ECM3412/ECMM409
Nature Inspired Computation
Lecture 15

Neural Networks 3: Applying Neural Networks

Today

- Briefly revisit learning
- Look at properties of neural networks
- Data representation issues
- Overfitting
- Applications – AI and Classification

Erratum – Sigmoid Function

$$P(t) = \frac{1}{1 + e^{-z}}$$

$$P(z) = \frac{1}{1 + e^{-z}}$$

- The steepness of the curve is changed by z

Supervised Learning

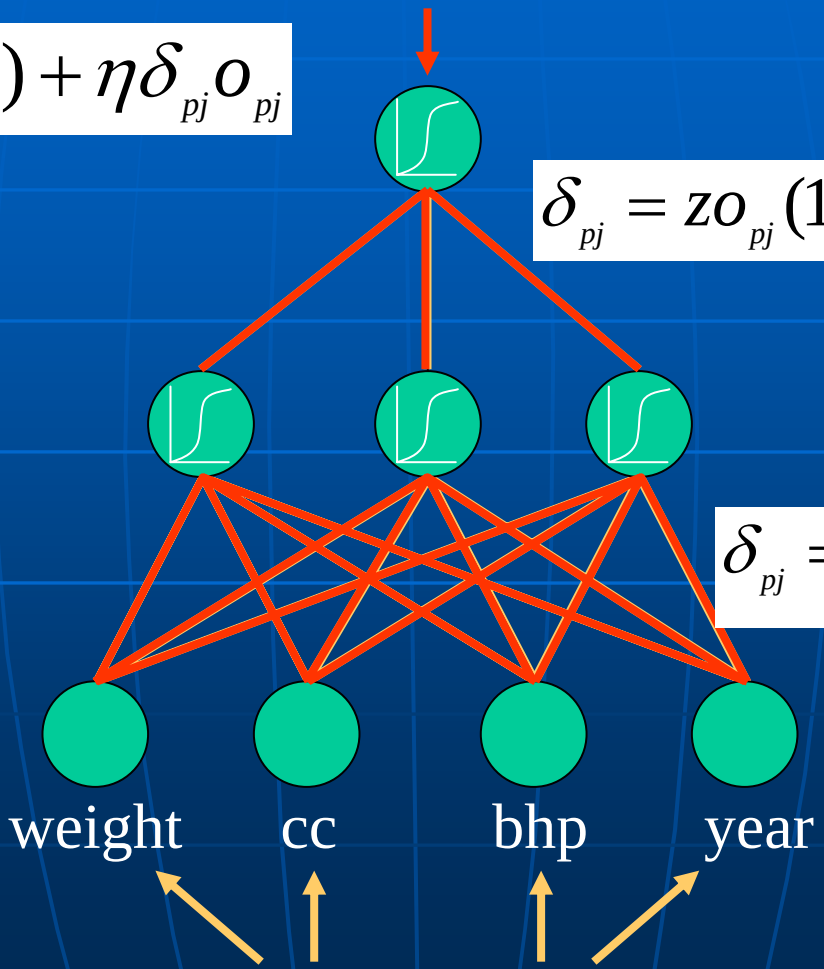
Output = 23.45 mpg Required = 18.00 mpg
Error = -5.45

$$w_{ij}(t+1) = w_{ij}(t) + \eta \delta_{pj} o_{pj}$$

$$\delta_{pj} = zo_{pj}(1-o_{pj})(t_{pj}-o_{pj})$$

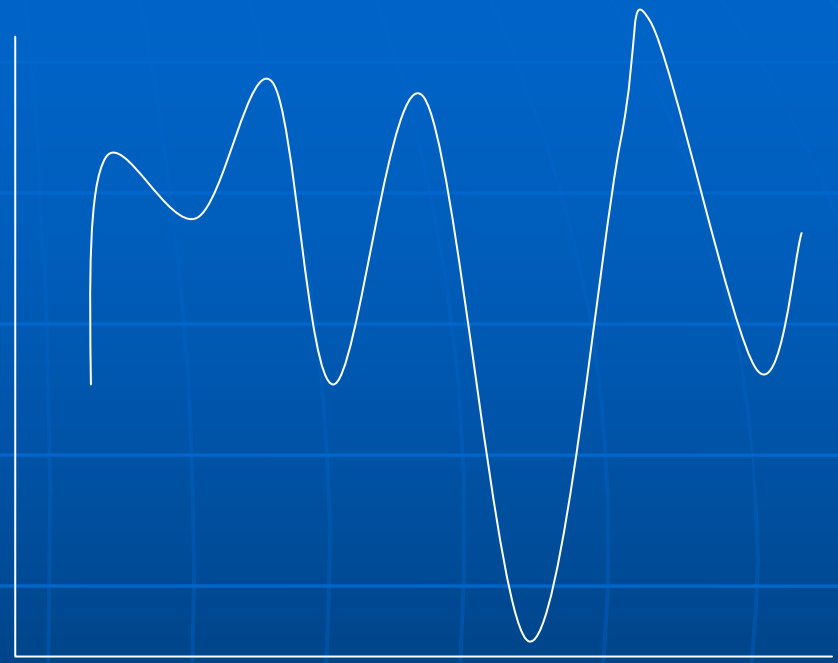
$$\delta_{pj} = zo_{pj}(1-o_{pj})\sum_{k}\delta_{pk} w_{jk}$$

Change Weights



Momentum

- The final addition we can make to the weight update term is momentum.
- Momentum encourages the network to make large changes to weights if the weight changes are currently large
- This allows the network to avoid local minima in the early stages as it can overcome hills



$$weightupdatefunction + \alpha(w_{ji}(t) - w_{ji}(t-1))$$

Two Types of Weight Updating

- Batch Updating
  - All patterns are presented, errors are calculated, then the weights are updated
- Online Updating
  - The weights are updated after the presentation of each pattern

Neural Network Properties

- Able to relate input variables to required output e.g.
  - Input car attributes and predict MPG
  - Predict stock market based on historical information
  - Classify individuals as ‘cancerous’ and ‘non-cancerous’ based on their genes
  - Many other control and learning tasks
- Is able to generalise between samples
- Shows ‘graceful degradation’ –

Graceful Degradation

- In symbolic systems, the removal of one component of the system usually results in failure
- Removal of neuron(s) from a neural network will
  - Reduce performance
  - Probably not result in overall failure
- Replicates our understanding of fault tolerance in the brain

‘Generalise’

- Symbolic systems tend to require explicit knowledge and workarounds when this is not available – ELIZA & CYC
- Can operate as expert systems in constrained environments
- Will quickly fail if taken out of environment
- General purpose AI machines such as CYC incredibly difficult to build – the makers of CYC have been trying since 1984.

Generalisation in Neural Networks

- Neural networks can learn common patterns in data
  - Therefore they can learn the distinctions between different classes of output
  - What makes an A? What makes a B?
  - Also allows them to be noise tolerant
  - How would you go about programming a computer to recognise these characters?

A B

Generalisation in Neural Networks

- Generalisation is useful because we do it.
  - We're able to infer properties of new objects/events
  - We're able to recognise most objects despite their orientation:

Which of these is a cup of coffee?

- Generalisation is also useful in classification problems
  - NNs can group samples together into one class which differentiates them from another
  - E.g. rugby players vs ballerinas

Neural Networks for Classification

- Classification is designed to group samples according to some known property:
  - Minimum 2 datasets required – training and testing
  - Training data
    - Consists of a set of measurements and a class
    - Is used for learning
  - Testing data
    - 'Unseen' examples
    - Is used only to test – no learning is undertaken

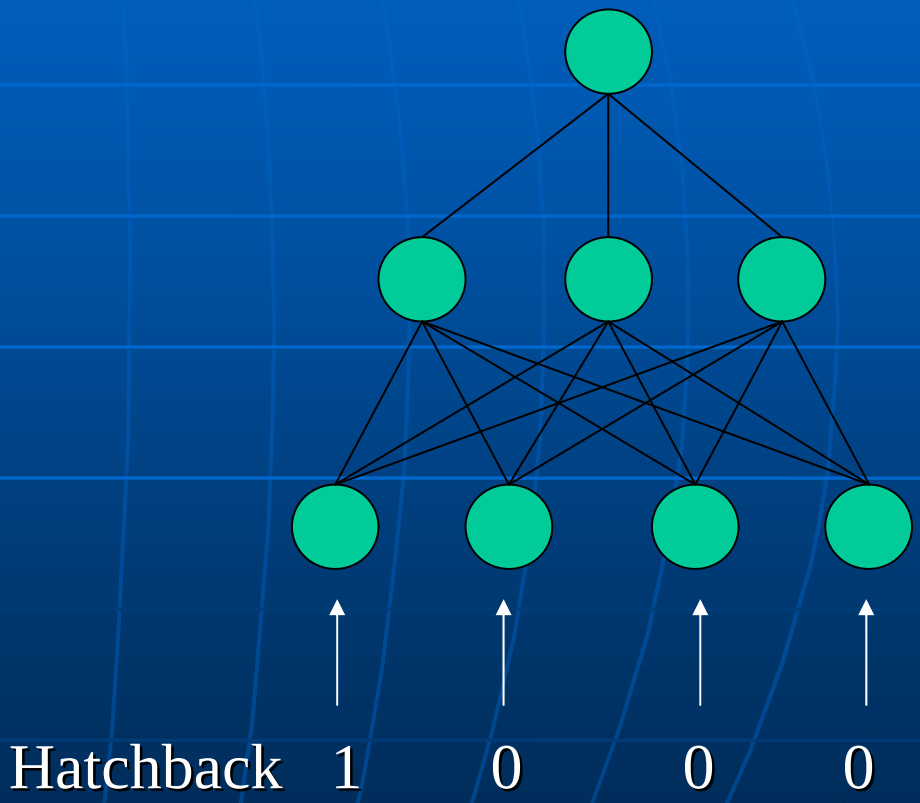
Name	*Hair colour*	*Height*	*Weight*	*Lotion used*	*Result*
Sarah	Blond	average	light	no	sunburned
Dana	Blond	tall	average	yes	not sunburned
Alex	Brown	short	average	yes	not sunburned
Annie	Blond	short	average	no	sunburned
Emily	Red	average	heavy	no	sunburned
Pete	Brown	tall	heavy	no	not sunburned
John	Brown	average	average	no	not sunburned
Katie	Blond	short	light	yes	not sunburned

Data Representation Issues

- Continuous data (e.g. floating point values)
  - Good data type for neural networks
  - Can require normalisation depending on the activation function – various ways to do this
- Integer-type Data
  - Can be entered into a single unit if scalar, but otherwise is likely to be better using approach below.
- Discrete Categories (e.g. car type – hatchback, saloon, estate etc..)
  - Needs to have a non-biased separate representation to the network
  - Two representations
    - Field-type
    - Thermometer-type

Discrete Representation Types

- Represent each category as a single unit:
- Field
  - Hatchback 1,0,0,0
  - Saloon 0,1,0,0
  - Estate 0,0,1,0
  - Coupe 0,0,0,1
- Thermometer
  - Hatchback 1,0,0,0
  - Saloon 1,1,0,0
  - Estate 1,1,1,0
  - Coupe 1,1,1,1



More Representation

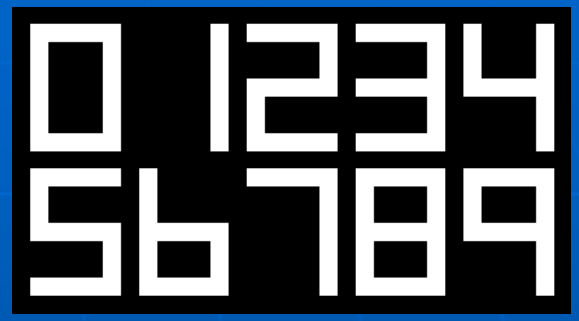
- Missing Values
  - Occur frequently in real world data
  - Cannot be entered directly into the network
  - Requires some value in each row
  - Normally an estimate of the value can be computed – number of ways to do this.

- Outputs from the network also need to be carefully considered (e.g.)
  - Continuous – MPG value
  - Discrete
    - Sunburnt 0,1
    - Non-Sunburnt 1,0
  - Image – collection of continuous values mapped to RGB

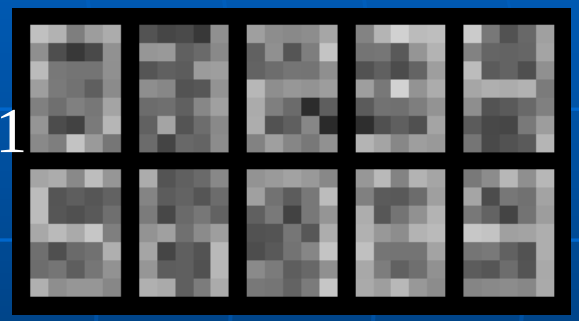
Overfitting/Overtraining

- Overfitting occurs when we train the network on a task for too long
- The network learns the noise in the input as well as the common patterns
- The result is poor performance on unseen examples
- In the example, if we train for long enough the network will learn the noise as well as the data
- We can overcome this by stopping the network learning earlier

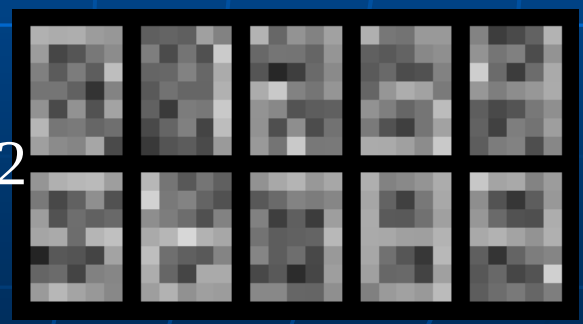
Data



Training 1



Training 2



Early Stopping

- A number of methods exist for early stopping of neural networks
- One of the best is cross-validation
  - Have three sets, training, testing and cross-validation
- Every epoch (iteration)
  - Train the network
  - ‘Clamp’ the weights
  - Test the network on cross-validation set
  - When the cross-validation set error starts to worsen – STOP!

Further Data Discussion

- Good commercial (and perhaps open source) neural network packages will do some of this work for you – not all of them though

- If your neural network is not learning, investigate the data representation first

- Also check – is your data representative of the problem you are trying to solve? A network trained on car data will probably not do well if presented with a motorbike

Key Applications – NetTalk (Sejnowski and Rosenberg, 1987)

- Neural network able to pronounce English text
- 203 input, 80 hidden and 26 output units
- Input
  - window of seven letters over the text
  - each letter represented by a field representation
  - 29 units for each letter (26 letters, spaces and other characters)
  - 29*7 = 203 units
- Output – one of 26 phonemes (sounds used in speech)

NetTalk

- During training the network produces sounds similar to an infant learning to talk
- Initially babbles and then becomes more coherent
- Eventually the system is able to pronounce 80-87% of new text correctly

Key Applications 2 – Classification Problems

- Many papers associated with classification tasks
- MLPs normally perform well
- BUT! – Extracting information as to how it has worked can be difficult
  - ANNs belong to the family of so called ‘black-box’ techniques

More Information

- Examinable reading – paper on ANN applications will be added
- Large number of free neural network simulators on the internet (e.g. SNNS) try them out
- Datasets can be downloaded from the machine learning repository:
  - http://mlearn.ics.uci.edu/databases/

Looking Ahead

- Tomorrow
  - Different architectures
  - Unsupervised learning