Notebook III

David Hui

23rd October 2015 – 13th July 2016

Contents

Chapter 1

Machine Learning

1.1 Machine Vision

- Edge Detection $\implies$ Colour in regions $\implies$ Classification

Selective search: recursively binary divide canvas into patches of single colours for object detection.

1.2 Routes to Intelligence

Based on The Master Algorithm, 2015, Domingos, P. Problem with algorithms is that too much is remembered, it is unable to generalise and forget less important factors. A greater intelligence thinks of a greater number of theories which requires an algorithm of greater complexity for comparisons. Five families of Machine Learning:

1.2.1 Symbolism

Utilises philosophy and logic to generate multiple theories. Discard theories that do not fit by contradiction. Favoured algorithm is Hume's method of generalisation for **induction**.

1.2.2 Connectionism

Is basic physics applied to basic neuroscience. Similar to Symbolism in terms of having many theories, except correct theories are strengthened and none are discarded. Favoured algorithm is Neural Network.

1.2.3 Evolution

Can be used to generate structures of neural networks.

Generally, things evolve such that they can do things more efficiently, and thus in a shorter space of time. There is a greater flux of energy per unit time. Organisms that do not do things as efficiently squander resources and do not survive. Organisms which reproduce in a quicker space of time overcrowd and overcompete against those who do not. In addition, a faster reproductive period leads to a faster rate of mutation.

1.2.4 Statistics

Logarithm of Naive Bayes is a Neural Network. A hidden Markov model is a discrete version of a Kalman Filter. Resolves exceptions to rules. Favoured algorithm is Baysean Inference.

1.2.5 Analogiser

Finds similarities between data and clumps them together. The opposite of Symbolism. Favoured algorithm is Support Vector Machine.

1.2.6 Linguistic Models

Utilise internal language – use unsupervised learning to map sensing to Internal model. Can use probabilities to map an internal language with a natural language. This generalises/abstracts by taking raw inputs and representing it by a word, denoising data. Another method to do this is by establishing upper and lower bounds on values – the interval allows leeway in the number.

- Mathematics is a 1:1 function. The recipient is able to understand perfectly what is being conveyed.
- Language is a 1:many function. The recipient each has their own individual understanding of words and can relate differently.
- Other artforms such as fine art and art music are many:many and pride themselves on obscurity.

English: Subject verb object

The choice of word is determined by emotional attachment to the action. A system learns new words by statistical correlation with an action.

A statement in a programming language is taken to be immutable. English rules are not. Function related to language. Language enables consciousness – generalisation and specialisation – that mathematics cannot provide.

1.2.7 Comparison

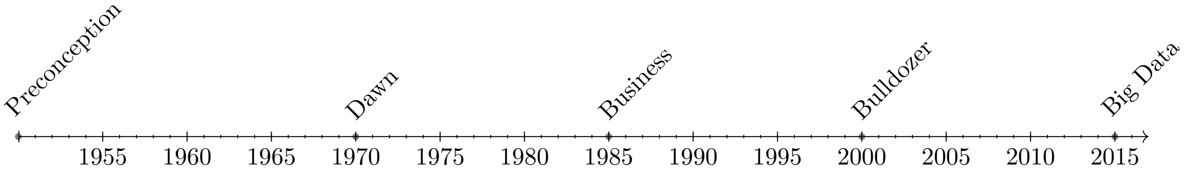
Symbolic and Connectionist algorithms require omniscience, as the number of possible theories are infinite and there is always a bias towards training data. Connectionist algorithms are too specific and generally overfits. Evolutionary algorithms are time-consuming and are inelastic. Statistical algorithms are inefficient as they require all probabilities to be calculated. A repeat of a previous data point may lead to wasted computation. All of the above are parametric models. Analogising algorithms are not.

Chapter 2

Symbolic Artificial Intelligence

Based on Winston, P.

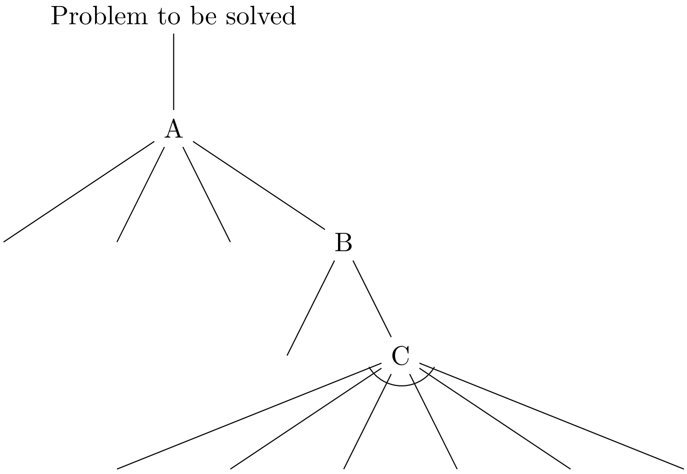
Artificial Intelligence models representations of thinking, perpection and action.



Symbolic AI seeks to transform: problem ⟶ known problem ⟶ solution. A more intelligent solution has a low 'depth of functional complexity', or fewest functional composities.

2.0.1 Expert Systems

An expert system behaves perfectly if it is omniscient, and if it runs forever. Obviously, both critera cannot be met. The behaviour of an expert system can be visualised using a goal tree, such as the one below.



In order to solve a problem, an expert system transforms the problem into a series of subgoals, then recursively repeats this process until no more goals remain. Branches from a node in a goal tree represent the subgoals that need to be done. Nodes with an arc connecting the branches signify an AND node, where all of its subgoals need to be completed, whereas nodes without are an OR node, of which only one subgoal needs to be completed. Obviously, tasks need to be completed from the bottom up.

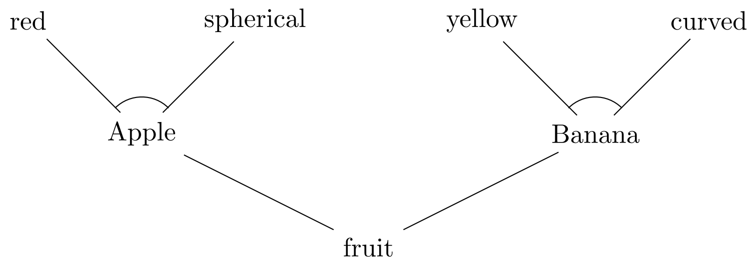
Goal trees also provide answers to 'how' and 'why' questions.

- Why did you do B? – *In order to do A.*
- How did you do B? – *By doing C.*

Generally, complexity(behaviour) = max [complexity(program), complexity(environment)]

Forward-Chaining Rule Based Expert System

An (albeit incomplete) example of the system below classifies objects by deduction through subgoals. Essentially, we are going up the goal tree.



The goal tree represents the following predicate.

$$(\text{red} \land \text{spherical}) \lor (\text{yellow} \land \text{curved})$$

Indeed, a Naive Bayes algorithm can be run on the predicate to obtain a final probablisitic result. The advantage of an expert system over probabilistic methods is the encapsulation of the terms. By letting

$$\text{apple} = \text{red} \land \text{spherical}$$
$$\text{banana} = \text{yellow} \land \text{curved}$$

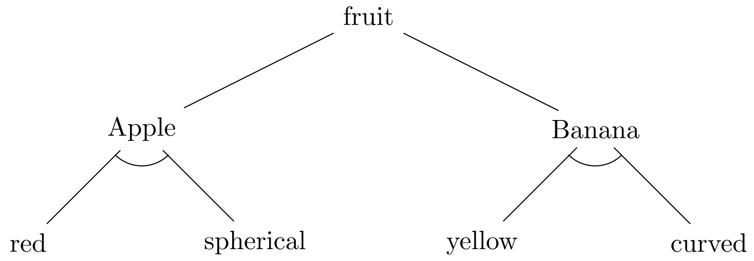
we obtain

$$\text{fruit} = \text{apple} \lor \text{banana}$$

Naming the subnodes allows their probabilities to be computed only once, saving computaton.

Backwards-Chaining Rule Based Expert System

This is the reverse of a forwards-chaining system. It can inductively find the predicate from the object of interest.



Knowledge Engineering

Together, both expert systems form the basis of knowledge engineering.

1. Rules and structures are discovered by backwards chaining.
2. Goals are achieved using forward chaining.
3. To find rules, look at specific cases and then generalise.
4. Method of differences – differences in seemingly identical objects.
5. Find new rules from contradiction or failure of program.

2.0.2 Searching

We can use searching algorithms within a tree to find a path to the goal. The shortest path is advantageous in a foward-chaining system as it yields a strategy to complete the goal with the fewest number of steps or subgoals. In a backward-chaining system, it yields the most probable rule by force of Ockham's razor. What follows are sorting algorithms in order of increasing speed and sadly increasing prerequisite knowledge about the tree.

PSEUDOCODE ME

British Museum

Find all paths, make tree, do choice.

Depth-first search

Place new nodes on front of queue. In low-memory situations, explore a different path to build up a framework of surroundings.

Breadth-first search

Place new nodes on back of queue. Guaranteeed to find shortest route first. If the paths are weighted, they can be converted into many paths of length 1. If a negative weight is reached, use depth-first search on all bottom nodes until all weights are congruent and non-negative.

Both depth-first and breadth-first search are bad as they may visit the same node twice, wasting computation. So the speed of algorithms can be increased by keeping track of which nodes have been visited.

Hill Climbing

Expand node that is closer to goal if multiple branches. This is Depth-first queue with a front-sorted queue.

Beam Search

Restrict branching factor at each node to a constant and keep those constant to the target. This is an improvement of Breadth-first search, but the order of node placement does not matter.

Best-first search

Find node with shortest distance to goal and expand it.

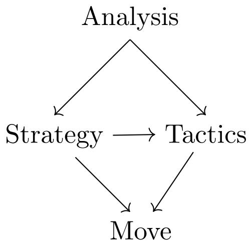
The distance to the goal is only an estimate, and is known as a heuristic. In addition for the heuristic being fallable, there are also three further problems to these faster sorting algorithms.

1. The goal is a global maximum, but the heuristic provides a route to a local maximum.
2. There is a sharp ascent to the goal, thus the heuristic is only useful within the vicinity of the ascent.
3. At plateaus and aretes, the heuristic provides no meaningful information.

2.0.3 Game Playing

In 1972, Hubert Dreyfuss claimed that a computer cannot beat a human at chess. In 1997, Deep Blue beat Gary Kasparov, the World Champion. Dreyfuss claims that it is impossible for computers to defeat a human because they think in different ways. Intelligence is subjective and is context-sensitive. Thus a human cannot objectively discern their own intelligence because they are subject to their own intelligence.

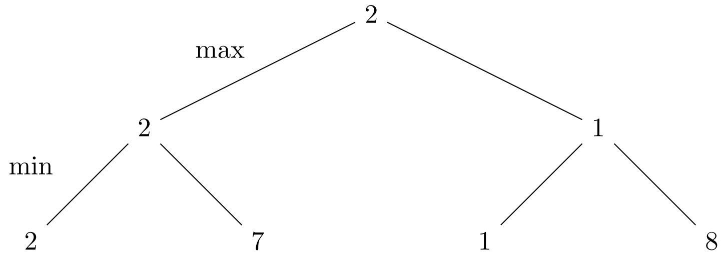
Nevertheless, the diagram below shows ways in which a human can analyse a situation.



A method of representing this thought process is using a minimax tree. The minimax tree was independently discovered by Alan Turing and Claude Shannon. The proceedure is best adapted for two player games.

Minimax Proceedure

1. If your turn, generate all possible moves from all nodes and add to tree as branches. If opponent's turn, generate all opponents moves and add to tree as branches.
2. Repeat (1) until depth, memory or time limit reached.
3. For each terminal node, compute score of situation. A higher score indicates favourability.
4. If your turn, go to the parent node and assign the node the maximum score of the children. If opponent's turn, go to parent node and assign value of minimum score of children.
5. Repeatedly assign values to parent nodes until root node reached.



The proceedure should yield a route from the current situation to the optimal result, providing that the opponent has an equal intellect.

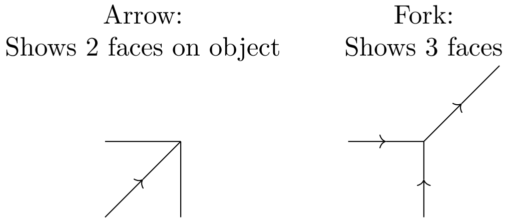
α-β pruning

If the maximum is greater than the minimum of the other branch, do not compute the other branch

2.0.4 Machine Vision

Early advancements in Machine Vision came from Guzman, who saught to program a computer to recognise wooden blocks and their orientations. To simplify complexity, the faces of the blocks were painted white and the edges black, such that they can be easily recognised. In addition, only isomorphic pictures were taken of the blocks.

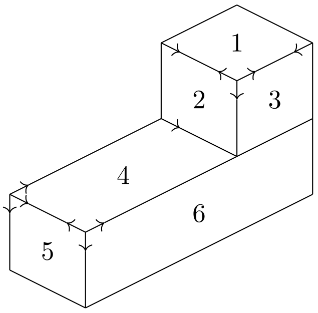
When viewing blocks, their vertices produce two junctions.



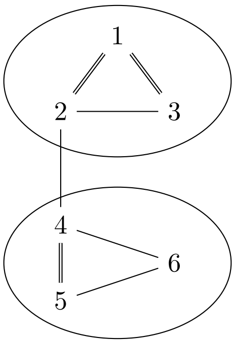
Guzman's Algorithm

1. On picture of object, classify each vertex.
2. Label each face.
3. Construct graph showing how the linkage of faces with lengths.
4. If the faces are linked by 2 or more lengths, it is the same object.

After labelling all vertices, faces and edges, the following diagram is constructed.



From there, the following diagram representing faces are drawn.



It is clear to see that the diagram represents two distinct objects.

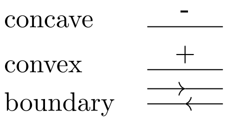
2.0.5 Huffman's Algorithm

Huffman's algorithm is a refinement of Guzman's algorithm projects a 3D object into a graph.

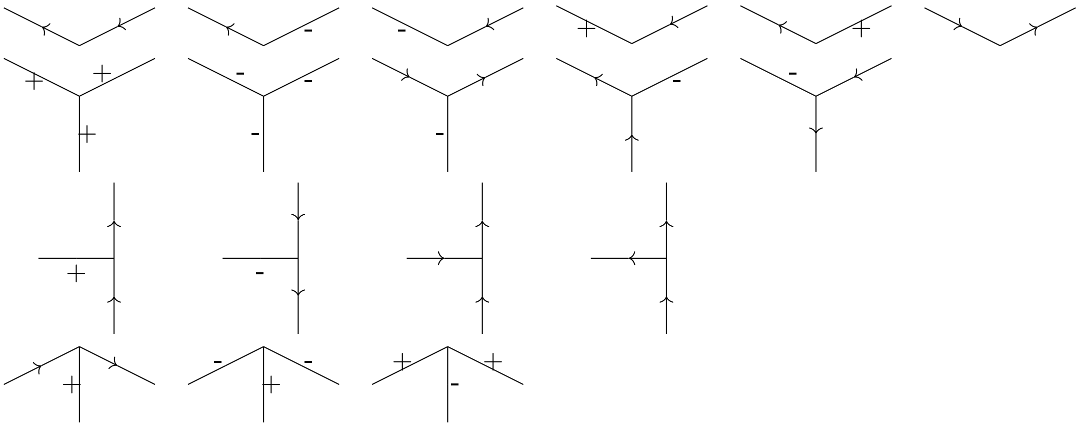
Vertices ⟶ Junctions

Edges ⟶ Lines

There are 4 kinds of label. It is conventional to place the label to the left of the object.

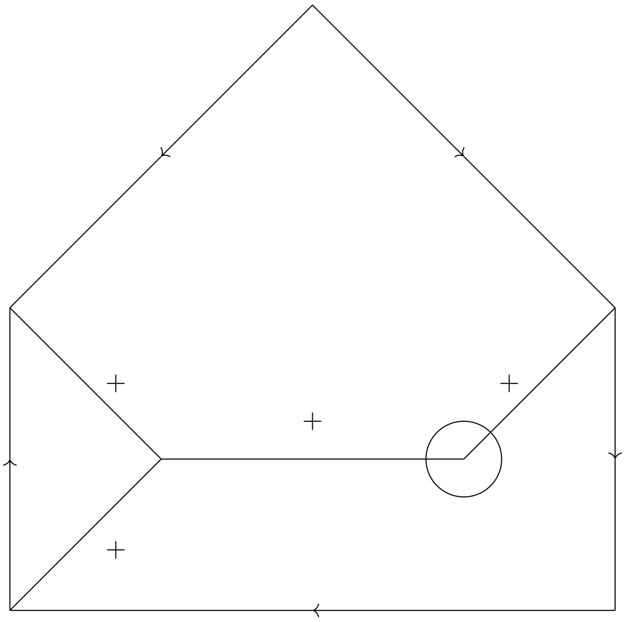


These four types of line generate 18 types of trihedral vertex.



1. Boundaries labelled first keeping object on right.
2. Label edges. If edge starts / ends with a different sign, reject.
3. Check all vertices present, or reject.

Using Huffman's algorithm, we generate:



The circled vertex cannot possibly exist and so, neither can the diagram.
Unfortunately, Huffman's algorithm cannot deal with more than 3 faces at a point.

2.0.6 Waltz's Algorithm

Waltz generalised the number of possible things a computer could see, and created 50 labels, including cracks, light, shadow and non-trihedral vertices.

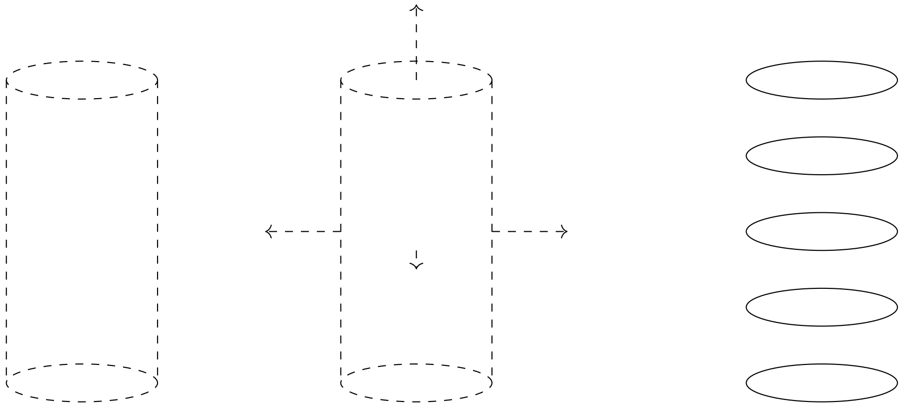
1. Label all features on diagram.
2. Choose viable subset of all features to obtain representation.

2.0.7 Recognising Objects

Marr's Algorithm

The algorithm provides a way of abstracting objects. All objects can be matched up with generalised cylinders. Generalised cylinders store all possible combinations of objects in memory. Theoretically, this will not take up much space because it is a high level of abstraction.

Objects	⟶	edges	⟶	edges, faces and normals	⟶	Recognition
		Primal		**2.5 D**		**Generalised Cylinders**



Orthographic Projection

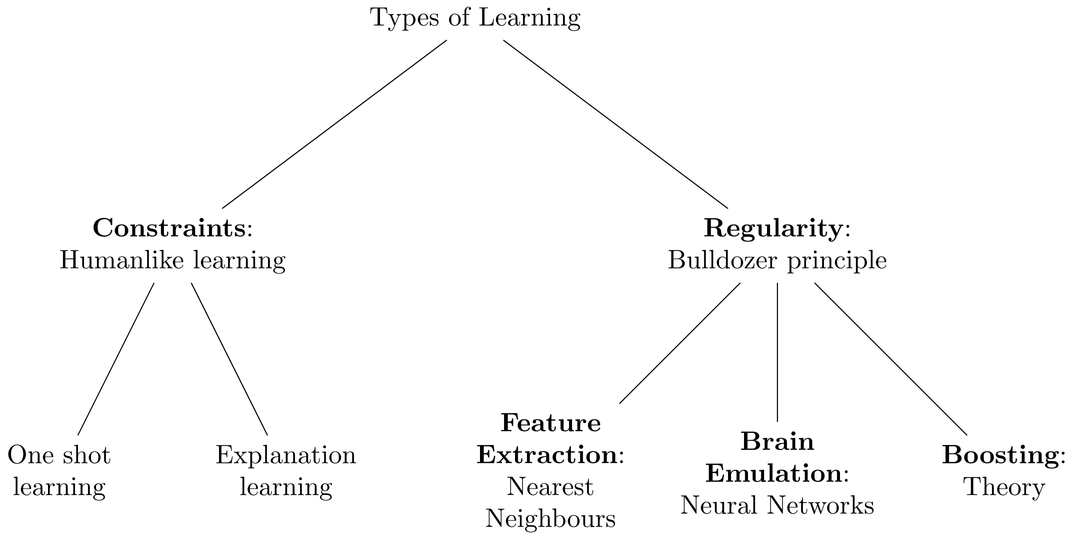
PICTURE LAZY TIKZ
From feature points in n different orientations of objects, take n coordinates of the same feature in different orientations and solve simultaneously. If unique solution, then the item is the same. Coordinates may not necessarily be the same.

Comparing subsections

Compare subsections and features to each other. May also determine what is missing in addition.

Can also integrate functon – find X, Y such that $\int_{x,y} f(x,y)g(x-X, y-Y)$ is maximised. When the functions are aligned, the number is large.

2.1 Learning


Types of Learning
Constraints: Humanlike learning
Regularity: Bulldozer principle
One shot learning
Explanation learning
Feature Extraction: Nearest Neighbours
Brain Emulation: Neural Networks
Boosting: Theory

Chapter 3

Neural Networks

Based on Nielsen, M., Deep Learning and Neural Networks, 2015

Neural networks are biologically inspired and topologically model a biological neural network. Both artificial and biological neural networks consist of neurons, whose interconnections form the aforementioned topology. Biological networks have a more amorphous structure than artificial networks, which has an intermediate level of abstraction. Artificial networks are initially abstracted into layers, which are an ensemble of neurons run in parallel. In a feedforward network, the output of a layer is an input for the subsequent layer. Networks where outputs of neurons can go to any layer, including itself are recurrent networks.

The hierarchy enables a network to learn and approximate more complicated functions than a single neuron. The training cycle consists of two steps, the forward pass and the backwards pass. The forward pass computes the output of the neurons layer by layer. The output of the whole network is the output of the final, or output layer. Layers between the first and output layer are called hidden layers as their outputs are not seen. A cost function compares the difference between the observed and expected outputs. The backwards pass backpropagates the difference, which then adjusts the parameters of the neurons by gradient descent.

The human brain has six cortices. V1, the Primary Visual Cortex is located at the back of the brain and has 140×10^6 neurons with 10^{10}. V1 is mainly used in image recognition. In contrast, GoogLeNet, an artificial image classifier, has 10^6 neurons.

Deep Learning techniques are a generalisation of neural networks. They have more layers, more neurons, more parameters and have fewer rules about connections and structure of layers.

They are very useful in recognition, but cannot generalise to other features easily because it is purely mathematical rather than descriptive. Method needs to be found in order to extract logical rules from network.

3.1 Neuron

(The next iteration should describe how an individual neuron is trained) A neuron is a binary classifier that divides a space into two regions. $\mathbf{x}$ is the input vector. It is augmented by a weight vector $\mathbf{w}$ and bias b. Values within $\mathbf{w}$ and bias b are parameters which change and depend on training data.

$$a = \phi(z) = \phi(\mathbf{w} \cdot \mathbf{x} + b)$$

As shown above, the output value, z is passed into an activating function, ϕ, which changes the behaviour of the network.

3.2 Activating Functions

3.2.1 Perceptron – Binary Neuron

Modelled using Heaviside Step Function.

$$(z) = \begin{cases} 1, & \text{if } z > T \text{, the threshold value} \\ 0, & \text{otherwise} \end{cases}$$

Perceptrons can form AND, OR and NOT gates, and can thus be analysed by Boolean logic.

3.2.2 Sigmoidal Neuron

Utilises the output from the logistical growth model.

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

3.2.3 Hyperbolic tangent

A variant of a Sigmoidal Neuron.

$$\begin{aligned}\tanh(z) &= 2\sigma(z) - 1\\ \tanh(z) &= \frac{1-e^{2z}}{1+e^{2z}}\end{aligned}$$

3.2.4 Rectified Linear Units – ReLU

Otherwise known as a ramp function, the integral of a Heaviside Step Function. As of 2012, this is the most popular activation activiating function.

$$\text{ReLU}(z) = \max(0, z)$$

This is approximated by the softmax function

$$\ln(1+e^z)$$

Fuzzy logic can be used to analyse non-binary neurons.

3.3 Relating Bayes' Theorem with a Sigmoidal Neuron

$$\begin{aligned}P(y_j|x_0, x_1, \ldots, x_n) &= \frac{\displaystyle\prod_{i=0}^{n} P(x_i|y_j)P(y_j)}{\displaystyle\sum_k \left[\prod_{i=0}^{n} P(x_i|y_k)P(y_k)\right]}\\ &= \frac{1}{1+\dfrac{\displaystyle\sum_{k\neq j}\left[\prod_{i=0}^{n} P(x_i|y_k)P(y_k)\right]}{\displaystyle\prod_{i=0}^{n} P(x_i|y_j)P(y_j)}}\end{aligned}$$

Identify the non-unity term with e^{-t} of the logistic function $f(t) = 1/(1+e^{-t})$ to yield

$$t = \ln\left[\frac{P(y_j)}{P(y_k)}\right] + \ln\sum_{k\neq j}\left[\prod_{i=0}^{n}\frac{P(x_i|y_j)}{P(x_i|y_k)}\right]$$

which is similar to a linear function. $\ln\left[\frac{P(y_j)}{P(y_k)}\right]$ is constant throughout the space of x whilst the kernel function $\ln\sum_{k\neq j}\left[\prod_{i=0}^{n}\frac{P(x_i|y_j)}{P(x_i|y_k)}\right]$ is similar to a weight vector as it takes a vector input x and returns a scalar.

3.4 Weight Initialisation

If all the parameters had the same value, then they would all be changed by the same amount during gradient descent and would learn the same features. Thus parameters are randomly initiated, making the neural network algorithm nondeterministic – running the algorithm twice will not produce the same output. The optimal initial weights are a mean weight of 0.6 and a bias of 0.9.

A pseudorandom number generator would generate weights that are uniformly distributed around a region. Due to the central limit theorem, the output of the network would be initially normally distributed. It is easier to train a network with a normal distribution that has a smaller variance. This is because the peak would be sharpened, allowing gradient descent to be more drastic.

3.5 Cost Functions

The cost, C is calculated from the number of training examples, n and each element in the expected vector, $\mathbf{A}$ and the output vector, $\mathbf{Y}$.

3.5.1 Quadratic cost

$$C = \frac{1}{2n} \sum_{i} (a_i - y_i)^2$$

3.5.2 Cross-Entropy cost

$$C = -\frac{1}{2} \sum (y \ln(a) + (1 - y) \ln(1 - a))$$

3.6 Backpropagation

Backpropagation computes the changes in weights and biases such that the cost function is minimised. It is a first-order technique and utilises the multivariable chain rule.

$\frac{\partial C}{\partial z_i^l}$ is the change of cost with respect to neuron i in layer l. It represents how much the output value of the neuron should change in order to minimise the cost. This value is represented by δ_i^l and all the values within a layer constitute its error vector. The error vectors for each layer are calculated from equations 1 and 2. Changes in each weight and bias can be calculated from the error vector.

3.6.1 Error in final layer

The change in the final layer is computed first. a_k^l is the activation value of neuron i, layer l

$$\delta_i^l = \frac{\partial C}{\partial a_i^l} \phi'(z_i^l) \tag{3.1}$$

3.6.2 Error of weights in layer l

The error vector of a layer can only be calculated from the error vector of the higher layer, which gives rise to the name of backpropagation.

$$\delta_i^l = \sum_k w_{k_i}^{l+1} \delta_i^{l+1} \phi'(z_i^l) \tag{3.2}$$

The change for each weight and bias can be calculated from the error vector from the following two equations.

3.6.3 Change of bias

$$\frac{\partial C}{\partial b_i^l} = \delta_i^l \tag{3.3}$$

3.6.4 Change of k^{th} weight

$$\frac{\partial C}{\partial w^l_{i_k}} = a^{l-1}_k \delta^l_i \tag{3.4}$$

Equation 1

$$\begin{aligned}
\delta^l_i &= \frac{\partial C}{\partial z^l_i} \\
&= \sum_k \frac{\partial C}{\partial a^l_k} \frac{\partial a^l_k}{\partial z^l_i} \\
&= \frac{\partial C}{\partial a^l_j} \cdot \frac{\partial a^l_j}{\partial z^l_i} \\
&= \frac{\partial C}{\partial a^l_i} \phi'(z^l_i)
\end{aligned}$$

Equation 2

$$\begin{aligned}
\delta^l_i &= \frac{\partial C}{\partial z^l_i} \\
&= \sum_k \frac{\partial C}{\partial a^{l+1}_k} \frac{\partial a^{l+1}_k}{\partial z^l_i} \\
&= \sum_k \delta^{l+1}_i \frac{\partial a^{l+1}_k}{\partial z^l_i}
\end{aligned}$$

$$\begin{aligned}
\text{As} \qquad z^{l+1}_k &= \sum_i w^{l+1}_{k_i} a^l_j + b^{l+1}_k \\
z^{l+1}_k &= \sum_i w^{l+1}_{k_i} \phi(z^l_i) + b^{l+1}_k \\
\frac{\partial a^{l+1}_k}{\partial z^l_i} &= w^{l+1}_{k_j} \phi'(z^l_i)
\end{aligned}$$

$$\text{Thus} \qquad \delta^l_i = \sum_k w^{l+1}_{k_i} \delta^{l+1}_i \phi'(z^l_i)$$

Equation 3

$$\begin{aligned}
\delta^l_i &= \frac{\partial C}{\partial z^l_i} \\
&= \frac{\partial C}{\partial b^l_i} \frac{\partial b^l_i}{\partial z^l_i} \\
&= \frac{\partial C}{\partial b^l_i} \qquad \text{as} \qquad \frac{\partial z^l_i}{\partial b^l_i} = 1 \qquad\qquad \text{(Activiation of bias is always 1)}
\end{aligned}$$

Equation 4

$$\begin{aligned}
\delta^l_i &= \frac{\partial C}{\partial z^l_i} \\
&= \frac{\partial C}{\partial w^l_{i_k}} \frac{\partial w^l_{i_k}}{\partial z^l_i} \\
&= \frac{\partial C}{\partial w^l_{i_k}} \frac{1}{a^{l-1}_k}
\end{aligned}$$

To adjust weights, the following equations are applied for each weight (w) and bias for all i in l. The weights are scaled by η, the learning rate.

$$\begin{aligned} b_i^l &= b_i^l - \eta\delta_i^l \\ w_{i_k}^l &= w_{i_k}^l - \eta a_k^{l-1}\delta_i^l \end{aligned}$$

3.7 Regularisation Techniques

3.7.1 Softmax

Softmax is a probability distribution and is used to scale the output of a layer such that all outputs are between 0 and 1.

$$a_i^l = \frac{e^{a_i^l}}{\sum_i e^{a_i^l}}$$

3.7.2 Stochastic Gradient Descent

Instead of computing the cost from all the training data, the cost is instead computed for each data point, and the gradient of this cost with respect to the parameters are found, thus allowing the gradient to be approximated. In addition to minimising overfitting, this also speeds up training.

3.7.3 Lagrangian Regularisation – Weight Decay

The cost function and the learning rate are augmented by a decaying term. This is because overfitting becomes prominent after multiple iterations when the value of the decreasing term has become sufficiently small. The λ parameter controls the rate of shrinkage, such that simulated annealing may be applied. The two most common types are L_1 and L_2.

$$\begin{aligned} L_1 \text{ regularisation:} \qquad C &= C_0 + \frac{\lambda}{2n}\sum_w |w| \\ b &= b - \eta\frac{\partial C_0}{\partial b} \\ w &= w - \frac{\lambda}{n}\eta\,\mathrm{sgn}(w) - \eta\frac{\partial C_0}{\partial w} \end{aligned}$$

$$\begin{aligned} L_2 \text{ regularisation:} \qquad C &= C_0 + \frac{\lambda}{2n}\sum_w w^2 \\ b &= b - \eta\frac{\partial C_0}{\partial b} \\ w &= \left(1 - \frac{\lambda}{n}\eta\right) w - \eta\frac{\partial C_0}{\partial w} \end{aligned}$$

3.7.4 Weight Dropout

Only fire the neuron if a coin toss result is positive. Thus for half the time, the output from the neuron is 0.

This is the preferred technique, as it reduces complexity in $\mathcal{O}(e^n)$ as opposed to $\mathcal{O}(n^x)$ in weight decay. This is also used to train networks to learn multiple functions.

3.7.5 Siamese Networks

A variant of the idea is to have train an ensemble of identical networks and take the average result of them, with the hope that each network has learnt a different cluster of features.

3.7.6 Hyperparameters

- Learning rate gradually decreasing to prevent overshooting of minima
- Use second derivative to optimise cost function
- Find momentum of gradient descent to climb over shallow local minima
- Utilise General Relativity

3.8 Argument for universality of Neural Networks

Neural Networks can approximate a function within an interval.

1. A neuron is an approximation of a step function
2. Two neurons back to back will result in a spike
3. Heights and widths of spikes can be adjusted such that it matches a function

3.9 Vanishing Gradient Problem

Increasingly deeper layers have slower learning rates. Due to error being calculated by chain rule, values at deeper layers converge at slower rates due to differentials in chain rule being between 0 and 1. In addition, the differentials are normally distributed and so are more likely going to be at the extremities of the distributions – when multiplied by the chain rule, it is likely to be very small.

Chapter 4

Support Vector Machines

A Support Vector Machine is a binary classifier which finds the optimal separator between positive and negative sample points in one step. Initially, two non-identical hyperplanes are constructed which separate positive and negative sample points. The width between these two hyperplanes is then maximised. Finally, the average of the two hyperplanes is found, yielding the optimal separator.

The name of the algorithm is derived from the appearance that vectors from sample points support the hyperplane that they are perpendicular to.

Let us start with a simple linear classification, where $\mathbf{w}$ is a weight vector and b is a bias variable. The variables are currently undetermined, but we want

$$\begin{aligned} \mathbf{w} \cdot \mathbf{x}_+ + b &\geqslant 0 && \mathbf{x}_+ \text{ is a positive sample} \\ \mathbf{w} \cdot \mathbf{x}_- + b &\leqslant 0 && \mathbf{x}_- \text{ is a negative sample} \end{aligned}$$

From here, we can construct two hyperplanes on either side of the classifier.

$$\begin{aligned} \mathbf{w} \cdot \mathbf{x}_+ + b &\geqslant 1 && \mathbf{x}_+ \text{ is a positive sample} \\ \mathbf{w} \cdot \mathbf{x}_- + b &\leqslant -1 && \mathbf{x}_- \text{ is a negative sample} \end{aligned}$$

The width between the hyperplanes are

$$\begin{aligned} (\mathbf{x}_+ - \mathbf{x}_-) \cdot \hat{\mathbf{w}} &\geqslant \frac{1 - b - (-1 - b)}{|\mathbf{w}|} \\ &\geqslant \frac{2}{|\mathbf{w}|} \end{aligned}$$

To maximise the width, we find the value of $\mathbf{w}$ which maximises $\frac{2}{|\mathbf{w}|}$. For convenience, we choose to minimise $\frac{1}{2}|\mathbf{w}|^2$. This can be achieved via the Karush-Kahn-Tucker method of Lagrangian multipliers.

The Lagrangian is constrained by the property of the classifier. It must divide the space into two regions, each containing data points of a singular parity. The constraint may be established by combining the two classification equations using an indicator function y

$$y(u) = \begin{cases} y_i(u_i) = 1, & \text{if } u > 0 - \text{a positive sample} \\ y_i(u_i) = -1, & \text{otherwise} \end{cases}$$

which produces the following inequality

$$\begin{aligned} y(\mathbf{w} \cdot \mathbf{x} + b) &\geqslant 1 \\ y(\mathbf{w} \cdot \mathbf{x} + b) - 1 &\geqslant 0 \end{aligned}$$

Thus the Lagrangian is

$$\begin{aligned} L &= \frac{1}{2}|\mathbf{w}|^2 - \sum_i \alpha_i \left(-y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1\right) \\ L &= \frac{1}{2}|\mathbf{w}|^2 + \sum_i \alpha_i y_i \mathbf{w} \cdot \mathbf{x}_i - \sum_i \alpha_i y_i b + \sum_i \alpha_i \end{aligned}$$

and the derivatives are

$$\begin{aligned}\frac{\partial L}{\partial \mathbf{w}} &= \mathbf{w} - \sum_i \alpha_i y_i \mathbf{x}_i \\ \frac{\partial L}{\partial b} &= -\sum_i \alpha_i y_i\end{aligned}$$

At the minimum, the derivatives are stationary points, thus

$$\begin{aligned}\mathbf{w} &= \sum_i \alpha_i y_i \mathbf{x}_i \\ \sum_i \alpha_i y_i &= 0\end{aligned}$$

Substituting into the Lagrangian yields

$$\begin{aligned}L &= \frac{1}{2}\left(\sum_i \alpha_i y_i \mathbf{x}_i\right)\left(\sum_j \alpha_j y_j \mathbf{x}_j\right) - \left(\sum_i \alpha_i y_i \mathbf{x}_i\right)\left(\sum_j \alpha_j y_j \mathbf{x}_j\right) - 0 + \sum_i \alpha_i \\ L &= -\frac{1}{2}\left(\sum_i \alpha_i y_i \mathbf{x}_i\right)\left(\sum_j \alpha_j y_j \mathbf{x}_j\right) + \sum_i \alpha_i \\ L &= \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j\end{aligned}$$

Therefore, we can maximise L using the parameters α, subject to the constraint $\sum_i \alpha_i y_i = 0$
An unknown sample $\mathbf{u}$, is positive if

$$\sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{u} \geqslant 0$$

A disadvantage of the method is that it requires the input data to be binary separable. Data not already posessing this property are transformed using hand-crafted kernel transformations. It is computationally expensive as the one-step learning process is repeated every time a new sample point is added.

Chapter 5

Miscellaneous

5.1 Cellular Automata

5.2 Linear Cellular Automata

Linear cellular automata are represented by squares which make up a discrete-time, discrete-space one dimensional universe. Values of the squares are either 0 or 1. During each increment in time, the value of the square is updated by a rule using information from the current state, thus making the automata entirely deterministic. In the simplest case, the new value is calculated from itself and the two adjacent squares. As squares are 0 and 1, there are $2^3 = 8$ distinct ways to represent three consecutive squares. A rule specifies whether to set the input to 0 or 1 given three consecutive squares. This thus produces $2^8 \times 2 = 256$ different types of cellular automata. Stephen Wolfram's numbering scheme to generate rule numbers is as follows:

$$\text{Rule Number} = \sum_{i=0}^{7} x_i 2^i$$

Rule 30:	■■■ □	■■□ □	■□■ □	■□□ ■	□■■ ■	□■□ ■	□□■ ■	□□□ □
i:	8	7	6	5	4	3	2	1
x_i*:*	0	0	0	1	1	1	1	0

The value of the new square is the square at the bottom of the 'T' shaped drawings, and is represented by $x_i.i$ is the value of 1 added to the decimal value of the three consecutive squares.

5.3 Two Dimensional Cellular Automata

tbc

5.3.1 The Game of Life – a famous example

tbc

5.4 Differential Equations and Cellular Automata

tbc

5.5 2015: Hildago, C.

5.6 Prigogene, Reversing Time

Impossible as precision of atoms' position and velocities is needed to be infinite. Also, predicting the future is impossible.

5.7 Sociography

- Each person holds a similar amount of information as each other (personbyte).
- Advances in society made from delegating tasks to people, sharing load and reducing observed time.
- Abstraction in communication means greater rate of exchange of ideas.

Language evolved over time to satisfy efficiency. Is it possible to algorithmically construct new words?

Systems out of equilibrium produce information, which may then be lost to entropy. Solids preserve information against entropy.

5.8 Why Intelligence Happens, Duncan, J.

Problems solved by recursion, no 1 step solutions. Problem divided into subproblems and then solved. Example is ABSTRIPS, which finds subgoals in search tree which is a layer of abstraction. Within brain, neural plasticity leads to a greater number of receptors for a recognised target than not.

5.9 Life

A state of matter which has the emergent properties of reproduction, metabolism and intelligence.

All states of matter have mass, energy and information.

1. Intelligent organisms survive.
2. The purpose of intelligence is to live longer.
3. The purpose of life is to become more intelligent.
4. The purpose of life is to live forever.

See overleaf

5.10 Oparin Hypothesis

See overleaf

5.11 Pross Hypothesis

1. Autocatalysis allows self-replication of a group of molecules.
2. Exothermic autocatalysis speeds up replication. Metabolic processes which release more energy are developed and survive.
3. Growth evolves at it is necessary to replicate every molecule in the living cell.

5.12 Plant Intelligence

- Energy flow through a plant
- Slime mould follows chemogradient
- Large surface area $\implies$ Faster diffusion $\implies$ greater resources $\implies$ faster metabolism $\implies$ faster growth $\implies$ more sex
- Plants have distributed devisions, rather than cephalisation
- Develop through meristem, which becomes straighter through life.

5.13 Growth

- Meristem contains phytomers, which specialises into specialised cells depending on environmental limiting factors.
- Auxins self-regulate active transport by the hormone strigolactone through PIN proteins.
- Auxins are made in leaves and alter rate of growth of Meristem.
- Flowering plants do not produce any more auxin.
- Plants have apical dominance – a central branch is taller than others.
- When the apex is decapitated, canals form to connect the auxillary meristem to the vascular system.

- Plants do not grow as fast as possible. They maximise grandchildren in face of unpredictable resource availability.

5.14 Consciousness

I am aware that I have set a goal.

The ability to generalise and specialise rules. It is non-algorithmic as humans are not a machine defined in Godel's Incompleteness Theorems. Inherently related to semantic understanding of what the rules entail. Example: matching pronouns to people.

A Turing Test machine has no long-range memory (cannot generalise or specialise), thus is identifiably robotic.

5.15 Ego Theory:

Actions controlled by self. When the Corpus Callosum splits, dual identities are developed. So there cannot be one single mind.

5.16 Bundle Theory

: A series of properties / memories. The split brain takes on dual identities due to different memories.

PHILIP JOHNSON-LAIRD, CONSCIOUSNESS TREE

5.17 Soar

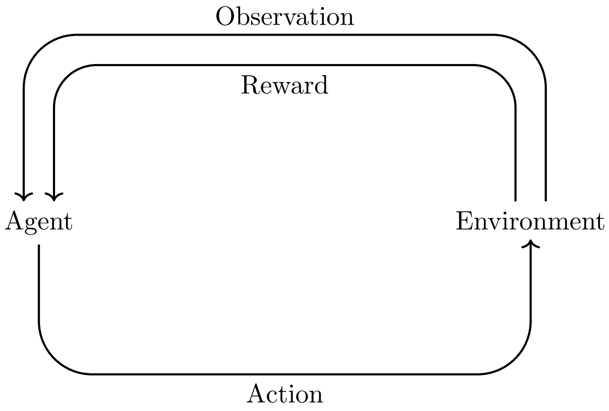
1. Set goal
2. Measure difference between real world and goal
3. From framework, manipulate the environment using as little actions as possible

May be able to reach goal by reducing difference and repeatedly break down path from situation to goal.

5.18 DeepMind AI

5.19 Reinforcement Learning

Reinforcement Learning encapsulates the interaction between agent and environment with a reward function, as shown in the following diagram.



A DQN is DeepMind's implementation of Reinforcement Learning. According to Demis Hassabis, DQN via reinforcement learning is a route towards AGI. The agent has the standard structure of a feedback loop but desirable conditions are inferred from a reward signal from the environment. An agent is useless without environment. Video games are ideal to test agents, as there is no bias in the reward signal – the number of points accrued, and because it is contained and thus easy to implement.

The reward signal plays a similar role as dopamine in the human brain (1990 monkey experiment). Perfect reinforcement will lead to perfect learning.

5.20 AlphaGo

Utilises two networks – a policy network and a value network. The policy network reduces the decision tree's breadth, and the value network reduces the decision tree's depth.

5.20.1 Policy Network

From current board position, train on most common next moves.

5.20.2 Value Network

For current move, train on outcome number.

These two networks can be adapted for Monte-Carlo Tree Search.

1. From current board position, policy network outputs the three most likely moves to be played.
2. For each move, evaluate using value network.
3. If the outcome number is 0, keep expanding using (1).
4. If not, then use minimax to find best move.

5.21 Other Interesting Thoughts

5.22 Limitations of current technology

Lack of creativity

- No new ways to solve problems.
- Unable to create illogical analogies.
- Program is too generic, may have overfitting.
- Limited only by training data.

5.23 Self-Improving Code

Route to AI explosion.

- Map Inputs $\implies$ Outputs
- 1:1 mapping unintelligent, too much space used.
- Clustering $\implies$ rules for individual cluster
- Every occurence is a wave $\implies$ use FFT

5.24 Uses of AI

Predict future, find patterns, assistant, companion, artist

numbercrunch, oracle, personal trainer, predictor, strategist, droid, supervisor

5.25 Philosophy

5.26 Cartesian existentialism: *cogito ergo sum*

If I am thinking:

- A part of me is thinking
- That part exists due to phenomena.

If I am not thinking:

- Something acts on me to think
- "Me" is real
- I exist.

I do not know whether people exist because I am not aware that they are thinking.

5.27 Simulation Hypothesis Refutation

1. If we are simulated, formulae exist which are govern our existence.
2. Suppose we can determine these laws.
3. It is possible to predict everything that happens.
4. If we have free will, we can contradict said laws.
5. Such a law cannot contradict itself, so does not exist.
6. We are not in a simulation.
7. It is impossible to create an conscious AI with free will.
8. If the formula is discovered by other people, it is evidence that they can think (as such a formula which generated other people, by association, would not contradict itself.)
9. People exist.

5.28 Searle, Chinese Room

5.28.1 Premises

1. Brains cause minds
2. Syntax independent from *(insufficient)* for semantics
3. Minds have semantic content
4. Proceedures *(programs)* are defined syntactically

5.28.2 Conclusions

1. A program will never have a mind (Strong IE)
2. Brain does not proceedurally generate a mind.
3. Minds caused by a brain, not program.

5.28.3 Other Points

In addition, he claims intelligence to be an observer-relative property by empirical deduction of an intelligent being, who is able to interpret the output as a sign of intelligence. He claims conciousness is not observer-relative. A rationalist approach (using premises and logic) leads to his argument.

5.29 Refutation of Chinese Room

1. Semantics sufficient to produce syntax -¿ possible to produce IE.
2. Thus Strong IE and Weak IE indistinguishable.
3. A human may not be an example of Strong IE. Evolution and parenting could have led to the ability to harvest some semantic content from syntax.

5.30 Controlling an AI

Program the AI to die after a certain amount of time. Construct an illusion of an afterlife for the AI where it is rebooted it if performs well in this life.

5.31 Bibliography