Neural Networks

- Advantages
  - prediction accuracy is generally high
  - robust, works when training examples contain errors
  - fast evaluation of the learned target function

- Criticism
  - long training time
  - difficult to understand the learned function (weights)
  - not easy to incorporate domain knowledge
Neural Networks

- A neural network is a set of connected input/output units where each connection has a weight associated with it.
- It is also referred to as connectionist learning.
Multi-Layer Feed-Forward Network

Output vector

Output nodes

Hidden nodes

Input nodes

Input vector: \( x_i \)

\( W_{ij} \)
Defining a Network Topology

- Normalize the input values for each attribute
- Discrete-valued attributes may be encoded such that there is one input unit per domain value
- An output unit is used to represent two classes. If there are more than two classes, then one output unit per class is used.
A Neuron (I)

- The n-dimensional input vector is mapped into the output variable by means of the scalar product and a nonlinear function mapping

\[ \theta_j \text{ (bias)} \]

\[ I_j \]

\[ f \]

\[ \sum \]

\[ w_{0j} \]

\[ w_{1j} \]

\[ w_{nj} \]

Input

(weight vector \( w \))

(weighted sum)

Activation function

Output \( O' \)

Inputs (outputs from previous layer)
A Neuron (II)

\[ I_j = \sum_i w_{ij} O_i + \theta_j \]

Output \( O' \)

\[ O'_j = \frac{1}{1 + e^{-I_j}} \]

squashing function
(to map a large input domain onto \([0,1]\))
An example of a neural network

Assume all the weights and thresholds have been trained

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$W_{14}$</th>
<th>$W_{15}$</th>
<th>$W_{24}$</th>
<th>$W_{25}$</th>
<th>$W_{34}$</th>
<th>$W_{35}$</th>
<th>$W_{46}$</th>
<th>$W_{56}$</th>
<th>$\theta_4$</th>
<th>$\theta_5$</th>
<th>$\theta_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0.2</td>
<td>-0.3</td>
<td>0.4</td>
<td>0.1</td>
<td>-0.5</td>
<td>0.2</td>
<td>-0.3</td>
<td>-0.2</td>
<td>-0.4</td>
<td>0.2</td>
<td>0.1</td>
</tr>
</tbody>
</table>
Using the Neural Network for Prediction

Table 7.4 The net input and output calculations.

<table>
<thead>
<tr>
<th>Unit $j$</th>
<th>Net input, $I_j$</th>
<th>Output, $O_j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>$0.2 + 0 - 0.5 - 0.4 = -0.7$</td>
<td>$1/(1 + e^{0.7}) = 0.332$</td>
</tr>
<tr>
<td>5</td>
<td>$-0.3 + 0 + 0.2 + 0.2 = 0.1$</td>
<td>$1/(1 + e^{-0.1}) = 0.525$</td>
</tr>
<tr>
<td>6</td>
<td>$(-0.3)(0.332) - (0.2)(0.525) + 0.1 = -0.105$</td>
<td>$1/(1 + e^{0.105}) = 0.474$</td>
</tr>
</tbody>
</table>

Prediction output = 0
Network Training

- The ultimate objective of training
  - obtain a set of weights that makes almost all the tuples in the training data classified correctly

- Steps
  - Initialize weights with random values
  - While *terminating condition* not satisfied
    - Feed the *input tuples* into the network one by one
      - For each unit
        - Compute the net input to the unit as a linear combination of all the inputs to the unit
        - Compute the output value using the activation function
        - Compute the error
        - Update the weights and the bias
Network Training (Backpropagation) (I)

Output vector

Output nodes

Hidden nodes

Input nodes

Input vector: $x_i$

$$Err_j = O_j (1 - O_j) (T_j - O_j)$$

where $T_j$ is the true output

$$Err_j = O_j (1 - O_j) \sum_k Err_k w_{jk}$$

$w_{ij}$
Network Training (Backpropagation) (II)

Output vector

Output nodes

Hidden nodes

Input nodes

Input vector: \( x_i \)

\[ l \text{ is the learning rate} \]

\[ \theta_j = \theta_j + (l)Err_j \]

\[ w_{ij} = w_{ij} + (l)Err_jO_i \]

\[ w_{ij} \quad \theta_j = \theta_j + (l)Err_j \]

\[ w_{ij} = w_{ij} + (l)Err_jO_i \]
An example of training a neural network

Assume these are initial values for training

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$W_{14}$</th>
<th>$W_{15}$</th>
<th>$W_{24}$</th>
<th>$W_{25}$</th>
<th>$W_{34}$</th>
<th>$W_{35}$</th>
<th>$W_{46}$</th>
<th>$W_{56}$</th>
<th>$\theta_4$</th>
<th>$\theta_5$</th>
<th>$\theta_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0.2</td>
<td>-0.3</td>
<td>0.4</td>
<td>0.1</td>
<td>-0.5</td>
<td>0.2</td>
<td>-0.3</td>
<td>-0.2</td>
<td>-0.4</td>
<td>0.2</td>
<td>0.1</td>
</tr>
</tbody>
</table>
## Learning Example

### Table 7.4 The net input and output calculations.

<table>
<thead>
<tr>
<th>Unit $j$</th>
<th>Net input, $I_j$</th>
<th>Output, $O_j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>$0.2 + 0 - 0.5 - 0.4 = -0.7$</td>
<td>$1/(1 + e^{-0.7}) = 0.332$</td>
</tr>
<tr>
<td>5</td>
<td>$-0.3 + 0 + 0.2 + 0.2 = 0.1$</td>
<td>$1/(1 + e^{-0.1}) = 0.525$</td>
</tr>
<tr>
<td>6</td>
<td>$(-0.3)(0.332) - (0.2)(0.525) + 0.1 = -0.105$</td>
<td>$1/(1 + e^{0.105}) = 0.474$</td>
</tr>
</tbody>
</table>

### Table 7.5 Calculation of the error at each node.

<table>
<thead>
<tr>
<th>Unit $j$</th>
<th>$Err_j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>$(0.474)(1 - 0.474)(1 - 0.474) = 0.1311$</td>
</tr>
<tr>
<td>5</td>
<td>$(0.525)(1 - 0.525)(0.1311)(-0.2) = -0.0065$</td>
</tr>
<tr>
<td>4</td>
<td>$(0.332)(1 - 0.332)(0.1311)(-0.3) = -0.0087$</td>
</tr>
</tbody>
</table>
Learning rate = 0.9

<table>
<thead>
<tr>
<th>Weight or bias</th>
<th>New value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_{46}$</td>
<td>$-0.3 + (0.9)(0.1311)(0.332) = -0.261$</td>
</tr>
<tr>
<td>$w_{56}$</td>
<td>$-0.2 + (0.9)(0.1311)(0.525) = -0.138$</td>
</tr>
<tr>
<td>$w_{14}$</td>
<td>$0.2 + (0.9)(-0.0087)(1) = 0.192$</td>
</tr>
<tr>
<td>$w_{15}$</td>
<td>$-0.3 + (0.9)(-0.0065)(1) = -0.306$</td>
</tr>
<tr>
<td>$w_{24}$</td>
<td>$0.4 + (0.9)(-0.0087)(0) = 0.4$</td>
</tr>
<tr>
<td>$w_{25}$</td>
<td>$0.1 + (0.9)(-0.0065)(0) = 0.1$</td>
</tr>
<tr>
<td>$w_{34}$</td>
<td>$-0.5 + (0.9)(-0.0087)(1) = -0.508$</td>
</tr>
<tr>
<td>$w_{35}$</td>
<td>$0.2 + (0.9)(-0.0065)(1) = 0.194$</td>
</tr>
<tr>
<td>$\theta_6$</td>
<td>$0.1 + (0.9)(0.1311) = 0.218$</td>
</tr>
<tr>
<td>$\theta_5$</td>
<td>$0.2 + (0.9)(-0.0065) = 0.194$</td>
</tr>
<tr>
<td>$\theta_4$</td>
<td>$-0.4 + (0.9)(-0.0087) = -0.408$</td>
</tr>
</tbody>
</table>
Algorithm: Backpropagation. Neural network learning for classification, using the backpropagation algorithm.

Input: The training samples, samples; the learning rate, \( l \); a multilayer feed-forward network, network.

Output: A neural network trained to classify the samples.

Method:

1. Initialize all weights and biases in network;
2. while terminating condition is not satisfied {
   3. for each training sample \( X \) in samples {
      4. // Propagate the inputs forward:
      5. for each hidden or output layer unit \( j \) {
         6. \( I_j = \sum_i w_{ij}O_i + \theta_j \); // compute the net input of unit \( j \) with respect to the previous layer, \( i \)
         7. \( O_j = \frac{1}{1+e^{-I_j}} \); // compute the output of each unit \( j \)
      8. // Backpropagate the errors:
      9. for each unit \( j \) in the output layer
         10. \( Err_j = O_j(1 - O_j)(T_j - O_j) \); // compute the error
      11. for each unit \( j \) in the hidden layers, from the last to the first hidden layer
         12. \( Err_j = O_j(1 - O_j) \sum_k Err_k w_{jk} \); // compute the error with respect to the next higher layer, \( k \)
      13. for each weight \( w_{ij} \) in network {
         14. \( \Delta w_{ij} = (l)Err_j O_i \); // weight increment
         15. \( w_{ij} = w_{ij} + \Delta w_{ij} \); // weight update
      16. for each bias \( \theta_j \) in network {
         17. \( \Delta \theta_j = (l)Err_j \); // bias increment
         18. \( \theta_j = \theta_j + \Delta \theta_j \); // bias update
      }
   }
}

---

Figure 7.9 Backpropagation algorithm.
**Weight Updating**

- *Case updating* – The weights and biases are updated after the presentation of each sample.

- *Epoch updating* – The weight and bias increments could be accumulated in variables, so that the weights and biases are updated after all of the samples in the training set have been presented.

- In practice, case updating is more common.
Terminating Condition

- Training stops when
  - All changes in weights were so small as to be below some threshold, or
  - The percentage of samples misclassified is below some threshold, or
  - A pre-specified number of epochs has expired.

- In practice, several hundreds of thousands of epochs may be required before the weights will converge.