Synapse

- **Synapse** is the contact point between a dendrite and an axon.
- Human brain learns by changing the strength of the synaptic connection between neurons upon repeat stimulation by the same impulse.

Modeling a Neuron

- In a Biological Neural Network (BNN), learning results are saved at synapses.
- In an Artificial Neural Network (ANN), learning results are saved in the weights.

Example

<table>
<thead>
<tr>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>?</td>
</tr>
</tbody>
</table>
Perceptron

Input, Weights, and Bias
- Input \( \mathbf{X} = [x_1, x_2, x_3] \)
- One per attribute
- Weights \( \mathbf{W} = [w_1, w_2, w_3] \)
- One per input
- Typically initialized to random values in the range \([-1.0, 1.0]\) or \([-0.5, 0.5]\)
- Bias \( t \)
- Typically a value in the range of \([-1.0, 1.0]\)

Output
\[
y = f \left( x_1 w_1 + x_2 w_2 + \ldots + x_n w_n + t \right)
= f \left( x_1 w_1 + x_2 w_2 + \ldots + x_n w_n + x_0 w_0 \right)
= f(\mathbf{X} \cdot \mathbf{W})
= f(\mathbf{XW}^T)
\]
\( t \) can be written as \( x_n w_n \) where \( x_n = 1 \) and \( w_n = t \)

Common Activation Functions
- Step Function
  \[
y = \begin{cases} 1 & \mathbf{X} \cdot \mathbf{W} \geq 0 \\ 0 & \mathbf{X} \cdot \mathbf{W} < 0 \end{cases}
\]
- Sign Function
  \[
y = \begin{cases} 1 & \mathbf{X} \cdot \mathbf{W} \geq 0 \\ -1 & \mathbf{X} \cdot \mathbf{W} < 0 \end{cases}
\]
- Linear Function
  \[
y = \mathbf{X} \cdot \mathbf{W}
\]
- Sigmoid Function
  \[
y = \frac{1}{1 + e^{-\mathbf{X} \cdot \mathbf{W}}}
\]

Learning
- Initialize \( \mathbf{W} \) and \( t \) to random values
- For each training record \((\mathbf{x}, y')\)
  - Compute the predicted output \( y \)
  - Update each weight \( w_i \)
    \[
w_i = w_i + \lambda (y' - y)x_i
\]
    \( \lambda \) is the learning rate

About Learning Rate
- Between 0 and 1
- Control the speed of adjustment
- Dynamic learning rate, e.g. \( 1 / t \) where \( t \) is the number of iterations so far
Learning Example

\[ W = [0.3, 0.3, 0.3], \quad t = -0.5, \quad \lambda = 0.5 \]

\[ y = \begin{cases} 
1 & 0.3x_1 + 0.3x_2 + 0.3x_3 - 0.5 \geq 0 \\
0 & 0.3x_1 + 0.3x_2 + 0.3x_3 - 0.5 < 0 
\end{cases} \]

<table>
<thead>
<tr>
<th></th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>R2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>R3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>R4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>R5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Learning Example

After \([W, t] \)

\begin{align*}
R1 & \quad [0.3, 0.3, 0.3, -0.5] \\
R2 & \quad [-0.2, 0.8, 0.3, 0] \\
R3 & \quad [-0.2, 0.3, 0.8, 0] \\
R4 & \quad [-0.7, 0.3, 0.3, -0.5] \\
R5 & \quad [-0.7, 0.3, 0.3, -0.5] 
\end{align*}

Intuition Behind Perceptron Learning ...

\[ y = wx + t \]

Intuition Behind Perceptron Learning ...

\[ y = wx + t \]

Intuition Behind Perceptron Learning ...

\[ y = wx + t \]
Multilayer ANN

Terminology

- Nodes/units
- Layers and number of layers
- Feed-forward and recurrent
- Fully connected

Learning

- Each node in a hidden or output layer is a perceptron
- The output of a node is used as the input for the nodes in the next layer
- Can we use the same perceptron learning process for multilayer ANN learning??

Understand Backpropagation

[Diagram of backpropagation with error propagation through layers]

Delta Rule – Linear Activation Function

[Diagram of linear activation function with weight update formula]

Adjust Weights

- To minimize sum of squared error

\[ E(W) = \sum_{i=1}^{n} (y'_i - y_i)^2 \]

\[ = \sum_{i=1}^{n} (y'_i - f(X \cdot W))^2 \]

\[ \Delta w_j = \lambda y_i (y'_i - y_i) \]
Delta Rule – Nonlinear Activation Function ...

For output nodes:
\[ \Delta w_{ij} = \lambda y_j f'(X \cdot W)(y'_j - y_j) \]
\[ Err_j = f'(X \cdot W)(y'_j - y_j) \]

See Delta Rule at: http://www.learnartificialneuralnetworks.com/backpropagation.html

... Delta Rule – Nonlinear Activation Function

For hidden nodes:
\[ \Delta w_{ij} = \lambda y_j f'(X \cdot W) \sum_{k=1}^{n} w_{jk} Err_k \]
\[ Err_j = f'(X \cdot W) \sum_{k=1}^{n} w_{jk} Err_k \]

See Delta Rule at: http://www.learnartificialneuralnetworks.com/backpropagation.html

Sigmoid Activation Function

\[ f(x) = \frac{1}{1 + e^{-x}} \]
\[ f'(x) = \frac{e^{-x}}{(1 + e^{-x})} = (1 - f(x))f(x) \]
\[ f'(X \cdot W) = (1 - f(X \cdot W))f(X \cdot W) = (1 - y)y \]

Multilayer ANN Example

Initial Values

\[
\begin{array}{cccccc}
1 & 0 & 1 & \lambda = 0.9 \\
0 & 2 & 3 & 4 & 5 & 6 \\
1 & 0 & 1 & 0.2 & -0.3 & -0.5 & 0.2 \\
& 0.4 & -0.2 & & & & \\
& & & 0.2 & -0.3 & 0.1 & \\
\end{array}
\]

Initial Values

\[
\begin{array}{cccccccc}
1 & x_1 & x_2 & x_3 & t_4 & t_5 & t_6 \\
0 & w_{14} & w_{15} & w_{24} & w_{25} & w_{34} & w_{35} & w_{46} & w_{56}
\end{array}
\]

\[
\begin{array}{cccccccc}
1 & 0 & 1 & -0.4 & 0.2 & 0.1 & \\
0 & 2 & 3 & 4 & 5 & 6 & y
\end{array}
\]

\[
\begin{array}{cccccccc}
0.2 & -0.3 & 0.4 & 0.1 & -0.5 & 0.2 & -0.3 & -0.2
\end{array}
\]
Forward Computation

Activation function: \( f(x) = \frac{1}{1 + e^{-x}} \)

<table>
<thead>
<tr>
<th>Node</th>
<th>( x \cdot w )</th>
<th>( f(x \cdot w) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>-0.7</td>
<td>0.332</td>
</tr>
<tr>
<td>5</td>
<td>0.1</td>
<td>0.525</td>
</tr>
<tr>
<td>6</td>
<td>-0.105</td>
<td>0.474</td>
</tr>
</tbody>
</table>

Backward Propagation ...

Assume \( y' = 1 \)

<table>
<thead>
<tr>
<th>Node</th>
<th>( f'(x \cdot w) )</th>
<th>Err</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>0.249</td>
<td>0.131</td>
</tr>
<tr>
<td>5</td>
<td>0.249</td>
<td>-0.0065</td>
</tr>
<tr>
<td>4</td>
<td>0.222</td>
<td>-0.0087</td>
</tr>
</tbody>
</table>

... Backward Propagation

<table>
<thead>
<tr>
<th>( w_{46} )</th>
<th>( w_{16} )</th>
<th>( t_b )</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.261</td>
<td>-0.138</td>
<td>0.218</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( w_{15} )</th>
<th>( w_{25} )</th>
<th>( w_{35} )</th>
<th>( t_b )</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.306</td>
<td>0.1</td>
<td>0.194</td>
<td>0.194</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( w_{14} )</th>
<th>( w_{24} )</th>
<th>( w_{34} )</th>
<th>( t_b )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.192</td>
<td>0.4</td>
<td>-0.508</td>
<td>-0.408</td>
</tr>
</tbody>
</table>

Input and Output of ANN

- **Input**
  - One input node for each binary or numeric attribute
  - What about categorical attributes??

- **Output**
  - One output node for a 2-class problem
  - K output nodes for a k-class problem

About ANN

- Multilayer feed-forward networks can approximate any function
- Determining the network topology is an empirical process
- Good at handling redundant features, but sensitive to noise
- Weight adjustment may converge to local minimum
- Training can be time consuming

Readings

- Textbook Chapter 9.2