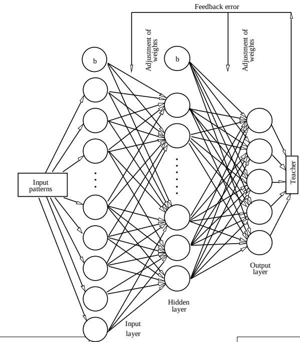
Neural Network: Example

Andrew Kusiak
Industrial Engineering
2139 Seamans Center
The University of Iowa
Iowa City, Iowa 52242 - 1527
Tel: 319 - 335 5934 Fax: 319-335 5669
andrew-kusiak@uiowa.edu
http://www.icaen.uiowa.edu/~ankusiak

BACK-PROPAGATION NEURAL NETWORK
Three-layer back-propagation neural network

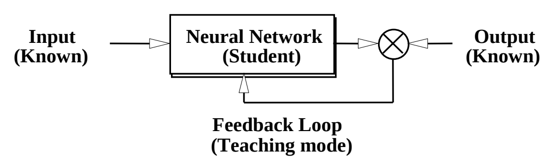


Supervised learning

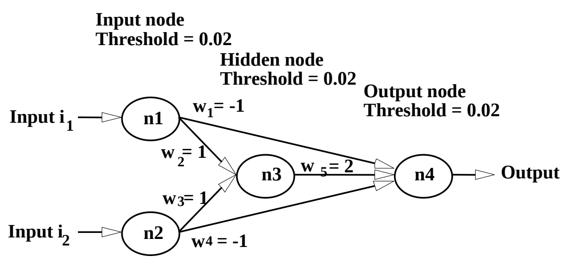
Example: Vacations Back-Propagation Learning

Inputs	Output
Cost	Customer satisfaction
Distance traveled	
Entertainment	

Back-Propagation Learning



Example: Trained NN



Consider the XOR Truth Table

Input 1	Input 2	Output
i_1	i_2	o_4
0	0	0
0	1	1
1	0	1
1	1	0

Example 1 $(i_1, i_2) = (0, 0)$

$o_4 = 0$

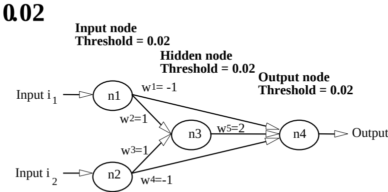
For $i_1 = i_2 = 0$

$o_1 = o_2 = 0$

$a_3 = w_2 \times o_1 + w_3 \times o_2$ $\quad a_3 = 1x0 + 1x0 = 0$

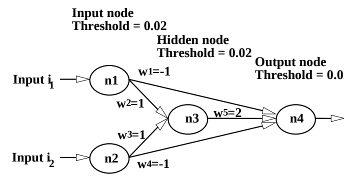
$$o_3 = \begin{cases} 0 & \text{IF } a_3 \le 0.02 \\ 1 & \text{IF } a_3 > 0.02 \end{cases}$$

$o_3 = 0$



$a_4 = w_1 \times o_1 + w_5 \times o_3 + w_4 \times o_2 = -1 \times 0 + 2 \times 0 + 1 \times 0$
$= 0$

$o_4 = 0$



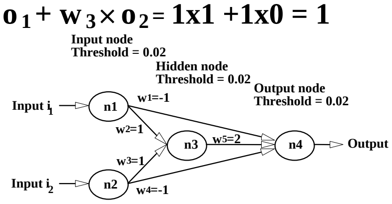
Example 2 (i1, i2) = (1, 0)

o4 = 1

o1 = 1, o2 = 0

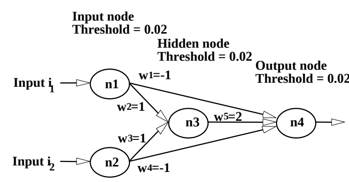
$a_3 = w_2 \times o_1 + w_3 \times o_2 = 1x1 + 1x0 = 1$

o3 = 1



$a_4 = w_1 \times o_1 + w_5 \times o_3 + w_4 \times o_2 = -1 \times 1 + 2 \times 1 + -1 \times 0$
$= 1$

$o_4 = 1$

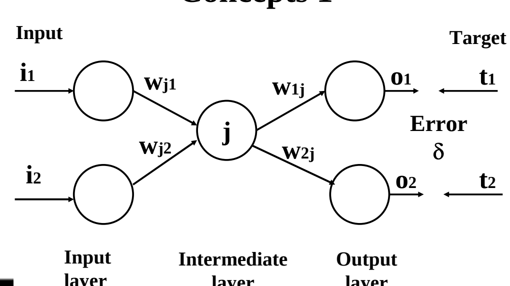


Fuzzy (Sigmoid) Activation Function

$$o_i = \frac{1}{1 + e^{-\alpha(\sum \text{Weight} \times \text{Input} - \theta)}}$$

$$\textit{where} \begin{cases} \alpha & \text{the degree of fuzziness (constant during training)} \\ \theta & \text{the threshold level (its value changes)} \end{cases}$$

Backpropagation Learning: Basic Concepts 1



Backpropagation Learning: Basic Concepts 2

$$\delta = 0.5 \sum_{k=1}^{n} (t_k - o_k)^2$$

δ_k = error occurring in the output layer k

The Back-Propagation Learning Algorithm

Step 1. Weight initialization

Set all weights and node thresholds to small random numbers.

Step 2. Calculation of output levels

(a) The output level of an input neuron is determined by the instance presented to the network.

(b) The output level o_j of each hidden and output neuron is determined

$$o_j = f(\sum w_{ji} o_i - \theta_j) = \frac{1}{1 + e^{-\alpha(\sum w_{ij} o_i - \theta_j)}} \qquad (1)$$

where w_{ij} is the weight from input o_i, α is a constant, θ_j is the node threshold, and f is a sigmoid function.

Step 3. Weight training

(a) The error gradient (gradient of the activation function x error of the neuron output) is completed as follows:

For the output neurons:

$$\delta_j = o_j(1 - o_j)(d_j - o_j) \qquad (2)$$

where d_j is the desired (target) output activation and o_j is the actual output activation at output neuron j.

For the hidden neurons:

$$\delta_j = o_j(1 - o_j)\sum_k \delta_k w_{kj} \qquad (3)$$

where δ_k is the error gradient at neuron k to which a connection points from hidden neuron j.

(b) The weight adjustment is computed as

$$\Delta w_{ji} = \eta \delta_j o_i \qquad (4)$$

where η is a trial-independent learning rate ($0<\eta<1$) and δ_j is the error gradient at neuron j.

(c) Start with the output neuron and work backward to the hidden layers recursively.

Adjust weights by

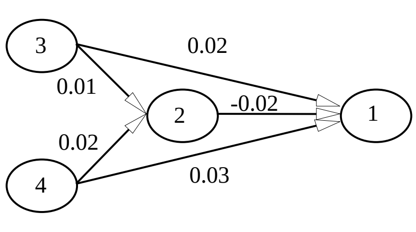
$$w_{ji}(t+1) = w_{ji}(t) + \Delta w_{ji} \qquad (5)$$

where $w_{ji}(t)$ is the weight from neuron i to neuron j at iteration t and Δw_{ji} is the weight adjustment.

(d) Perform the next iteration (repeat Steps 2 and 3) until the error criterion is met, i.e., the algorithm converges. An iteration includes: presenting an instance, calculating activation levels, and modifying weights.

Example

Back-Propagation Network for Learning the XOR Function with Randomly Generated Weights



Step 1. The weights are randomly initialized as follows: $w_{13} = 0.02$, $w_{14} = 0.03$, $w_{12} = 0.02$, $w_{23} = 0.01$, $w_{24} = 0.02$

Step 2. <u>Calculation of activation levels</u>: Consider a training instance (the fourth row from the XOR table) with the input vector = (1, 1) and the desired output = 0. From the figure,

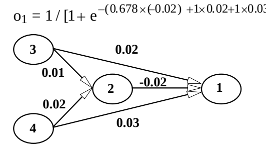
$$o_3 = i_3 = 1$$

$$o_4 = i_4 = 1$$

From equation (1) for $\alpha = 1$ and $\theta_j = 0$

$$o_2 = 1 / [1 + e^{-(1\times 0.01 + 1\times 0.02)}] = 0.678$$

$$o_1 = 1 / [1 + e^{-(0.678\times(-0.02) + 1\times 0.02 + 1\times 0.03)}] = 0.509$$



Step 3. Weight training: Assume the learning rate $\delta = 0.3$

Eq. 2 $\delta_j = o_j(1-o_j)(d_j - o_j)$ $\delta_1 = 0.678(1-0.678)(0-0.678) = -0.148$

Eq. 4 $\Delta w_{ji} = \eta\delta_j o_i$ $\Delta w_{13} = 0.3(-0148)\times 1 = -0.044$

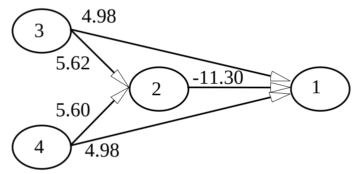
Eq. 5 $w_{ji}(t+1) = w_{ji}(t) + \Delta w_{ji}$ $w_{13} = 0.02-0.044 = -0.024$

Eq. 2 $\delta_j = o_j(1-o_j)\sum_k \delta_k w_{kj}$ $\delta_2 = 0.678(1-0.678)(-0.148)(-0.02) = 0.0006$

From $\Delta w_{ji} = \eta\delta_j o_i$ $\Delta w_{23} = 0.3\times 0.0006\times 1 = 0.00018$

From $w_{ji}(t+1) = w_{ji}(t) + \Delta w_{ji}$ $w_{23} = 0.01 + 0.00018 = 0.01018$

The Previous Network with New Weights



o3 = 1
o4 = 0

$o_2 = 1 / [1 + e^{-(1\times 5.62 + 0\times 5.62)}] = 0.9964$

$o_1 = 1 / [1 + e^{-(1\times 4.98) + 0\times 4.98 - 11.30\times 0.9964}] = 0.9999$