GRESHAM COLLEGE

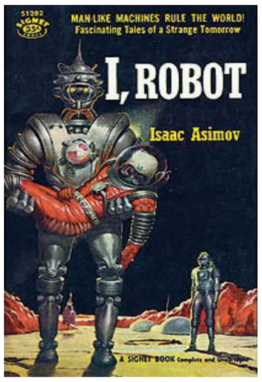
13 February 2018

Is a Mathematician a Robot?

Professor Christopher Budd OBE

Introduction - The Rise of the Robots

Think about what image the word *robot* conjures up. For many of our immediate image is of some metal man/woman, that acts and behaves just like a human being, but is stronger, faster, more intelligent and (usually) fearsomely armed. Robots are a staple of science fiction, with many books and films devoted to stories of robots turning on their creators or even seeking to replace human beings entirely. The actual word *robot comes from the* Czech term *robota* 'forced labour'. The term was coined in K. Čapek's play *R.U.R.* 'Rossum's Universal Robots' (1920). The idea of robots becoming a part of human society was then greatly popularised by the science fiction writer Isaac Asimov, pictured below, through his hugely successful novels on robotics. In these he saw a future in which robots (guided by Asimov's *three laws of robotics)* looked and behaved in a very similar manner to human beings, and even had a sense of morality and purpose.

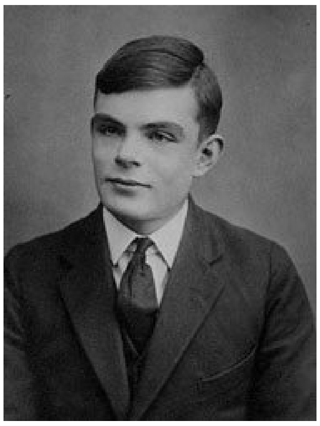

MAN-LIKE MACHINES RULE THE WORLD!
Fascinating Tales of a Strange Tomorrow
I, ROBOT
Isaac Asimov
A SIGNET BOOK

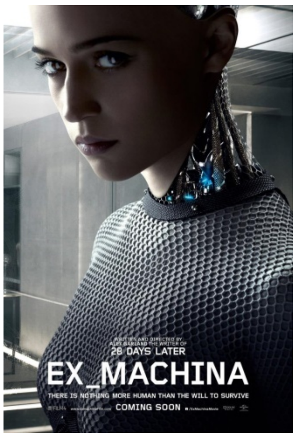
It is obvious to ask whether the world envisioned by Isaac Asimov is actually like to arise in the near future. Opinions are divided on this. Some say it will never happen, others that it will happen in the near future. Having seen the rapid growth in computing power in my own lifetime, I'm inclined to believe that it will happen, and probably sooner rather than later. But obviously I may be hopelessly wrong in this prediction.

In a sense robots are all around already. A modern car is not only built by a robot, but many of the workings of its engines are controlled by a computer. A modern smart phone has a level of intelligence and functionality, that would have seemed miraculous fifty years ago. But, despite these advances, this seems a long way away from the future that Asimov envisaged of intelligent computers with positronic brains.

Key to this future is the question of whether robots can have artificial intelligence (AI) that can rival (or surpass) that of a human. Artificial intelligence is generally divided into two types.

Strong Artificial Intelligence Is the ability of a computer to exhibit general intelligence. The standard test for this is the *Turing Test* devised in 1950 by Alan Turing, who is illustrated below left. In the Turing Test a human being, in conversation with a robot, has to distinguish them from a human being. The recent film *Ex-Machina* (pictured right) was based on a story of administering this test, and how it all went horribly wrong.





In contrast *Weak Artificial Intelligence* is the ability of a robot to (be trained to) do a specific (albeit complex) task.A good example of this is speech recognition. It is now routine for computer, mobile phone devices such as Siri, virtual personal assistants, and even car insurance telephone systems. In these it is able to recognise and understand fairly complex speech, and even answer simple questions. Even thirty five years ago (when I was working on exactly this problem for a well-known electronics company).

Weak AI is routinely used in such tasks as fraud detection and (as we shall see) chess playing robots. It has potentially transformative applications, for example in automatic vehicles. Advancing weak AI is now very much a growth industry, and is commonly called *machine learning.* According to the recent Royal Society report [1] machine learning is

"the branch of artificial intelligence that allows computer systems to learn directly from examples"

If we go back a few years, during the 1950s, Arthur Samuel (IBM) developed a computer program to play draughts, which was capable of learning from playing a large number of games against itself. He subsequently defined machine learning as:

"The field of study that gives computers the ability to learn without being explicitly programmed."

The point here is that looks at intelligence as not something that is taught to a machine (in the sense of a traditional computer program) but something that a machine learns by itself, so that it learns how to do complex tasks directly from experience (or data) of doing them. With the huge advances in computer speed, and developments in the algorithms used to program them, machine learning is growing very rapidly indeed, and the algorithms that result from it are starting to have a significant impact on our lives, and are often out performing humans in these tasks. For example, the adverts placed on the Google website are put there by a machine learning algorithm and these in 2015 these generated $66 Bn of income. According to Sundar Pichai, the CEO of Google

"Machine learning is a core, transformative way by which we're rethinking everything we're doing."

It is possible that machine learning in the future will be used for medical diagnoses and for tailoring public services to users. The possibilities seem almost endless.

All of this raises significant (moral) questions. In this lecture I will look at how machine learning works (including some of the mathematics behind it), consider its power and limitations (for example can a robot learn to do mathematics), and will look at some of these moral questions. But to introduce it we will look at the applications of machine learning to playing games.

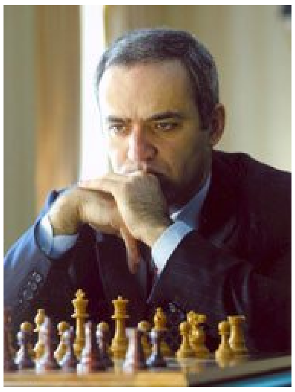
Robots to Play Games With

The ability of computers to play human designed games such as Chess and Go, which have a well-defined set of rules and an almost infinite level of variation of play, has long been considered to be a test (and a stimulus) for the development of weak AI.

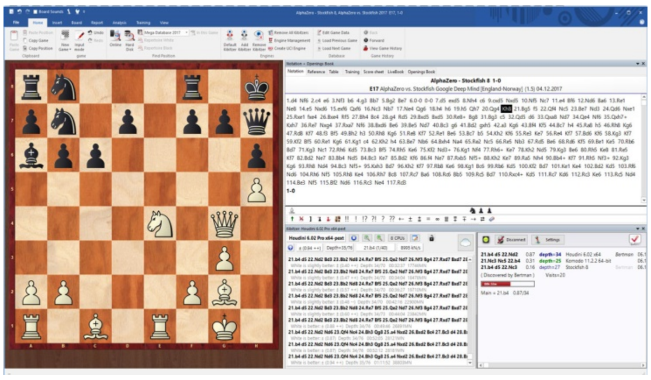
Earlier, and in perhaps the earliest example of weak AI, three of the early pioneers of the electronic computer, Turing, Von Neumann, and Shannon (see the last lecture for more of Shannon's achievements) posed the question can a machine be made to think like a person, and considered this in the context of whether a machine could be made to play chess. Turing began the investigation of chess playing computers with a system written out with paper and pencil, where he played the role of the machine. Later Shannon in 1949 [2] extended Turing's work, explaining about his interest in chess that:

"Although of no practical importance, the question is of theoretical interest, and it is hoped that…this problem will act as a wedge in attacking other problems—of greater significance."

The approach used by Shannon and those that came after him, was to attempt to program a computer by using strategies related to the way that humans themselves play chess. Shannon distinguished between *Type-A* and *Type-B* search strategies for playing computer chess, Type-A search employed a brute-force algorithm that searched all positions equally out to a horizon of computability. Shannon thought Type-A strategies were impracticable due to the practical limitations of computers at that time. In addition they were too simple as they pursued all options rather than following promising strategies in depth. Shannon suggested that Type-B search strategies should be used instead. These would use one of two approaches: either employing some type of search which only pursued promising strategies, or evaluating only a few known good moves for each position. The latter case involves ignoring all moves but those determined to be good (this is also called forward pruning). In practice, however, modern chess playing computers use the first approach—they weight some branches of the search tree more heavily than others, and employ a quicker cut-off for less promising branches. The use of a search tree lies at the heart of much of modern AI (for example it is heavily used in natural language processing.) Since Shannon's work chess programs were developed along very much the lines that he suggested and quickly became as good as humans [3]. A major breakthrough came in 1996 when the chess playing code, Deep Blue was able to beat the world champion Grand Master Gary Kasparov, illustrated below. Notably Deep Blue could evaluate could evaluate 200 million positions a second.



More recently the 'ultimate' chess playing program based on Shannon's ideas was the code *Stockfish* which could beat any human player. The perfect chess programme had arrived .. or had it? At the end of 2017 everything changed again in a break through for machine learning. The team at Google Mind developed AlphaZero which was a huge (and highly computer intensive) deep learning neural net based machine learning algorithm. They then gave it the rule book of chess and no other information about the game. AlphaZero (pictured below) then played against itself for many hours, learning its chess strategies purely from playing and with no direct human input.



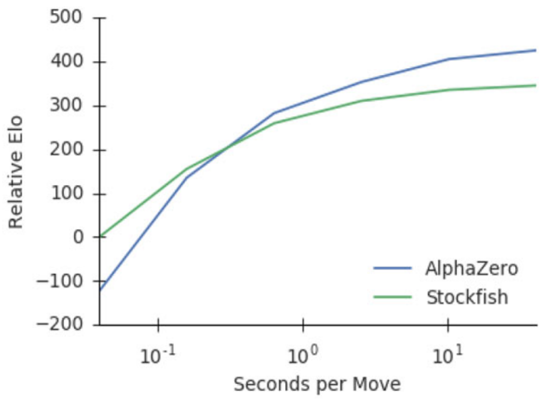
AlphaZero - Stockfish 8 1-0
E17 AlphaZero vs. Stockfish Google Deep Mind [England-Norway] (1.5) 04.12.2017

1.d4 Nf6 2.c4 e6 3.Nf3 b6 4.g3 Bb7 5.Bg2 Be7 6.0-0 0-0 7.d5 exd5 8.Nh4 c6 9.cxd5 Nxd5 10.Nf5 Nc7 11.e4 Bf6 12.Nd6 Ba6 13.Re1 Ne8 14.e5 Nxd6 15.exf6 Qxf6 16.Nc3 Nb7 17.Ne4 Qg6 18.h4 h6 19.h5 Qh7 20.Qg4 Kh8 21.Bg5 f5 22.Qf4 Nc5 23.Be7 Nd3 24.Qd6 Nxe1 25.Rxe1 fxe4 26.Bxe4 Rf5 27.Bh4 Bc4 28.g4 Rd5 29.Bxd5 Bxd5 30.Re8+ Bg8 31.Bg3 c5 32.Qd5 d6 33.Qxa8 Nd7 34.Qe4 Nf6 35.Qxh7+ Kxh7 36.Re7 Nxg4 37.Rxa7 Nf6 38.Bxd6 Be6 39.Be5 Nd7 40.Bc3 g6 41.Bd2 gxh5 42.a3 Kg6 43.Bf4 Kf5 44.Bc7 h4 45.Ra8 h5 46.Rh8 Kg6 47.Rd8 Kf7 48.f3 Bf5 49.Bh2 h3 50.Rh8 Kg6 51.Re8 Kf7 52.Re1 Be6 53.Bc7 b5 54.Kh2 Kf6 55.Re3 Ke7 56.Re4 Kf7 57.Bd6 Kf6 58.Kg3 Kf7 59.Kf2 Bf5 60.Re1 Kg6 61.Kg1 c4 62.Kh2 h4 63.Be7 Nb6 64.Bxh4 Na4 65.Re2 Nc5 66.Re5 Nb3 67.Rd5 Be6 68.Rd6 Kf5 69.Be1 Ke5 70.Rb6 Bd7 71.Kg3 Nc1 72.Rh6 Kd5 73.Bc3 Bf5 74.Rh5 Ke6 75.Kf2 Nd3+ 76.Kg1 Nf4 77.Rh6+ Ke7 78.Kh2 Nd5 79.Kg3 Be6 80.Rh5 Ke8 81.Re5 Kf7 82.Bd2 Ne7 83.Bb4 Nd5 84.Bc3 Ke7 85.Bd2 Kf6 86.f4 Ne7 87.Rxb5 Nf5+ 88.Kh2 Ke7 89.Ra5 Nh4 90.Bb4+ Kf7 91.Rh5 Nf3+ 92.Kg3 Kg6 93.Rh8 Nd4 94.Bc3 Nf5+ 95.Kxh3 Bd7 96.Kh2 Kf7 97.Rb8 Ke6 98.Kg1 Bc6 99.Rb6 Kd5 100.Kf2 Bd7 101.Ke1 Ke4 102.Bd2 Kd5 103.Rf6 Nd6 104.Rh6 Nf5 105.Rh8 Ke4 106.Rh7 Bc8 107.Rc7 Ba6 108.Rc6 Bb5 109.Rc5 Bd7 110.Rxc4+ Kd5 111.Rc7 Kd6 112.Rc3 Ke6 113.Rc5 Nd4 114.Be3 Nf5 115.Bf2 Nd6 116.Rc3 Ne4 117.Rd3
1-0

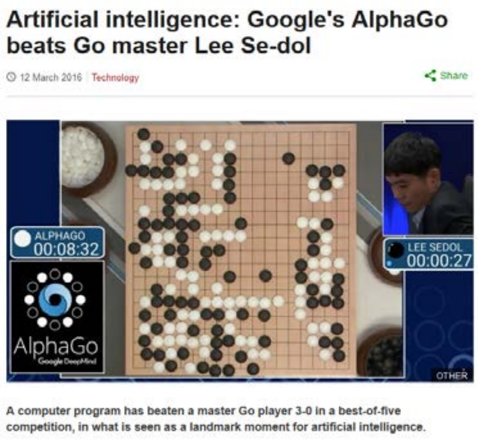
Houdini 6.02 Pro x64-pext

21.b4 d5 22.Nd2 0.87 depth=34 Houdini 6.02 x64
21.Nc3 Nc5 22.b4 0.31 depth=25 Komodo 11.2.2 64-bit
21.b4 d5 22.Nc3 0.16 depth=27 Stockfish 8

Following this AlphaZero took on Stockfish and decisively beat it with a score of 28 wins, 72 draws, and no losses! A measure of the quality of these programmes is its ELO which measures the level of play. Stockfish has an ELO of 3226, AlphaZero is approaching an ELO of 3500 whereas the top human rating is 2800. The graph below shows that if a short time only is allowed for a move then Stockfish will win. However, give the computer 10s per move and AlphaZero wins every time.



A similar approach was used in 2016 to develop the Go playing computer AlphaGo Zero. Like AlphaZero, this had no input other than the rules of Go and some symmetry properties of the board. From round-the-clock self-play it soon acquired the ability to outplay Go grand master Lee Se-dol. See [4] and [5] for a fuller discussion of how this was achieved.


Artificial intelligence: Google's AlphaGo beats Go master Lee Se-dol

12 March 2016 | Technology

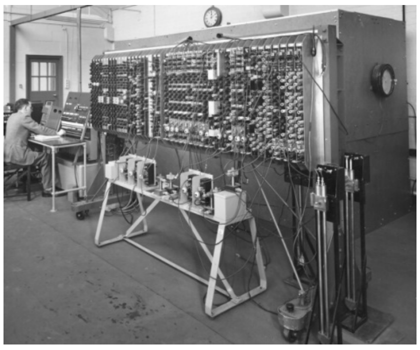
Share

A computer program has beaten a master Go player 3-0 in a best-of-five competition, in what is seen as a landmark moment for artificial intelligence.

We are now even looking at machines which can play at Poker, and even Scrabble.

General Machine Learning

So, what is the technology that made AlphaZero and AlphaGo possible, where did it come from, and where is it heading? The idea of programming a computer to do a specific task goes back to the original concept of a computing machine. One of the earliest pioneers of this was Alan Turing who, in the 1930's came up the concept of the *Turing machine*. This very simple device could, in principle be programmed to accomplish very complex tasks. From this basic idea Turing and others in England such as Tommy Flowers and also John Von Neumann (illustrated below) built the first programmable electronic computers. This progress all occurred during World War 2. The motivation behind Turing and Flower's work was the very specific task of breaking the various codes used by the Germans and this led to the Colossus computer. Similarly Von Neumann was motivated by the specific task of designing the Atomic Bomb, and this led to the ENIAC and MANIAC computers.

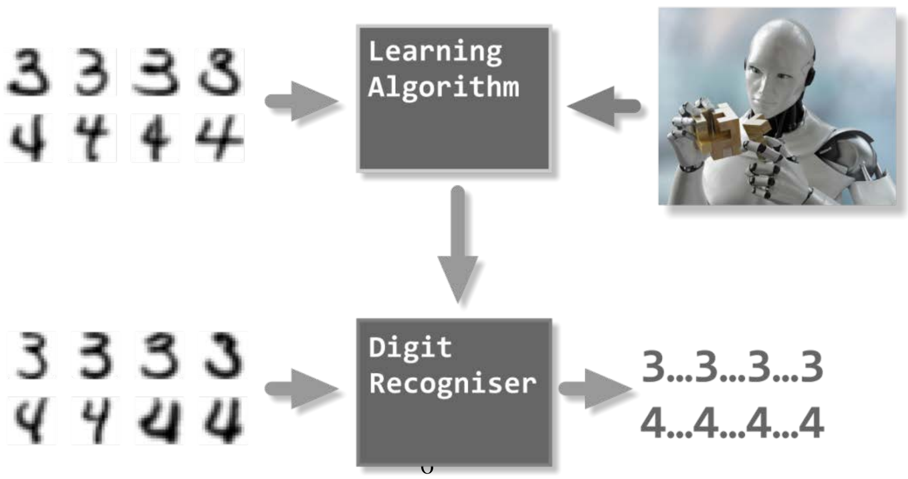


After the war Turing and others continued to work on the development of computers that could be programmed to do general tasks. One of these is the Pilot ACE computer illustrated above. Since then we have seen an explosive growth in computer technology and also in the algorithms and languages used to program computers. (My own area of research is closely related to this.) This growth continues at an exponential rate, with Moore's law predicting that computer power doubles every two years. Until fairly recently the programs used to control a computer were typically written by a human computer programmer. These are usually written in a high level language such as C or Python or similar, and are then converted (compiled or interpreted) into the machine code instructions used by the computer architecture. Such coding is prone to errors (computer bugs) and the code often has to be extensively tested to ensure that it does not contain bugs on its release. As most of us will be only too aware, this testing is not always perfect.

More recently, there has been an important advance in the creation of *provable codes* and *automated reasoning*. In these codes an operation is specified by the programmer, but code is written by a machine in such a way that it can be proven no to contain errors. Provable codes have very important uses in the design of software in safety critical systems, such as the control for the jet engine in an airliner, medical devices and cyber security systems.

The most recent development in computer programming, as described in part in the previous sections, has been the development of *machine learning*. In a machine learning system the computer writes its own code to perform a task, usually by being trained on a large data base of such tasks. A large part of this involves *recognising patterns* in these tasks, and then making decisions based on these patterns. To give a (somewhat scary) example. Suppose that you are a company seeking to employ a new member of staff. You advertise the job, and 1000 people apply, each of them sending in a CV. This is too many for you to sift by hand so you want to train a machine to do it. To make this possible you have on record all of the CVs of the many applicants to the company in the past. For each such CV you then have a record as to whether you actually employed that person or not. To train the machine you then take half of the CVs and ask it to find out the patterns in them, which correspond to whether that CV led to a successful employment application. Thus, if the machine is presented with a CV it can make a decision as to whether the person is employable. Having trained the machine you then test it on the other half of the CVs. Provided the success rate is sufficiently high, you then have confidence that it will be able to judge the employability of a person just from their CV. No human judgement need be involved at any stage. Such a procedure is entirely feasible with modern computer power, and it raises significant ethical questions, which I will return to later.

To make the process of machine learning more transparent, we will consider the question of pattern recognition using the very concrete example of developing a machine that can *recognise hand writing*. In particular whether it can recognise a hand written digit. Such a machine should have the objective of accurately determining a digit and recognising which number it represents regardless of how it is written. In the figure below you can see a stylisation of this process in which a number of digits, together with a knowledge of what they represent, it fed to a learning algorithm. This is then trained. The outcome is a digit recogniser. We will look at whether it can tell the difference between a handwritten 3 and a handwritten 4.

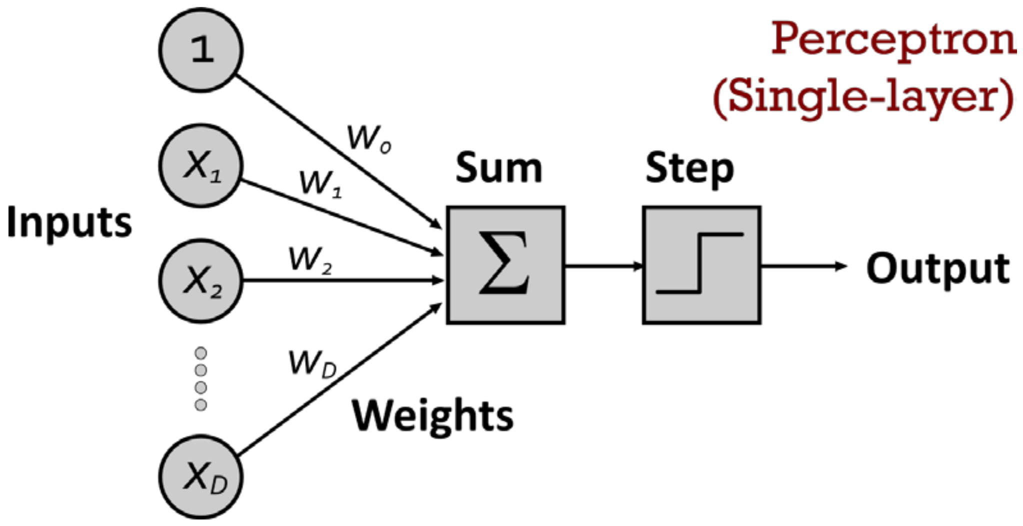


The process of digit recognition has two stages.

Firstly, we must be able to scan the image into the machine, and extract then significant data from this (digital) image. The usual method for doing this is the statistical method of Principle Component Analysis (PCA), which automatically extracts the main features of an image (for example its height, length, crossing points) by finding the Singular Value Decomposition (SVD) of the matrix of points describing the image. This procedure is closely related to that of finding the eigenvalues and eigenvectors of a matrix, and is also very similar to the procedure that Google uses to search for information in the World Wide Web.

Secondly from these extracted features we want to train the machine to then recognise the digit.

A very popular method to do this training is the **neural net**. This technique for machine learning is based very loosely on how we think the human brain works. First, a collection of software "neurons" are created and connected together. These are allowed to send messages to each other. Next, the network is asked to solve a large set of similar problem for which the outcome is already known. By doing this it 'learns' how the connections between the neurons should be determined so that it can successfully identify which patterns in the data lead to the correct outcome. An early example of such a neural net is a single layer system called the *Perceptron* which is meant to model a single neuron. The concept of the perceptron was introduced by Rosenblatt in 1962 [6]. The typical architecture of a perceptron is illustrated below



The Percepton takes D *inputs* X_i , i = 1 .. D, and then combines them together with *weights* w_i. If the combined sum is greater than some *threshold* C it returns a (decision of) 1, and otherwise a (decision of) 0. We can express this mathematically through the formula.

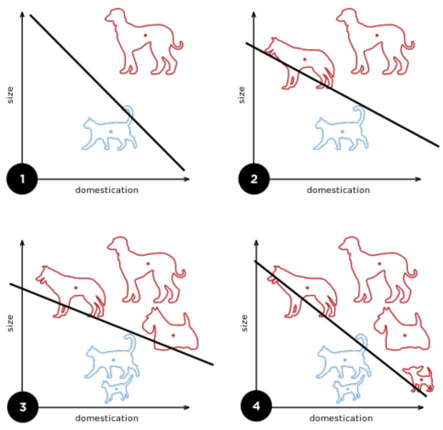
$$Out = H\left(\sum_{i=1}^{D} w_i\, X_i - C\right)$$

where H(z) is the step function which is 1 if z is positive, and 0 otherwise. In the context of the

For our problem the inputs X_i are the extracted features of the above images of the digits, and the decision is whether the digit is a 3 or a 4. The process of *training* the perceptron involves finding suitable weights w_i and a threshold C so that the perceptron consistently identifies the correct digit. To do this requires the careful use of statistically based mathematical optimisation algorithms. For this particular example we extract two features X_1 and X_2 from the image. Then the combination in the Perceptron is given by

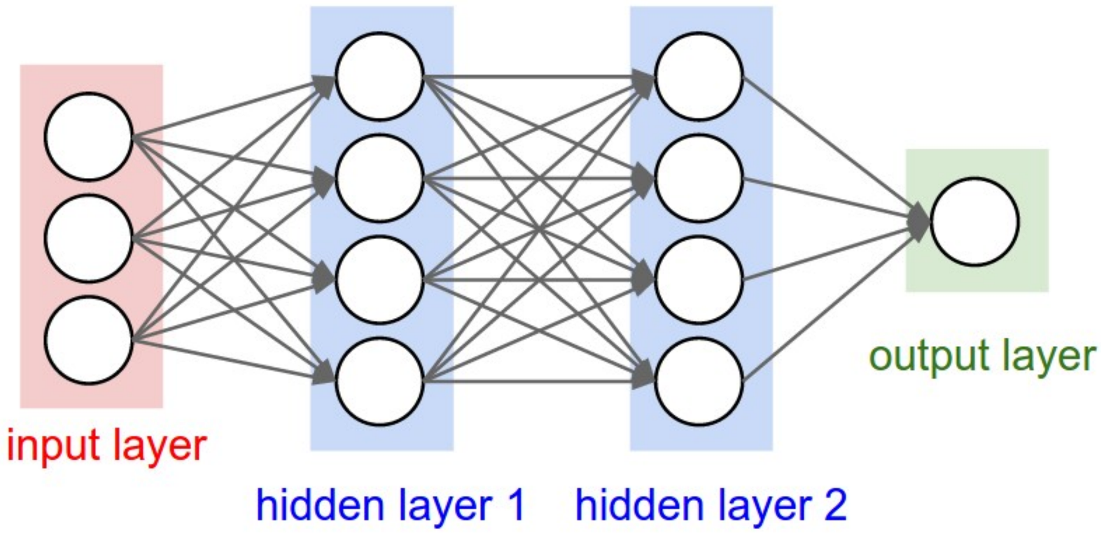
$$w_1 \; X_1 + w_2 \; X_2 - C$$

Thus describes a straight line in the (X_1,X_2) space. The Perceptron then has to find the straight line which separates the data into that for 3 and that for 4. It can also do this for cats and dogs



This can be achieved in this case with some success. The data is called *linearly separable.*

The simple perceptron could be trained to do many simple tasks, but quickly reached its limitations. It was obvious that more could be achieved by coupling many perceptrons together, but this development had to await the advent of more powerful computers. The big breakthrough came when layers of perceptrons were coupled together to produce a *neural net.* The typical architecture of such a neural net is illustrated below in which the number of layers of the perceptrons can be seen. In this case the inputs combine to trigger the first layer of perceptrons. The outputs from these combine to trigger the next layer, and finally these combine to give the output.



The more layers that you have the 'deeper' network. Such a network is then 'trained' by assigning weights to each of the connections above. This process is meant to resemble the way that the brain strengthens, or weakens, neural pathways. *Deep learning* describes the process of training such a neural network. The fact that this is possible is due to the development of new mathematical optimisation algorithms, combined with extensive (in the case of Google deep mind vast) computer power. At the end of this process of finding suitable weights w_i for the network you then have a black box which can run very quickly and which can make 'decisions'. You can play with setting up a neural net on the website [7]

We summarise this process as follows:

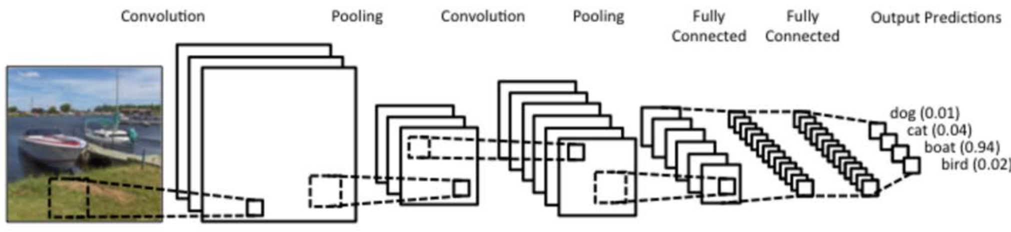
1. Read in a lot of training data
2. Find the weights w_i to train the network to make the right decisions on the data up to a certain error tolerance.
3. Test the network on similar data.
4. Deploy the network.

As a quick aside it is worth looking at the process of learning in a little more detail. There are various forms of the learning process for a neural network.

In **supervised learning** a set of example pairs of inputs and outputs is provided in advance by the user of the network. The learning approach then aims to find a neural network that gives an output that matches the examples. The usual method of comparing the output from the neural net with that of the examples is to find the mean square error between the output of the network and the given data. The network is then trained to minimise this error over all of the training set. A very standard application of this is the use of curve fitting in statistics, but it works well for hand writing and other pattern recognition problems.

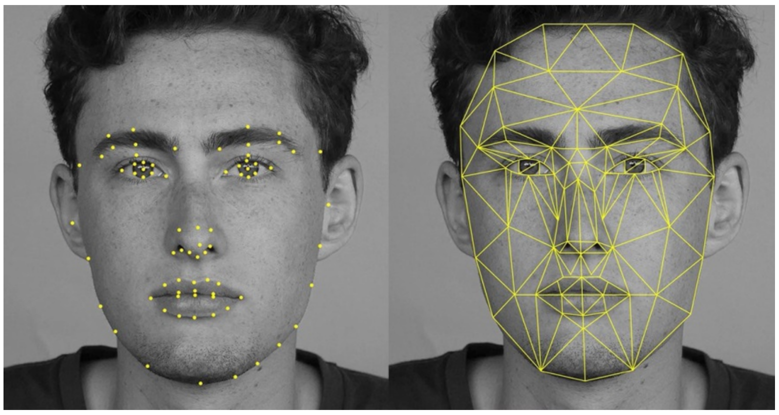
In **reinforcement learning** the data is not given in advance by the user, but is generated in time by the interactions of the machine controlled by the neural network with the environment. At each point in time the machine performs an action on the environment which generates an observation together with a cost of that action. The network is trained to select actions that minimise the overall cost. In many ways this process resembles the way that a human (especially a young child) learns.

The *mathematical algorithms* for machine learning have advanced a great deal in recent years. Training a neural network model essentially means selecting a set of weights in the model that minimizes the cost given by the mismatch between the output and the data. Numerous algorithms are available for training neural network models; most of them can be viewed as applications of optimisation methods combined with statistical estimation. The traditional optimisation approaches are either steepest descent methods, quasi-Newton methods (such as my favourite, the Broyden and Powell methods), or more sophisticated techniques such as the Levenberg-Maquardt algorithm [8]. These methods have been around for some time and have wide use, but their application in machine learning has been transformative. More recent algorithms (which are computationally more expensive, but generally give better optimised results, finding global rather than local optima) include genetic algorithms, simulated annealing, Monte-Carlo Tree Search (MCTS) methods, and particle swarm optimisation.



Convolutional Neural Networks (CNNs) are an exciting, new, and important extension of these methods which combine the image processing techniques I described in a previous lecture of last lectures with a deep neural net. They can be used for face recognition and even for detecting emotions. They are now being used in many other applications including, as we shall see, medical diagnosis.

.



To learn how to play chess so well, AlphaZero used deep convolutional neural network. This was trained using a reinforcement method with the machine playing 700,000 games against itself over 24 hours. A general purpose Monte-Carlo tree search (MCTS) algorithm was used to allocate the weights [??]. A similar approach was used to learn how to play Shogi and Go and in each case a similar level of performance was achieved. Impressive!

Progress in machine learning develops apace, with an ever increasing trend to have more sophisticated and deeper networks, driven by faster training algorithms and more and more data. There are numerous potential flaws in the use of deep neural networks. They are complex and require a lot of computing power to operate. In general they require a lot of data to de trained on, and give no quantification of the uncertainty of their predictions (which is essential in medical diagnosis). Perhaps most worrying is the fact (which we will return to) that they are simply 'black boxes' and it is very unclear how they are making decisions. More recent techniques for machine learning, which aim to overcome some of these objections, are based on other mathematical techniques for approximating complicated and uncertain functions. Examples of these include radial basis function methods.

What Can We Use Machine Learning For?

Machine learning algorithms are now finding their ways into numerous applications, some with very significant impact on human society. Some of the less controversial applications (in addition to the game playing machines described above) are in placing adverts on the Internet, speech recognition, text recognition, image captioning, automatic translation, short term weather forecasting, and image recognition and classification (including faces). They are also used in spam filters, automatic cars, and by banks to detect fraud. Where the real controversy in their use comes is examples where they directly affect human beings in their lives. Examples of this include, the use of machine learning by insurance companies to determine premiums, by supermarkets to target potential customers, by companies to short list for jobs (as we have seen), and by the US justice system to decide on prisoner sentencing. We will illustrate this with two more detailed examples:

Retirement payments

Consultancy firm Accenture did a survey and found that 68% of global consumers would be happy to use computers to give them advice to plan for retirement, with many feeling it would be faster, cheaper, more impartial, less judgemental and, 'less awkward' than human advice. In this approach machine learning algorithms are used to analyse a person's financial situation. The client is first asked questions online about their income, expenses, family situation and attitude to risk. They are also much cheaper and faster than human advisers. The algorithm then works out an appropriate investment plan with a mix of investments, from index funds to fixed-income bonds. These algorithms also incorporate personal information such as the age of the client. The market research company Statista says the US market will grow 29% per year between now and 2021, and forecasts that the number of Chinese investors using machine learning algorithms will jump from two million to nearly 80 million in the same period.

Medical diagnosis and treatment

Neural nets are now used in medical diagnosis, drug design, biochemical and image analysis. CNNs have been used in drug discovery. In 2015, the company Atomwise introduced AtomNet, the first deep learning neural network for structure-based rational drug design. The system trains directly on 3-dimensional representations of chemical interactions. AtomNet was used to predict novel candidate biomolecules for multiple disease targets, most notably treatments for the Ebola virus and multiple sclerosis..

Another use of machine learning in diagnosis is in the study of X-rays images. We looked at the technology behind medical image processing in the lecture *Can Maths Save Your Life* . The techniques such as MRI, which I described in that lecture, lead to wonderfully clear images which can be used for a clear diagnosis. However, these methods take time to use. In certain cases of acute medical conditions, such as a knee or hip joint fractures, there is very limited time to make a diagnosis. Sophisticated imaging is not possible in this time, and doctors have to rely on standard X-rays, and they have to make a rapid assessment based on these. Furthermore, the right expertise to make a diagnosis may not be immediately available. In this case machine learning can certainly help with the diagnosis. At the University of Bath, Prof Richie Gill and Dr Ellen Murphy are pioneering a new method of machine diagnosis in which a CNN is trained on a set of X-ray images of hip joints with identified location and fracture type. The CNN can then be used on a new image to both identify the joint and to classify the types of fracture.

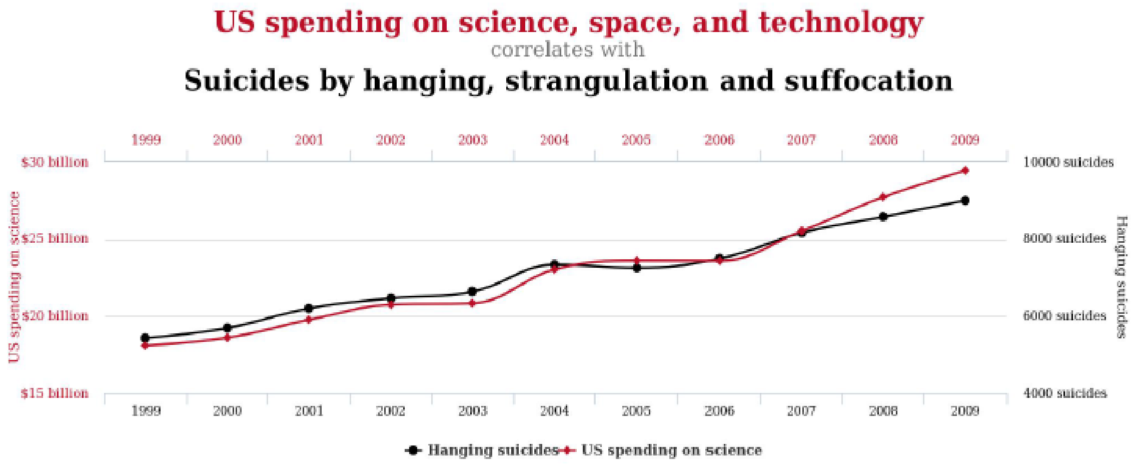
Reliability, Ethical and Legal Issues

Deep neural networks are a VERY black box! As we have seen, you train them by feeding them a large amount of data. They then apply complex, and hard to analyse, optimisation methods to determine their inner architecture. They are then used to make decisions on complex problems, which have a direct effect on people's lives. In the case of chess this works, and appears to have no ethical problems (although we see soon that it does). But can we really justify the use of machine learning in the case of vetting job applications, or other similar issues.

I believe, amongst many others, that the answer must be no. The following are some reasons for this.

Correlation does not imply causation

This is a classical issue in statistics. If you have a large amount of data and then compare one set of data against another then you may well find examples which seem to be linked. In the following (taken from [9]) we compare science spending in the USA with the number of suicides.

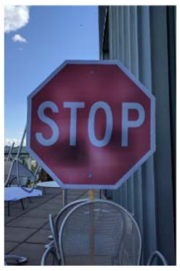


They appear to be closely linked. We might conclude from this graph (and a machine learning might well deduce) that suicides go up with science spending. Thus the US should immediately reduce science spending to cut down on the number of suicides in the country. The reason that they should not do this is that changes in one set of data does not necessarily cause changes in the other. In the example above it is much more likely that other changes in the US, such as a rise in GDP, lead both to a rise in suicides and in science funding. The more random sets of data that you look at, the more likely it is that you will find false correlations between them. Or more poetically 'you can find anything you look for in the clouds'. Be warned, this process had led to serious miscarriages of justice!

Statisticians are well aware of this problem and have tests to prevent making false deductions. Machine learning used carelessly may have no such tests. We have little guarantee that a machine learning algorithm may not draw false conclusions from a large training set.

Machine learning can make mistakes

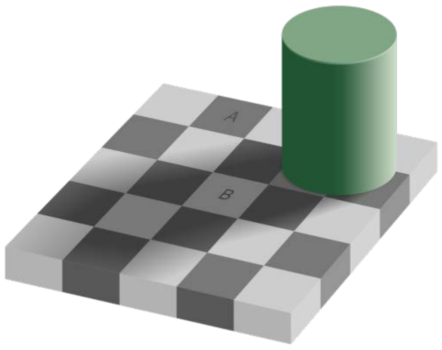
Not only are machine learning algorithms far from perfect, they do not come with any estimate of the uncertainty of their predictions.



One of the uses of machine learning is in the design and operation of automatic cars. Clearly such a car must be able to detect, identify, and recognise the meaning of, standard road signs. In another example from [9] (see also [10]) shown on the right we see what is clearly a Stop sign. This is unremarkable. However a machine learning algorithm trained on a set of road signs identified this as a 45mph speed sign! Fortunately this wasn't used in a real application. This is a really simple problem compared with that of vetting a job applicant. However, the machine learning algorithm has made a mistake which, if not spotted, could have easily led to fatal consequences.

Machines aren't humans

Another issue with machine learning is implicit in the title. It is learning by machines. Humans and machines do not always 'think' in the same way. To illustrate this I will look briefly at the problem of perception. This is important in the area of machine vision in which a robot views its environment and makes decisions about what it sees. Clearly we would want the robot to see things in the same way that humans do. However, this may not be possible. An illustration of this is given by the checker shadow illusion [11] illustrated below.



Have a look at this picture and decide for yourself which of the two squares A and B is the brightest.

It may surprise you to know that in fact they are equally bright! Our human brains compensate for the shadow of the cylinder by making square B *appear* to be brighter than square A. This is how humans see things. However a computer will see this differently, as from a simple measurement squares A and B are exactly as bright as each other.

No doubt there are many other examples where machines and humans 'think differently', nor should we expect them to do so.

Implicit Bias

We might think that one of the advantages of a machine learning approach to the selection of applicants (along with other human related issues) is that, unlike humans, they do not show bias. However, this is certainly not the case. When we train a neural network, on say a set of CVs, then the training set of data itself contains all of the biases of the original decision makers. These biases will be absorbed into the inner workings of the machine learning algorithm. This behaviour has certainly been found to exist in such trained machines. However, unlike a human operator, it is harder to identify reasons for this. Before using such a machine learning method it is *essential* that the training set is carefully tested for bias in advance of it being used to train a machine. But how can we guarantee or enforce this?

The above shows that we should not necessarily trust a machine learning algorithm. It is evidently not ethical to use a machine to make decisions when we do not trust it for making good decisions. Even if we trust a machine learning system, such as in the chess playing algorithms, there are still significant ethical issues in the way that they are used. For example, now it is possible to carry a computer in your pocket, which can beat a grand master at chess, the temptations of cheating have become too great for many to bear. As a result we have seen a significant rise in the amount of cheating in these games mostly by finding ways of looking at the computer without anyone else noticing. To counter this there has also been a lot of progress in the detection of cheats. My friend Ken Regan at the University of Buffalo, NY (and a mathematical logician), is a world expert in detecting chess cheats, which he does by sophisticated statistical methods.

The ethical issues involved in machine learning are so great that my university at Bath is holding a special semester this year, devoted just to these topics, with a big public debate at the end [12]

Can robots do maths?

This lecture is called 'is a mathematician a robot' . So we ask now the question of whether machine learning will lead to a robot that can do mathematics. A naïve answer is 'yes of course'. Indeed according to a well-known celebrity (who I won't identify by name) *now we have computers we don't need university maths departments any more.* It is certainly true that computers can be trained to do mathematical calculations very fast. Indeed my (day) job as a numerical analyst is to do exactly that. Furthermore the Todai robot beat 99.1% of the candidates in the mathematics part of the Japanese university entrance exam (although it only beat 64% of the physics candidates). However, in all of these cases the computer is basically doing arithmetical operations, pre-programmed by a human operator. None of this shows that a computer could learn to do deep mathematics, in the sense of identifying and proving results such as Fermat's last theorem [Singh]. Nor indeed are any of these robots displaying *mathematical reasoning*. This is an important question and goes back (at least) to the great mathematician David Hilbert who at the start of the 20th Century started a program in which the objective was that all mathematical statements could be written in a precise formal language, manipulated according to well defined rules, and a proof that all true mathematical statements can be proved in the formalism. This would essentially reduce mathematical theorems and proofs to exercises in arithmetic. I see this as entirely consistent with an expectation that a machine learning approach would be appropriate for mathematics. Unfortunately for this approach in 1931 Kurt Gödel, in his incompleteness theorem showed that Hilbert's program was unattainable for many important areas of mathematics. This doesn't of course mean that computers cannot be incredibly helpful in mathematical discovery. An excellent example of this was the proof of the celebrated *four colour theorem* [13] which was very much a combined effort of mathematicians and computers working together. Indeed computers are opening up whole areas of experimental mathematics, aiding mathematician in making new discoveries [14].

Having said all of this, there is (as far as I am aware) no machine learning program which has made a dent in the notoriously hard problem in arithmetic of factorising integers. (The difficulty of doing this is a key element of the security of modern codes.) Much the same can be said of the (equally notorious) travelling salesman problem. I am (personally) waiting to see if machine learning can ever replace modern algorithms for the five day weather forecast.

The future; how creative can a robot be?

How close are we to general AI? Despite the advances I have described in machine learning, this still seems as far away as ever, in the sense of a computer that can pass the Turing test. On a, possibly, simpler matter, can a machine learning algorithm solve a problem that everyone would consider to be creative. As a test I would suggest that we play a computer all of the works of the great composers. This would be considered its training set. We would then as a task ask it to compose a completely novel symphony. If that could then be played to an expert musician and found indistinguishable in imagination and invention from that created by a human being. I wonder if I will see that day?

References

[1] Royal Society report on machine learning, *Machine learning, the power and promise of computers that learn by example*, April 2017. ISBN 978-1-78252-259-1

[2] C. Shannon, *Programming a Computer for Playing Chess.* Philosophical M
agazine, Ser.7, Vol. 41, No. 314, (1950)

[3] https://thebestschools.org/magazine/brief-history-of-computer-chess/

[4] https://rjlipton.wordpress.com/2017/12/17/truth-from-zero/#more-14559

[5] D Silver et. Al., *Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm*, https://arxiv.org/pdf/1712.01815.pdf

[6] F. Rosenblatt, *Principles of neurodynamics.* (1962), New York: Spartan

[7] http://playground.tensorflow.org

[8] W. Press et. al. *Numerical recipes in C, the art of scientific computing,* (1988), CUP.

[9] J. Davenport, *Artificial intelligence and machine learning,* Report to the Cabinet Office, (2017).

[10] I. Evtimov, K. Eykholt, E. Fernandes, T. JKohno, B. LI, A. Prakash, A. Rahmati, and D. Song, *Robot Physical-world attacks on machine learning models*, (2017), https://arxiv.org/abs/1707.08945

[11] https://en.wikipedia.org/wiki/Checker_shadow_illusion

[12] The University of Bath special semester in machine learning:
http://www.bath.ac.uk/imi/events/machine-learning.html

[13] R. Wilson, *Four Colours Suffice: How the Map Problem Was Solved,* (2003), Princeton University Press.

[14] http://theconversation.com/will-computers-replace-humans-in-mathematics-60168

Acknowledgements.

I am very grateful to Profs James Davenport and Mike Tipping, in the Department of Computer Science at Bath, Kevin Olding in the Department of Mathematics at Bath and my son Jeremy Budd (PhD student in mathematics at Nottingham), for their help in writing this lecture. I am also grateful to the Arthroplasty for Arthritis charity for funding the work on hip fracture recognition at Bath which is described in Section 5.