Artificial Neural Networks

Yuki Osada
Andrew Cannon
Humans are an intelligent species.  
  ◦ One feature is the ability to learn.

The ability to learn comes down to the brain.  

The brain learns from experience.  
  ◦ Research shows that the brain stores information as patterns.  
  ◦ This information is stored in neurons.
Neurons do not regenerate suggesting that these cells are what provide us with the abilities to:
- remember,
- think, and
- apply previous experiences.

Humans generally have between 80 and 120 million neurons.
- Each neuron typically connect with 1000 to 10000 other neurons.
- The human brain is a huge network of neurons – a neural network.
The power of the human mind comes from the sheer number of these neurons, and its connections.

- The individual neurons act as a function of their incoming signals.
- Although neurons themselves are complicated, they don't exhibit complex behaviour on their own.
  - This is the key feature that makes it a viable computational intelligence approach.
Artificial Neural Networks are a computational model inspired by the neural structure of the human brain, a biological neural network.

- They attempt to replicate only the basic elements of this complicated, versatile, and powerful organism.
- It consists of an interconnected group of artificial neurons.
- It learns by changing its structure based on information that flows through the network.

They are used to model complex relationships between inputs and outputs, or to find patterns in data.
Artificial Neurons

- Neurons are the fundamental processing elements of a neural network.

A biological neuron basically:
1. receives inputs from other sources (dendrites),
2. merges them in some way (soma),
3. performs an operation on the result (axon), then
4. outputs the result – possibly to other neurons (axon terminals).

Artificial neurons follow this basic approach.
Artificial Neurons

- The basic structure of an artificial neuron consists of:
  1. input connections (dendrites) with weights,
  2. a summation function or input function (soma),
  3. a transfer function or activation function (axon), and
  4. output connections (axon terminals).

- It has no learning process as such.
Artificial Neurons

- The function:
  1. Input values enter the neuron via the connections.
  2. The inputs are multiplied with the weighting factor of their respective connection.
     - There is often a separate bias connection, which can act as a threshold for the neuron to produce some useful output.
  3. The modified inputs are fed into a summation function.
     - Usually just sums the products.
  4. The result from the summation function is sent to a transfer function.
     - Usually a step function, or a sigmoid function.
  5. The neuron outputs the result of the transfer function into other neurons, or to an outside connection.
Neural Networks

- How can the neurons be clustered together?
  - The structure used in these networks is a layering approach.
  - These layers are connected to each other in a linear fashion.
    - It's possible that a neuron may have an output connection to itself.
    - How these layers may be connected is generally problem-dependent.
Neural Networks

- Single layer neurons are the simplest networks.
  - Multiple input sources will be fed into the set of neurons, which produce the outputs to the neural network.
  - These are called perceptrons.
  - These perceptrons can only represent linearly-separable functions.
- We can make the system represent more complex functions by adding more layers.
Multi-layered neural networks are more powerful than single-layered neural networks.

- The cost is that these hidden layers increase the complexity and training time of these networks.

Networks with a single hidden layer can approximate any continuous function with arbitrary accuracy.

Networks with two hidden layers can represent discontinuous functions.
Neural Networks

Neural Networks

- There are two main types of multi-layered neural networks:
  1. Feedforward.
     - A simple acyclic structure:
       - Information always moves in one direction; it never goes backwards.
       - Stateless encoding; no information is accumulated.
There are two main types of multi-layered neural networks:

2. Recurrent.
   ◦ A structure with cyclic feedback loops:
     • Information may be sent to any layer; it can process arbitrary sequences of input, and produce more complex results.
     • Stateful encoding; introduces short-term memory into the system, and allows dynamic temporal behaviour.
Training

- Artificial neural networks are used to model complex systems that were not understood by the programmer.
  - We usually don't know how to construct a perfect neural network for a problem.
  - We must train them to produce better results.
  - We can only train aspects of a neural network.
Training

- Training is the adjusting of parameters with the aim to minimise a measure of error, the cost function.
- What parameters in the artificial neural network do we want to adjust?
  - The weighting factors.
    - The link weights influence the function represented by the neural network.
    - In the case when we have no idea for the link weights, these might be randomly generated at initialisation.
Training

- There are 2 main approaches to training:
  - Supervised:
    - The user provides sample input and output data. The network adjusts its weights to match the expected results.
  - Unsupervised:
    - Only input data is supplied, and the neural network must find patterns on its own.
Sources

Applications

- Language processing
- Character recognition
- Pattern recognition
- Signal processing
- Prediction
Learning

- Supervised learning
  - Perceptron
  - Feedforward, back-propagation

- Unsupervised learning
  - Self organizing maps
Perceptron

- Simplest type of neural network
- Introduced by Rosenblatt (1958)

Adapted from Haykin, SS 2009 (p48)
Perceptron

- Simplest type of neural network
- Introduced by Rosenblatt (1958)

Adapted from Haykin, SS 2009 (p48)
Perceptron

- Simplest type of neural network
- Introduced by Rosenblatt (1958)

Adapted from Haykin, SS 2009 (p48)
Perceptron

- Input is a real vector $i = (i_1, i_2, \ldots, i_n)$

- Calculate a weighted scalar $s$ from inputs
  \[ s = \sum_j w_j i_j + b \]

- Calculate output
  \[ r = \text{sgn}(s) \]
Perceptron

- Categorises input vectors as being in one of two categories
- A single perceptron can be trained to separate inputs into two linearly separable categories
Training a Perceptron

- Need a training set of input/output pairs
- Initialise weights and bias (randomly or to zero)
- Calculate output
- Adjust the weights and bias in proportion to the difference between actual and expected values
Training a Perceptron

- Repeat until termination criteria is reached
- Rosenblatt (1962) showed that the weights and bias will converge to fixed values after a finite number of iterations (if the categories are linearly separable)
We want to classify points in $\mathbb{R}^2$ into those points for which $y \geq x + 1$ and those for which $y < x + 1$.
Perceptron Example

- Initialise bias/weight vector to (0,0,0)
- Input is the point (-1,-1) (below the line) – expressed as (1,-1,-1)

- $s = 0 \times 1 + 0 \times -1 + 0 \times -1 = 0$

- Actual output is $\text{sgn}(0) = +1$
- Expected output is -1 (below the line)
Perceptron Example

- Error (expected–actual) is \(-2\)
- Constant learning rate of 0.25

- So new weight vector is

\((0,0,0)\)
Perceptron Example

- Error (expected–actual) is −2
- Constant learning rate of 0.25

- So new weight vector is

\[(0,0,0) + 0.25\]
Perceptron Example

- Error (expected–actual) is $-2$
- Constant learning rate of 0.25

- So new weight vector is

$$(0,0,0) + 0.25(-2)$$
Perceptron Example

- Error (expected–actual) is $-2$
- Constant learning rate of 0.25

- So new weight vector is

$$(0,0,0) + 0.25(-2)(1,-1,-1)$$
Perceptron Example

- Error (expected–actual) is $-2$
- Constant learning rate of 0.25

- So new weight vector is

$$(0,0,0) + 0.25(-2)(1,-1,-1) = (-0.5,0.5,0.5)$$
Perceptron Example

- New bias/weight vector is \((-0.5, 0.5, 0.5)\)
- Input is the point \((0, 2)\) (above the line) – expressed as \((1, 0, 2)\)

\[ s = -0.5 \times 1 + 0.5 \times 0 + 0.5 \times 2 = 0.5 \]

- Actual output is \(\text{sgn}(0.5) = +1\)
- Expected output is \(+1\) (above the line) – no change to weight
Perceptron Example

- Eventually, this will converge to the correct answer of \((-a,-a,a)\) for some \(a>0\)

- Generally, we won’t know the correct answer!
Feedforward, Back-Propagation

- Feedforward network has no connections looping backwards
- Back-propagation algorithm allows for learning
Back-Propagation

- Operates similarly to perceptron learning
Back-Propagation

- Inputs are fed forward through the network
Back-Propagation

- Inputs are fed forward through the network
Back-Propagation

- Inputs are fed forward through the network
Back-Propagation

- Inputs are fed forward through the network
Back-Propagation

- Inputs are fed forward through the network

Input

Output

Compare to expected
Back-Propagation

- Errors are propagated back
Back-Propagation

- Errors are propagated back
Back-Propagation

- Errors are propagated back
Back-Propagation

- Errors are propagated back
Back-Propagation

- Adjust weights based on errors

Input

Output
Back-Propagation

- Weights might be updated after each pass or after multiple passes
Supervised Learning Issues

- Need a comprehensive training set
- Network cannot be too large for the training set
- No guarantees the network will learn
- Network design and learning strategies impact the speed and effectiveness of learning
Unsupervised Learning

- More powerful (if you can make it work)
- No external notion of correct/incorrect output – the network uses internal rules to adjust its output in response to inputs
Self Organising Maps

- One or more inputs connected to a set of outputs
Self Organising Maps

- Output neurons form a lattice in (usually) two dimensional space
Self Organising Maps

- Measurable distance between output neurons
Self Organising Maps

- Based on the network weights, for each input, each output neuron is excited to a different degree
Self Organising Maps

- Select best matching unit (BMU)
Self Organising Maps

- Identify a neighbourhood around BMU
Self Organising Maps

- Based on their levels of excitation, adjust the weights of each output neuron in this neighbourhood to more closely match the input
- Hope that output neurons diverge into stable (and distinct) categories allowing the input data to be classified
Self Organising Maps

Adapted from:
Sources

Questions?