Homework 7: Learning in Neural Networks
Due: 9:00am, Friday 2nd November.

For the first two questions of the homework you will need to understand the learning algorithm using the delta rule (which we went over on Friday session that half of you missed). For the third question of the homework you will need to understand the Backpropagation Algorithm (which we will go over on Monday).

Important: When answering, please make sure to work cleanly and systematically and show every step of the learning process for the network, from start to finish.

Learning Algorithm:

1. Calculate the output of the neuron for all of the training examples with the current weights and biases.
2. Pick a training pattern that the network gets wrong, and change all of its incoming weights and bias using the delta rule (see below).
3. Go back to step 2.

The algorithm stops once the output of the network is correct for all possible training patterns.

Delta Rule:

\[ \Delta w_{ij} = n \ x^p_i \ (t^p - y^p_j) \]

where \( w_{ij} \) is the weight of the connection between the input \( x_i \) and the output neuron \( y_j \), \( n \) is a constant learning rate (for this exercises you can use either 0.5 or 1.0), \( x^p_i \) is the input \( i \) for pattern \( p \), \( t^p \) is the target output for pattern \( p \), and \( y^p_j \) is the output of the neuron \( j \) for pattern \( p \). Remember that this rule must be applied to the bias as well. Remember that the input to the bias is always 1.

Remember that \( \Delta \) means that the weight is going to change by that amount. For example, if the \( w \) is currently 0, and the learning rate is 1, and for the pattern of input is 1, the output of the neuron is 0, but the target output should be 1, then the weight is going to change, according to: \( w = w + n \cdot x \cdot (t - y) = 0 + 1 \cdot 1 \cdot (1 - 0) = 1 \). So the weight is going to change from 0 to 1.
Transfer Function

All neurons have the following step function as a transfer function:

\[ f(x) = \begin{cases} 
1 & \text{if } x > 0 \\
0 & \text{if } x \leq 0 
\end{cases} \]

Logic interpretation

In order to solve logic functions, we will assume 0 means \textit{False} and 1 means \textit{True}. 
1. Use the learning algorithm with the delta rule given above to train a single-layer feedforward network with two inputs and one output to solve the AND logic gate, using the transfer function \( f(x) \) and the logic interpretation given in the previous.

Assume the weights and bias of the network start off with the following values:

\[
\begin{align*}
    w_0 &= 0 \\
    w_1 &= 0 \\
    w_2 &= 0
\end{align*}
\]

Remember that \( w_0 \) is the threshold term. You can think of it as the weight from an additional input \( x_0 \), which is always 1.

Show clearly each of the steps of the training process, until you arrive at the weights change enough to produce the correct solution.

<table>
<thead>
<tr>
<th>Training Pattern</th>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>Target output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
2. Use the learning algorithm with the delta rule given above to train a single-layer feedforward network with two inputs and one output to solve the OR logic gate, using the transfer function \( f(x) \) and the logic interpretation given in the previously.

Assume the weights and bias of the network start off with the following values:
\[
\begin{align*}
    w_0 &= 0 \\
    w_1 &= 0 \\
    w_2 &= 0 \\
\end{align*}
\]

Show clearly each of the steps of the training process, until you arrive the weights change enough to produce the correct solution.

<table>
<thead>
<tr>
<th>Training Pattern</th>
<th>x1</th>
<th>x2</th>
<th>Target output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
3. Use the backpropagation algorithm to change the weights of the network below. Do only one run through the algorithm.

This includes four main steps:

a. Propagate the activity of the network forward. Use the sigmoid function.
   i. \( f(x) = \frac{1}{(1 + e^{-x})} \)
   ii. \( y_k = f\left(\sum_{j=1}^{N} w_{jk}y_j + \theta_k\right) \), where \( \theta \) is the threshold or bias, the \( w_{0k} \) weights.

b. Calculate the error of the output neuron.
   i. \( \Delta_k = (y_k)(1-y_k)(T-y_k) \), where \( T \) is the target output.

c. Propagate the error backwards to the hidden neurons.
   i. \( \Delta_j = (y_j)(1-y_j)(w_{jk}\Delta_k) \), where \( \Delta_k \) is the error of the output neuron.

d. Update the weights.
   i. \( w_{jk} \leftarrow w_{jk} + \eta \cdot y_j \cdot \Delta_k \), where \( \eta \) is the learning rate (use \( \eta = 1.0 \)).

When training a network to solve a task, you will start your weights at random. For the purposes of this assignment, you can start off with the following weights and biases:

\[
\begin{align*}
  w_{01} &= 0.0 \\
  w_{11} &= 0.1 \\
  w_{21} &= 0.5 \\
  w_{02} &= 0.0 \\
  w_{12} &= 0.3 \\
  w_{22} &= 0.2 \\
  w_{03} &= 0.0 \\
  w_{13} &= 0.2 \\
  w_{23} &= 0.1 
\end{align*}
\]

When training a network, you will usually have a whole set of training examples. For the purposes of this assignment, you will only run the algorithm once through the loop, with just one training example:

\[
\begin{align*}
  x_1 &= 0.1 \\
  x_2 &= 0.2 \\
  \text{Target output, } T &= 0.0 
\end{align*}
\]

Important: Show as clearly as possible each of the steps of the training process, including your equations and calculations. Show also the resulting weights after that first pass through the algorithm.