Aims

- Section aim:
  - To teach the fundamental theory and practical applications of artificial neural networks.

- Session Aim:
  - This will entail an introduction to the biological background of these methods and implementation issues relevant to the development of practical systems.

Biological neuron

- http://hepux.rl.ac.uk/~candreop/minos/NeuralNets/neuralNetIntro.html
McCulloch-Pitts model

Y = 1 if \( W_1 x_1 + W_2 x_2 + W_3 x_3 \geq T \)
Y = 0 if \( W_1 x_1 + W_2 x_2 + W_3 x_3 < T \)

Training
- Rosenblatt invented perceptron training in which a neuron is shown the function to be implemented and the weights are found.
- This could only be done to single neurons, so Minsky's criticism was valid.
- In the 80s, back-propagation was discovered which allowed networks to be trained.

Multi-layered perceptron
- Feedforward network
- Usually has 3 layers
- All neurons in one layer are connected to all neurons in the next layer
- Good for pattern recognition
Multi-layered perceptron

Neural network approach
- Set up examples of numerals
- Train a network
- Done, in a matter of seconds

Learning and generalising
- Neural networks can do this easily because they have the ability to learn and to generalise from examples
- Learning is achieved by adjusting the weights
- Generalisation is achieved because similar patterns will produce an output
Introduce the bias

Take the threshold over to the other side of the equation and replace it with a weight $W_0$ which equals $-T$, and include a constant input $X_0$ which equals 1.

$Y = 1$ if $W_1X_1 + W_2X_2 + W_3X_3 + W_0X_0 \geq 0$

$Y = 0$ if $W_1X_1 + W_2X_2 + W_3X_3 + W_0X_0 < 0$

Logic functions - OR

$Y = X_1 \text{ OR } X_2$

Logic functions - AND

$Y = X_1 \text{ AND } X_2$
Logic functions - NOT

\[ y = \text{NOT} \ x_1 \]

The weighted sum

- The weighted sum, \( \sum W_i X_i \) is called the “net” sum.
- \( \text{Net} = \sum W_i X_i \)
- \( y = 1 \) if \( \text{net} \geq 0 \)
- \( y = 0 \) if \( \text{net} < 0 \)

Hard-limiter

The threshold function is known as a hard-limiter.

When \( \text{net} \) is zero or positive, the output is 1, when \( \text{net} \) is negative the output is 0.
**Example**

Original image

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Weights

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 0
- +1

Net = 14

**Example with bias**

With a bias of -14, the weighted sum, net, is 0. Any pattern other than the original will produce a sum that is less than 0. If the bias is changed to -13, then patterns with 1 bit different from the original will give a sum that is 0 or more, so an output of 1.

![Image](3.3.3.3)

**Generalisation**

- The neuron can respond to the original image and to small variations
- The neuron is said to have generalised because it recognises patterns that it hasn’t seen before
The AND function shown earlier had weights of -2, 1 and 1. Substituting into the equation for net gives:

\[ \text{net} = W_0X_0 + W_1X_1 + W_2X_2 = -2X_0 + X_1 + X_2 \]

Also, since the bias, \( X_0 \), always equals 1, the equation becomes:

\[ \text{net} = -2 + X_1 + X_2 \]

The change in the output from 0 to 1 occurs when:

\[ \text{net} = -2 + X_1 + X_2 = 0 \]

This is the equation for a straight line.

\[ X_2 = -X_1 + 2 \]

Which has a slope of -1 and intercepts the \( X_2 \) axis at 2. This line is known as a decision surface.
Linear separability

When a neuron learns it is positioning a line so that all points on or above the line give an output of 1 and all points below the line give an output of 0.

When there are more than 2 inputs, the pattern space is multi-dimensional, and is divided by a multi-dimensional surface (or hyperplane) rather than a line.

Are all problems linearly separable?

- No
- For example, the XOR function is non-linearly separable
- Non-linearly separable functions cannot be implemented on a single neuron
**Exclusive-OR (XOR)**

<table>
<thead>
<tr>
<th>X1</th>
<th>X2</th>
<th>Y</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>

Learning

- A single neuron learns by adjusting the weights.
- The process is known as the delta rule.
- Weights are adjusted in order to minimise the error between the actual output of the neuron and the desired output.
- Training is supervised, which means that the desired output is known.

Delta rule

The equation for the delta rule is:

$$\Delta W_i = \eta x_i (d - y)$$

where d is the desired output and y is the actual output.

The Greek “eta”, $\eta$, is a constant called the learning coefficient and is usually less than 1.

$\Delta W_i$ means the change to the weight, $W_i$. 
**Example**

- Assume that the weights are initially random
- The desired function is the AND function
- The inputs are shown one pattern at a time and the weights adjusted

---

**The AND function**

<table>
<thead>
<tr>
<th>X0</th>
<th>X1</th>
<th>X2</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

---

**Example**

Start with random weights of 0.5, -1, 1.5

When shown the input pattern 1 0 0 the weighted sum is:

\[ \text{net} = 0.5 \times 1 + (-1) \times 0 + 1.5 \times 0 = 0.5 \]

This goes through the hard-limiter to give an output of 1.
The desired output is 0. So the changes to the weights are:

- W0 negative
- W1 zero
- W2 zero
**Example**

New value of weights (with $\eta$ equal to 0.1) of 0.4, -1, 1.5

When shown the input pattern 1 0 1 the weighted sum is:

$$\text{net} = 1 \times 0.4 + (-1) \times 0 + 1.5 \times 1 = 1.9$$

This goes through the hard-limiter to give an output of 1. The desired output is 0. So the changes to the weights are:

- $W_0$ negative
- $W_1$ zero
- $W_2$ negative

---

**Example**

New value of weights of 0.3, -1, 1.4

When shown the input pattern 1 1 0 the weighted sum is:

$$\text{net} = 1 \times 0.3 + (-1) \times 1 + 1.4 \times 0 = -0.7$$

This goes through the hard-limiter to give an output of 0. The desired output is 0. So the changes to the weights are:

- $W_0$ zero
- $W_1$ zero
- $W_2$ zero

---

**Example**

New value of weights of 0.3, -1, 1.4

When shown the input pattern 1 1 1 the weighted sum is:

$$\text{net} = 1 \times 0.3 + (-1) \times 1 + 1.4 \times 1 = 0.7$$

This goes through the hard-limiter to give an output of 1. The desired output is 1. So the changes to the weights are:

- $W_0$ zero
- $W_1$ zero
- $W_2$ zero
### Example - with \( \eta = 0.5 \)

<table>
<thead>
<tr>
<th>X0</th>
<th>X1</th>
<th>X2</th>
<th>W0</th>
<th>W1</th>
<th>W2</th>
<th>Net</th>
<th>Y</th>
<th>( \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>-1.0</td>
<td>1.5</td>
<td>0.5</td>
<td>1</td>
<td>-0.5</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0.0</td>
<td>-1.0</td>
<td>1.5</td>
<td>1.5</td>
<td>1</td>
<td>-0.5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-0.5</td>
<td>-1.0</td>
<td>1.0</td>
<td>-1.5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-0.5</td>
<td>-1.0</td>
<td>1.0</td>
<td>-0.5</td>
<td>0</td>
<td>0.5</td>
</tr>
</tbody>
</table>

### Example

<table>
<thead>
<tr>
<th>X0</th>
<th>X1</th>
<th>X2</th>
<th>W0</th>
<th>W1</th>
<th>W2</th>
<th>Net</th>
<th>Y</th>
<th>( \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.0</td>
<td>-0.5</td>
<td>1.5</td>
<td>0.0</td>
<td>1</td>
<td>-0.5</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-0.5</td>
<td>-0.5</td>
<td>1.5</td>
<td>1.0</td>
<td>1</td>
<td>-0.5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-1.0</td>
<td>-0.5</td>
<td>1.0</td>
<td>-1.5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1.0</td>
<td>-0.5</td>
<td>1.0</td>
<td>-0.5</td>
<td>0</td>
<td>0.5</td>
</tr>
</tbody>
</table>

### Example

<table>
<thead>
<tr>
<th>X0</th>
<th>X1</th>
<th>X2</th>
<th>W0</th>
<th>W1</th>
<th>W2</th>
<th>Net</th>
<th>Y</th>
<th>( \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-0.5</td>
<td>0.0</td>
<td>1.5</td>
<td>-0.5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-0.5</td>
<td>0.0</td>
<td>1.5</td>
<td>1.0</td>
<td>1</td>
<td>-0.5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-1.0</td>
<td>0.0</td>
<td>1.0</td>
<td>-1.0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
### Example

<table>
<thead>
<tr>
<th>X0</th>
<th>X1</th>
<th>X2</th>
<th>W0</th>
<th>W1</th>
<th>W2</th>
<th>Net</th>
<th>Y</th>
<th>0.5δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-1.0</td>
<td>0.0</td>
<td>1.0</td>
<td>-1.0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-1.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>1</td>
<td>-0.5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-1.5</td>
<td>0.0</td>
<td>0.5</td>
<td>-1.5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1.5</td>
<td>0.0</td>
<td>0.5</td>
<td>-1.0</td>
<td>0</td>
<td>0.5</td>
</tr>
</tbody>
</table>

### Example

<table>
<thead>
<tr>
<th>X0</th>
<th>X1</th>
<th>X2</th>
<th>W0</th>
<th>W1</th>
<th>W2</th>
<th>Net</th>
<th>Y</th>
<th>0.5δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-1.0</td>
<td>0.5</td>
<td>1.0</td>
<td>-1.0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-1.0</td>
<td>0.5</td>
<td>1.0</td>
<td>0.0</td>
<td>1</td>
<td>-0.5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-1.5</td>
<td>0.5</td>
<td>0.5</td>
<td>-1.0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1.5</td>
<td>0.5</td>
<td>0.5</td>
<td>-0.5</td>
<td>0</td>
<td>0.5</td>
</tr>
</tbody>
</table>

### Example

<table>
<thead>
<tr>
<th>X0</th>
<th>X1</th>
<th>X2</th>
<th>W0</th>
<th>W1</th>
<th>W2</th>
<th>Net</th>
<th>Y</th>
<th>0.5δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>-1.0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>0.0</td>
<td>1</td>
<td>-0.5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-1.5</td>
<td>1.0</td>
<td>0.5</td>
<td>-0.5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1.5</td>
<td>1.0</td>
<td>0.5</td>
<td>0.0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Example

<table>
<thead>
<tr>
<th>X0</th>
<th>X1</th>
<th>X2</th>
<th>W0</th>
<th>W1</th>
<th>W2</th>
<th>Net</th>
<th>Y</th>
<th>0.5δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-1.5</td>
<td>1.0</td>
<td>0.5</td>
<td>-1.5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-1.5</td>
<td>1.0</td>
<td>0.5</td>
<td>-1.0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-1.5</td>
<td>1.0</td>
<td>0.5</td>
<td>-0.5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1.5</td>
<td>1.0</td>
<td>0.5</td>
<td>0.0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

What happened in pattern space

What happened in pattern space
Conclusions

- A single neuron can be trained to implement any linearly separable function
- Training is achieved using the delta rule which adjusts the weights to reduce the error
- Training stops when there is no error
- Training is supervised
Conclusions

- To understand what a neuron is doing, it helps to picture what’s going on in pattern space.
- A linearly separable function can divide the pattern space into two areas using a hyperplane.
- If a function is not linearly separable, networks of neurons are needed.