35891648$$

$$w_6^+ = 0.408666186$$

$$w_7^+ = 0.511301270$$

$$w_8^+ = 0.561370121$$





$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$

$$\frac{\partial E_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial out_{h1}}$$

$$\frac{\partial E_{o1}}{\partial net_{o1}} = \frac{\partial E_{o1}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} = 0.74136507 * 0.186815602 = 0.138498562$$

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$\frac{\partial net_{o1}}{\partial out_{h1}} = w_5 = 0.40$$

$$\frac{\partial E_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial out_{h1}} = 0.138498562 * 0.40 = 0.055399425$$

$$\frac{\partial E_{o2}}{\partial out_{h1}} = -0.019049119$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}} = 0.055399425 + -0.019049119 = 0.036350306$$

$$out_{h1} = \frac{1}{1+e^{-net_{h1}}}$$

$$\frac{\partial out_{h1}}{\partial net_{h1}} = out_{h1}(1 - out_{h1}) = 0.59326999(1 - 0.59326999) = 0.241300709$$

$$net_{h1} = w_1 * i_1 + w_3 * i_2 + b_1 * 1$$

$$\frac{\partial net_{h1}}{\partial w_1} = i_1 = 0.05$$

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial w_1} = 0.036350306 * 0.241300709 * 0.05 = 0.000438568$$

$$w_1^+ = w_1 - \eta * \frac{\partial E_{total}}{\partial w_1} = 0.15 - 0.5 * 0.000438568 = 0.149780716$$

$$w_2^+ = 0.19956143$$

$$w_3^+ = 0.24975114$$

$$w_4^+ = 0.29950229$$