

COMP30030: Introduction to Artificial Intelligence

Neil Hurley

School of Computer Science
University College Dublin
`neil.hurley@ucd.ie`

November 5, 2018

1 Problem Solving by Search

- Uninformed Search
- Informed Search
- Adversarial Search
- Game Playing with Reinforcement Learning

2 Optimisation

- Optimisation Overview
- Combinatorial Optimisation Problems
- Simulated Annealing
- Optimisation Problem Examples
- Convergence of Simulated Annealing
- Genetic Algorithms
- Optimisation in Continuous Spaces

3 Machine Learning

- Supervised Machine Learning
- Feature Selection
- Decision Trees
- Neural Networks

Feature Selection

- Often, in real problem settings, there are many data attributes.
- Using spurious features will, at best, be wasteful and, at worst, may be detrimental to the performance of the classifier, if the feature contains noisy patterns.
- So, a key part of **practical Machine Learning** is the selection of an appropriate set of features to describe the problem instances.

Filter Methods for Feature Selection I

- One way to carry out feature selection is to apply a statistical measure to assign a score to each feature and select only the top scoring features.
- For example, in the case of numerical features, we can calculate the **correlation coefficient** between each feature and the output variable.

$$c_j = \frac{\sum_{i=1}^n (x_{ij} - \bar{x}_j)(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_{ij} - \bar{x}_j)^2 \sum_{i=1}^n (y_i - \bar{y})^2}}$$

- c_j is the Pearson correlation between the j^{th} feature and the output variable y , $-1 \leq c_j \leq 1$. Features for which $|c_j|$ is high should be good predictors of y .
- When c_j is positive, we expect that class $y_i = +1$ is likely whenever x_{ij} is high. When c_j is negative, we expect that class $y_i = -1$ is likely whenever x_{ij} is high.

Filter Methods for Feature Selection II

- Note that, above, the sum ($\sum_{i=1}^n$) is over all instances in the training set in order to get a score for each feature, $j = 1, \dots, m$.
- In general, we are trying to determine how **dependent** the output variable is on each feature variable.

1 Problem Solving by Search

- Uninformed Search
- Informed Search
- Adversarial Search
- Game Playing with Reinforcement Learning

2 Optimisation

- Optimisation Overview
- Combinatorial Optimisation Problems
- Simulated Annealing
- Optimisation Problem Examples
- Convergence of Simulated Annealing
- Genetic Algorithms
- Optimisation in Continuous Spaces

3 Machine Learning

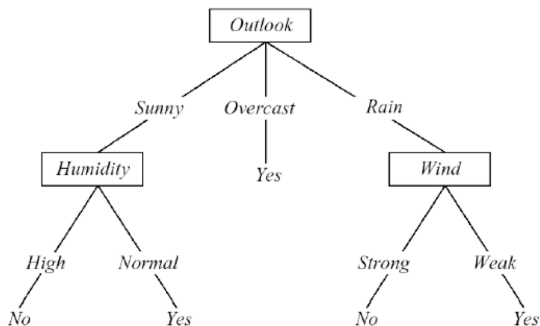
- Supervised Machine Learning
- Feature Selection
- Decision Trees
- Neural Networks

Decision Trees

- So far, we have considered the feature vector to be a set of numerical features.
- In many cases, however, the data is **categorical**, rather than numerical.
 - To describe the weather, I might use a set of categories: {sunny, rainy, windy, fine}.
- It's also possible to take a numerical feature, and e.g by thresholding, to describe it in categorical terms:
 - If height $> 180cm$ then tall
 - If height $> 190cm$ then very_tall
 - If $170 < \text{height} < 180$ then average. etc,
- **Decision Trees** are an intuitive way to carry out Machine Learning that is appropriate for categorical data.

What is a Decision Tree?

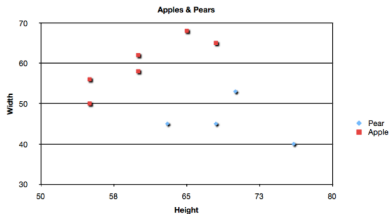
- A Tree structure where each node is labelled with a test or a question, each branch with possible answers, and leaf nodes with some decision or solution. By traversing the tree (answering questions and following appropriate branches), leaf nodes are reached and a decision can be made.



Decision Tree Representation

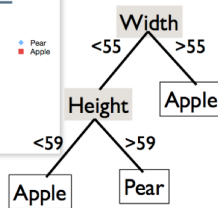
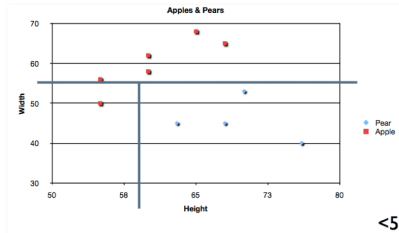
- Each internal node tests a feature or **attribute**.
- Each branch corresponds to an **attribute value**.
- Each leaf node assigns a **classification**.

	Greenness	Height	Width	Taste	Weight	Height/Width	Class
No. 1	210	60	62	Sweet	186	0.97	Apple
No. 2	220	70	53	Sweet	180	1.32	Pear
No. 3	215	55	50	Tart	152	1.10	Apple
No. 4	180	76	40	Sweet	152	1.90	Pear
No. 5	220	68	45	Sweet	153	1.51	Pear



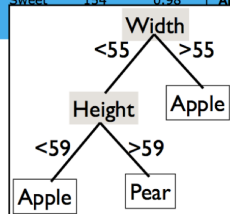
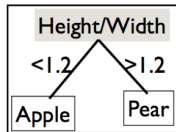
1	0.96	Apple
0	1.40	Pear
4	0.98	Apple
1	1.05	Apple
4	1.03	Apple





- A better answer is possible. . .

	Greenness	Height	Width	Taste	Weight	Height/Width	Class
No. 1	210	60	62	Sweet	186	0.97	Apple
No. 2	220	70	53	Sweet	180	1.32	Pear
No. 3	215	55	50	Tart	152	1.10	Apple
No. 4	180	76	40	Sweet	152	1.90	Pear
No. 5	220	68	45	Sweet	153	1.51	Pear
No. 6	160	65	68	Sour	221	0.96	Apple
No. 7	215	63	45	Sweet	140	1.40	Pear
No. 8	180	55	56	Sweet	154	0.98	Apple
No. 9	220	68	65				Apple
No. 10	190	60	58				Pear



Top-Down Induction of Decision Trees: ID3 Algorithm

Top-Down Induction of Decision Trees

DT-Learn(Examples)

- ▷ Choose *best* attribute.
 - ▷ Extend tree by adding a new node for this attribute and a new branch for each attribute value.
 - ▷ Sort training examples down through this node to the current leaves.
 - ▷ If the training examples are unambiguously classified then stop.
 - ▷ Otherwise repeat from the top.
-

Decision Tree Example

Learn a DT for the following training examples.

Should I play tennis? Yes (+) or no (-)...

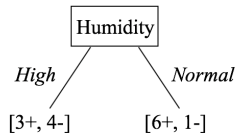
Day	Outlook	Temp.	Humidity	Windy	Class
D1	Sunny	Hot	High	False	-
D2	Sunny	Hot	High	True	-
D3	Overcast	Hot	High	False	+
D4	Rain	Mild	High	False	+
D5	Rain	Cool	Normal	False	+
D6	Rain	Cool	Normal	True	-
D7	Overcast	Cool	Normal	True	+
D8	Sunny	Mild	High	False	-
D9	Sunny	Cool	Normal	False	+
D10	Rain	Mild	Normal	False	+
D11	Sunny	Mild	Normal	True	+
D12	Overcast	Mild	High	True	+
D13	Overcast	Hot	Normal	False	+
D14	Rain	Mild	High	True	-

Decision Tree Example

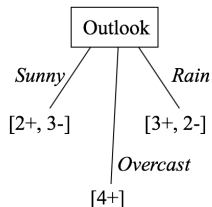
Does the order of attribute selection matter?

Which attribute should be selected first?

Choose Humidity first ...



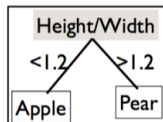
Choose Outlook first ...



Feature Selection Again

What is the best feature/attribute?

- The attribute that makes the most difference to the classification of an example.
- The ideal attribute would divide the examples into sets, each with only one classification, and we would be finished; i.e. an immediate classification of all examples.



- More likely, certain attributes may lead to an immediate classification of some examples but not all; for example Outlook = Overcast.

Feature Selection for Categorical Variables

- Measures such as Pearson correlation test for linear dependence. The **Information Gain** tests more generally for any statistical dependency between variables.
- Information Gain is a measure that is formulated in **Information Theory**, the mathematical theory of communication.
- The **entropy** of a *discrete* random variable x is defined as

$$H(X) = - \sum_{k=1}^K p_k \log p_k$$

where $p_k = \text{Prob}(X = v_k)$, where v_k is one of the K possible values that it can take.

- One interpretation of entropy is that it is a *lower bound* on the minimum number of bits required on average to send a “message” in which the message symbols are made up of the values v_k .

Information Gain

- Now consider two random variables X and Y , we could ask, what is the minimum number of bits required to encode Y , *knowing* that X has already occurred.
- Intuitively, if Y always occurs whenever X occurs that knowing X is enough to know that Y occurred and 0 extra bits are required.
- Similarly if the occurrence of Y has *nothing to do* i.e. is **independent** of the occurrence of X , then knowing that X occurred is not at all helpful. The minimum number of bits is determined by the entropy of Y .
- In general, the more dependent Y is on X , the fewer the number of bits required to encode Y , when X is known.

Information Gain I

- The **Mutual Information** between two random variables X and Y is defined as

$$I(Y; X) = H(Y) - H(Y|X)$$

where

$$H(Y|X) = - \sum_x p(x) \sum_y p(y|x) \log p(y|x)$$

- Averaging over all possible values x that X can take, we find the entropy of Y , knowing that $X = x$.
- In the case of feature selection for decision trees, X is the feature and Y is the classification label.
 - $p(x)$ is the fraction of samples for which feature $X = x$
 - $-\sum_y p(y|x) \log p(y|x)$, is the entropy of the label, considering only those samples for which $X = x$.

- Looking at the Decision Tree Example above, we note that

$$p(x) = p(\text{sunny}) = 5/14$$

$$p(y|x) = p(+1|\text{sunny}) = 2/5$$

$$p(y|x) = p(-1|\text{sunny}) = 3/5$$

$$p(x) = p(\text{overcast}) = 4/14$$

$$p(y|x) = p(+1|\text{overcast}) = 4/4$$

$$p(y|x) = p(-1|\text{overcast}) = 0/4$$

$$p(x) = p(\text{rain}) = 5/14$$

$$p(y|x) = p(+1|\text{rain}) = 3/5$$

$$p(y|x) = p(-1|\text{rain}) = 2/5$$

$$\begin{aligned}
 H(Y) &= -\left(\frac{9}{14} \log \frac{9}{14} + \frac{5}{14} \log \frac{5}{14}\right) \\
 &= -\frac{1}{14}(9 \log 9 + 5 \log 5) + \log 14 \\
 &= 0.940
 \end{aligned}$$

$$\begin{aligned}
 H(Y|X = \text{sunny}) &= -\left(\frac{2}{5} \log \frac{2}{5} + \frac{3}{5} \log \frac{3}{5}\right) \\
 &= -\frac{1}{5}(2 \log 2 + 3 \log 3) + \log 5 \\
 &= 0.971
 \end{aligned}$$

$$\begin{aligned}
 H(Y|X = \text{overcast}) &= -\left(\frac{4}{4} \log \frac{4}{4} + \frac{0}{4} \log \frac{0}{4}\right) \\
 &= -\frac{1}{4}(4 \log 4 + 0 \log 4) + \log 4 \\
 &= 0
 \end{aligned}$$

$$\begin{aligned}
 H(Y|X = \text{rain}) &= -\left(\frac{3}{5} \log \frac{3}{5} + \frac{2}{5} \log \frac{2}{5}\right) \\
 &= -\frac{1}{5}(3 \log 3 + 2 \log 2) + \log 5 \\
 &= 0.971
 \end{aligned}$$

Therefore

$$\begin{aligned}
 I(Y; \text{outlook}) &= H(Y) - \left(\frac{5}{14} 0.971 + \frac{4}{14} 0 + \frac{5}{14} 0.971\right) \\
 &= 0.246 \text{ bits}
 \end{aligned}$$

- Looking at the Decision Tree Example above, we note that

$$\begin{aligned}p(x) &= p(\text{hot}) &&= 4/14 \\p(y|x) &= p(+1|\text{hot}) &&= 2/4 \\p(y|x) &= p(-1|\text{hot}) &&= 2/4 \\p(x) &= p(\text{mild}) &&= 6/14 \\p(y|x) &= p(+1|\text{mild}) &&= 4/6 \\p(y|x) &= p(-1|\text{mild}) &&= 2/6 \\p(x) &= p(\text{cool}) &&= 4/14 \\p(y|x) &= p(+1|\text{cool}) &&= 3/4 \\p(y|x) &= p(-1|\text{cool}) &&= 1/4\end{aligned}$$

$$\begin{aligned}
 H(Y|X = \text{hot}) &= -\left(\frac{2}{4} \log \frac{2}{4} + \frac{2}{4} \log \frac{2}{4}\right) \\
 &= -\frac{1}{4}(2 \log 2 + 2 \log 2) + \log 4 \\
 &= 1
 \end{aligned}$$

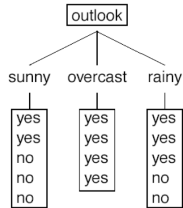
$$\begin{aligned}
 H(Y|X = \text{mild}) &= -\left(\frac{4}{6} \log \frac{4}{6} + \frac{2}{6} \log \frac{2}{6}\right) \\
 &= -\frac{1}{6}(4 \log 4 + 2 \log 2) + \log 6 \\
 &= 0.9183
 \end{aligned}$$

$$\begin{aligned}
 H(Y|X = \text{cool}) &= -\left(\frac{3}{4} \log \frac{3}{4} + \frac{1}{4} \log \frac{1}{4}\right) \\
 &= -\frac{1}{4}(3 \log 3 + 1 \log 1) + \log 4 \\
 &= 0.8113
 \end{aligned}$$

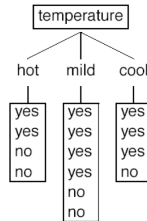
Therefore

$$\begin{aligned} I(Y; \text{temperature}) &= H(Y) - \left(\frac{4}{14}1 + \frac{6}{14}0.9183 + \frac{4}{14}0.8113 \right) \\ &= 0.0289 \text{ bits} \end{aligned}$$

Comparing Root nodes for the weather data set:



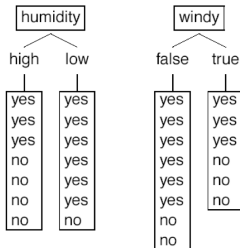
$$\text{Gain}(S, \text{outlook}) = 0.25$$



$$\text{Gain}(S, \text{temperature}) = 0.03$$

- In a similar manner, you can check:

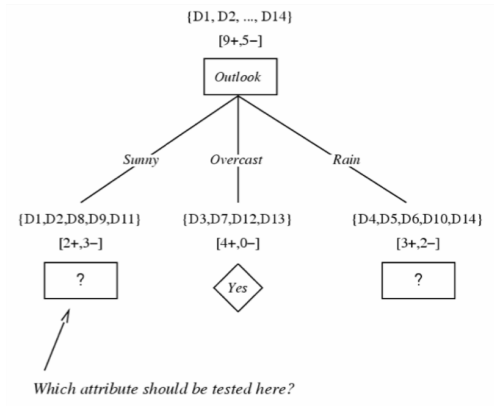
Comparing Root nodes for the weather data set:

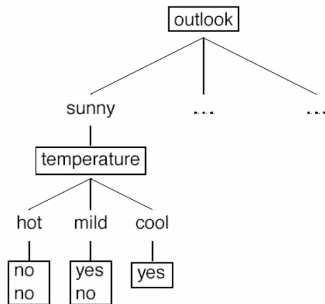


$$\text{Gain}(S, \text{humidity}) = 0.15$$

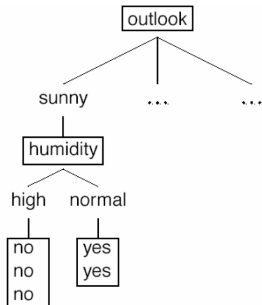
$$\text{Gain}(S, \text{windy}) = 0.05$$

outlook achieves
highest information
gain \Rightarrow should be used
as root for decision tree

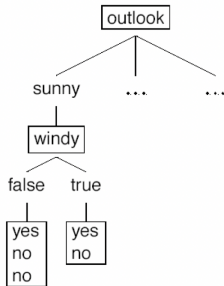




$$\text{Gain}(S_{\text{sunny}}, \text{temperature}) = 0.970 - (2/5) 0.0 - (2/5) 1.0 - (1/5) 0.0 = 0.570$$



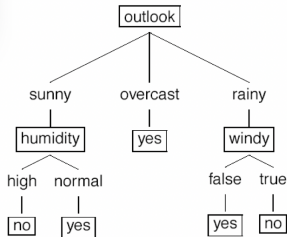
$$\text{Gain}(S_{\text{sunny}}, \text{humidity}) = 0.970 - (3/5) 0.0 - (2/5) 0.0 = 0.970$$



humidity achieves
highest information
gain \Rightarrow should be
chosen for outlook =
sunny

$$\text{Gain}(S_{\text{sunny}}, \text{windy}) = 0.970 - (2/5) 1.0 - (3/5) 0.918 = 0.019$$

Final decision tree for the weather data:



Note: all leaf nodes are associated with training examples from the same class (entropy = 0).

Note: the attribute temperature is not used.

Decision Trees

- We can't always expect a decision tree to yield a consistent learner.
- Sometimes, we may have used all the features and still have leaf nodes containing both positive and negative instances.
 - In this case, the decision tree will report the majority class.
- Similarly, a decision tree might **overfit** the data – irrelevant attributes are used to make spurious distinctions among the examples.
 - An Information Gain ≈ 0 indicates that whatever dependence is seen may not be significant.

1 Problem Solving by Search

- Uninformed Search
- Informed Search
- Adversarial Search
- Game Playing with Reinforcement Learning

2 Optimisation

- Optimisation Overview
- Combinatorial Optimisation Problems
- Simulated Annealing
- Optimisation Problem Examples
- Convergence of Simulated Annealing
- Genetic Algorithms
- Optimisation in Continuous Spaces

3 Machine Learning

- Supervised Machine Learning
- Feature Selection
- Decision Trees
- Neural Networks

Neural Networks

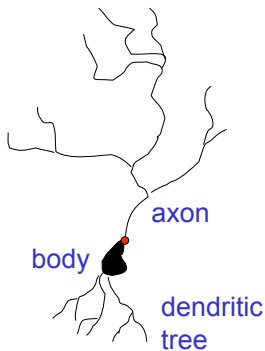
- Neural networks arose initially from a researchers who wished to model how the human brain works.
- The idea was to build a simple model of a neuron, and then to connect these models together to form model of the brain.
- The question arises, how is it possible to train such a system to learn to perform tasks such as classification.
- Many neural networks are trained in a **supervised** manner, by presenting them with examples for which the output is known.
- In this framework, they are nothing more than another example of a supervised classifier.

Facts about the brain

- The brain consists of around 10^{11} neurons.
- Neurons are connected: each neuron receives between 10^3 and 10^4 connections. Hence there are 10^{14} to 10^{15} connections in the brain (100-1000 Tbytes to store 1 number for each of them).
- The cortex consists of two laminar sheets of nerve cells about 2 mm thick.
- The "currency" of the brain is the action potential or voltage spike.
- There appears to be considerable localisation of function in the brain.

A typical cortical neuron

- **Gross physical structure:**
 - One axon that branches
 - A dendritic tree that collects input from other neurons
- **Axons typically contact dendritic trees at synapses**
 - A spike of activity in the axon causes charge to be injected into the post-synaptic neuron
- **Spike generation:**
 - Outgoing spikes whenever enough charge has flowed in at synapses to depolarise the cell membrane

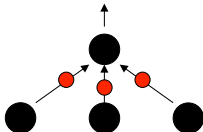


Synapses

- **When a spike travels along an axon and arrives at a synapse it causes transmitter chemical to be released**
 - There are several kinds of transmitter
- **The transmitter molecules diffuse across the synaptic cleft and bind to receptor molecules in the membrane of the post-synaptic neuron thus changing their shape.**
 - This opens up holes that allow specific ions in or out.
- **The effectiveness of the synapse can be changed**
 - vary the amount of transmitter
 - vary the number of receptor molecules.
- **Synapses are slow, but they have advantages over RAM**
 - Very small
 - They adapt using locally available signals (but how?)

How the brain works (actually, we don't know precisely)

- Each neuron receives inputs from other neurons
 - Cortical neurons use spikes to communicate
 - The timing of spikes is important
- The effect of each input line on the neuron is controlled by a synaptic weight
 - The weights can be positive or negative



- The synaptic weights **adapt** so that the whole network learns to perform useful computations
 - Recognising objects, understanding language, making plans, controlling the body
- A huge number of weights can affect the computation in a very short time. Much better bandwidth than computers.

Modularity and the brain

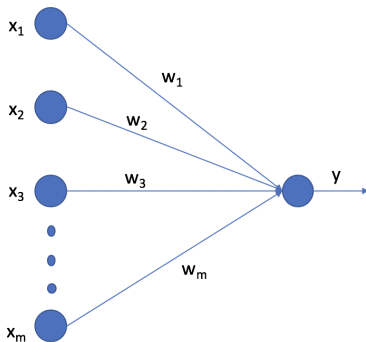
- **Different bits of the cortex do different things.**
 - Local damage to the brain has specific effects
 - Specific tasks increase the blood flow to specific regions.
- **But cortex looks pretty much the same all over.**
 - Early brain damage makes functions relocate
- **Cortex is made of general purpose stuff that has the ability to turn into special purpose hardware in response to experience.**
 - This gives rapid parallel computation plus flexibility
 - Conventional computers get flexibility by having stored programs, but this requires very fast central processors to perform large computations.

Idealised neurons

- **We want to model neurons in an idealised fashion**
 - Removing complicated details that are not essential for understanding the main principles.
 - Allowing us to apply mathematics.
 - Complexity can always be added
- **It is often worth understanding models that are known to be wrong (but we mustn't forget that they are wrong!)**
 - E.g. neurons that communicate real values rather than discrete spikes of activity.
 - E.g. neurons that do stuff in predetermined moments (with a clock) instead of whenever they want.

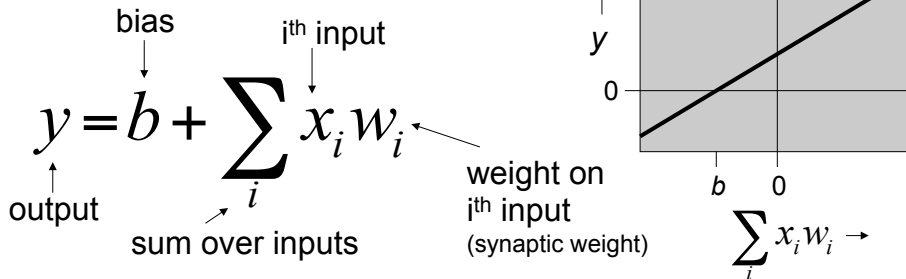
The perceptron

- The **single layer perceptron** is nothing more than a linear classifier such as the ones we've studied already.



Linear neurons

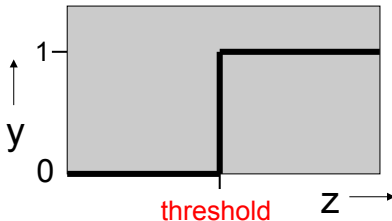
- Simple but limited
 - If we can make them learn we **may** get insight into more complicated neurons



Binary threshold neurons

- **McCulloch-Pitts (1943):**
 - First compute a weighted sum of the inputs from other neurons
 - Then send out a fixed size spike of activity *if the weighted sum exceeds a threshold.*
 - Maybe each spike is like the truth value of a proposition and each neuron combines truth values to compute the truth value of another proposition.

$$z = \sum_i x_i w_i$$
$$y = \begin{cases} 1 & \text{if } z > \theta \\ 0 & \text{otherwise} \end{cases}$$

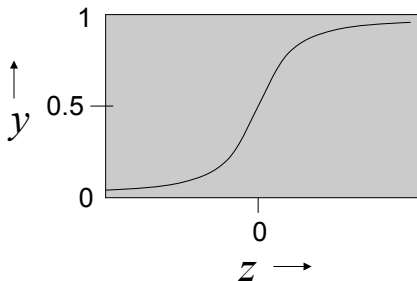


Sigmoid neurons

- These give a real-valued output that is a smooth and bounded function of their total input.
 - Typically they use the logistic function or tanh function
 - They have nice derivatives which is why we like them.
- If we treat y as a probability of producing a spike: stochastic binary neurons
- Otherwise, simply a neuron that can be spiking a bit..

$$z = b + \sum_i x_i w_i$$

$$y = \frac{1}{1 + e^{-z}}$$



Activation Functions

$$f(z) = z \quad \text{linear}$$

$$f(z) = \text{step}(z) \quad \text{McCulloch - Pitts}$$

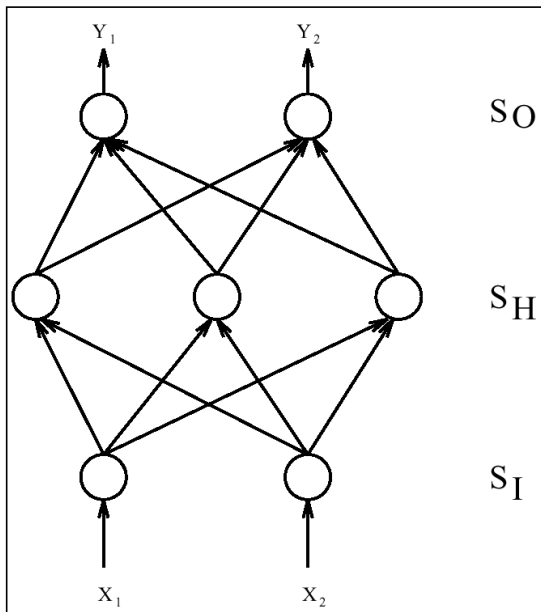
$$f(z) = \sigma(z) \quad \text{sigmoid}$$

$$\text{step}(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

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

Supervised Learning

- *Hebbian Learning* “When an axon of cell A is near enough to excite cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A’s efficiency as one of the cells firing B, is increased .” (Hebb, 1949)
- Rosenblatt (1957) implemented something close to Hebbian learning in a McCulloch-Pitts neuron – from a mathematical perspective this was nothing more than gradient descent, on a squared error loss function.
- In the 1980’s Rumelhart, Hinton & Williams generalised this algorithm to **multilayer perceptrons**, so that weights are learned by “back propagation” from the output layer back to the previous layers.



Neural Networks and Classification

- A **multilayer perceptron** is a set of perceptrons connected together in layers.
- The inputs to any node determine whether that node is activated, which in turn propagates to the nodes to which it is connected.
- A Neural Network is typically trained in a **supervised** manner, by adjusting its weights to return desired outputs for a set of training instances.

Neural Networks and Classification I

- Neural network can be used to perform classification, just like the classifiers already discussed e.g. each output neuron can correspond to a class
- The challenge is that the Neural Network has a weight on each link connecting pairs of neurons. We must select these weights appropriately.
- As above, we can choose the weights to minimise a loss function, such as the squared loss, in a supervised algorithm in which the network is presented with input feature vectors and their desired class.
- The weights can be learned through the gradient descent **back propagation** algorithm.
- Neural networks provide a non-linear, complex mapping between input and output.

Neural Networks and Classification II

- Deep learning is a hot topic in AI research – it involves new algorithms for exploiting neural networks to solve complex real-world problems

MLP applications: handwritten digit recognition

- **“Hand-written digit recognition with a back-propagation network”, Le Cun et al. 1990**
- **Multi-layer perceptron applied to handwritten digits.**
- **Relatively little non-connectionist preprocessing: digits split, centred, normalised.**

The sets

- 9298 digits from letters passing through the Buffalo office of the US PS + 3349 printed digits from 35 fonts.

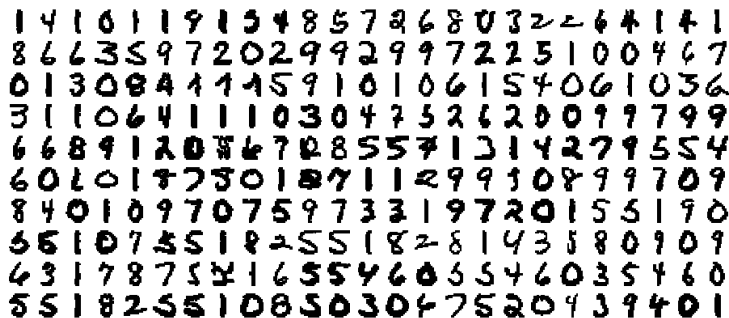
- Training set: 7291+2549

- Test set: 2007+700

40004 75246
14199-2087 23505
96203 14310
44151 05153

preprocessing

- Enough to obtain digits in this format, each one is a 16x16 greyscale image:



1 4 1 0 1 1 9 1 3 4 8 5 7 2 6 8 0 3 2 2 6 4 1 4 1
8 6 6 3 5 9 7 2 0 2 9 9 2 9 9 7 2 2 5 1 0 0 4 6 7
0 1 3 0 8 4 1 1 1 5 9 1 0 1 0 6 1 5 4 0 6 1 0 3 6
3 1 1 0 6 4 1 1 1 0 3 0 4 7 5 2 6 2 0 0 9 9 7 9 9
6 6 8 9 1 2 0 8 6 7 2 8 5 5 7 1 3 1 4 2 7 9 5 5 4
6 0 2 0 1 7 7 5 0 1 8 7 1 1 2 9 9 1 0 8 9 9 7 0 9
8 4 0 1 0 9 7 0 7 5 9 7 3 3 1 9 7 2 0 1 5 5 1 9 0
5 5 1 0 7 5 5 1 8 2 5 5 1 8 2 8 1 4 3 5 8 0 9 0 9
4 3 1 7 8 7 5 2 1 6 5 5 4 6 0 5 5 4 6 0 3 5 4 6 0
5 5 1 8 2 5 5 1 0 8 5 0 3 0 4 7 5 2 0 4 3 9 4 0 1

The network

- In theory one could simply feed the image into a Multi-layer perceptron with M hidden units.
- The input would be an array of $16 \times 16 = 256$ real values, the output an array of ten values, one for each class (0,1,..9).
- This means $(256+1)M + (M+1)10 = 267M + 10$ weights. If $M=50 \rightarrow 13360$ weights, fairly large for 10k examples (and difficult to train in 1990).

The network

- Le Cun et al. design a network with:
- partial connectivity, i.e. not all the units in layer i are connected to all the units in layer $i+1$.
- weight sharing, i.e. different parts of the networks are forced to use the same weights.

Training the network

- Training is by gradient descent, using backpropagation.
- For each copy j of a shared weight there will be a Δw_j . They are simply added together.

Results on Handwriting Problem

- Le Cun's method produced a 1.1% error on the training set.
- The test set error was 3.4%.
- Examples that the NN failed to correctly classify were also difficult for humans to decipher.