Neural Network: Examples

Andrew Kusiak
Intelligent Systems Laboratory
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 Learning

Example 1: Trained NN

Consider the XOR Truth Table

<table>
<thead>
<tr>
<th>Input 1</th>
<th>Input 2</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Test \((i_1, i_2) = (0, 0)\)  
\[ o_4 = 0 \quad o_1 = o_2 = 0 \]

\[ a_3 = w_2 \times o_1 + w_3 \times o_2 = 1 \times 0 + 1 \times 0 = 0 \]

\[ o_3 = \begin{cases} 0 & \text{IF } a_3 \leq 0.02 \\ 1 & \text{IF } a_3 > 0.02 \end{cases} \]

\[ o_3 = 0 \]

\[ a_4 = w_1 \times o_1 + w_3 \times o_3 + w_4 \times o_2 = -1 \times 0 + 2 \times 0 + 0 \times 0 = 0 \]

Test \((i_1, i_2) = (1, 0)\)  
\[ o_4 = 1 \]

\[ o_1 = o_2 = 0 \]

\[ a_3 = w_2 \times o_1 + w_3 \times o_2 = 1 \times 1 + 1 \times 0 = 1 \]

\[ o_3 = 0 \]

\[ a_4 = w_1 \times o_1 + w_3 \times o_3 + w_4 \times o_2 = -1 \times 1 + 2 \times 1 + -1 \times 0 = 1 \]
Fuzzy (Sigmoid) Activation Function

\[ o_j = \frac{1}{1 + e^{-\alpha(\text{Weight} \times \text{Input} - \theta)}} \]

where 
- \( \alpha \) the degree of fuzziness (constant during training)
- \( \theta \) the threshold level (its value changes)

Example 2
Backpropagation Learning: Basic Concepts

1

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

\( \delta_k \) = 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 a hidden and output neuron is determined

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

where \( w_{ij} \) is the weight from input \( o_i \), \( \alpha \) is a constant, \( \theta_j \) is the node threshold, and \( f \) is a sigmoid function.
Step 3. Weight training
(a) The error gradient is completed as follows:
For the output neurons:
\[ \delta_j = o_j(1 - o_j)(d_j - o_j) \]
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:
\[ \delta_j = o_j(1 - o_j) \sum_k \delta_k w_{kj} \]
where \[ \delta_k \] is the error gradient at neuron \( k \) to which a connection points from hidden neuron \( j \).
(b) The weight adjustment is computed as
\[ \Delta w_{ji} = \eta \delta_j o_i \]
where \( \eta \) is a trial-independent learning rate \( (0 < \eta < 1) \) and \( \delta_j \) is the error gradient at neuron \( j \).

(c) Start with the output neuron and work backward to the hidden layers recursively. Adjust weights by
\[ w_{ji}(t+1) = w_{ji}(t) + \Delta w_{ji} \]
where \( w_{ji}(t) \) is the weight from neuron \( i \) to neuron \( j \) at iteration \( t \) and \( \Delta w_{ji} \) 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

\[ \begin{align*}
3 & \rightarrow 0.01 \\
2 & \rightarrow -0.02 \\
4 & \rightarrow 0.02 \\
\end{align*} \]

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. Calculation of activation levels: 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,
\[ o_3 = 1 \]
\[ o_4 = 1 \]
From equation (1) for \( \alpha = 1 \) and \( \theta_j = 0 \)
\[ o_2 = 1 / [1 + e^{-(0.02)(1+1+0.02)}] = 0.678 \]
\[ o_1 = 1 / [1 + e^{-(0.02)(1+1+0.02)}] = 0.509 \]
Step 3. Weight training: Assume the learning rate $\delta = 0.3$

Eq. 2 $\delta_j = o_j (1 - o_j)(d_j - o_j)$

Eq. 4 $\Delta w_{ji} = \eta \delta_j o_i$

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

From $\Delta w_{ji} = \eta \delta_j o_i$

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

The Previous Network with New Weights

$$o_3 = \frac{1}{1 + e^{-1(5.62 + 0 + 5.62)}} = 0.9964$$
$$o_4 = \frac{1}{1 + e^{-1(4.98 + 0 + 4.98 + 11.30 + 0.9964)}} = 0.9999$$