Back to article page

London Review of Books

The Concept of 'Cat Face'

Paul Taylor on machine learning

Over the course of a week in March, Lee Sedol, the world's best player of Go, played a series of five games against a computer program. The series, which the program AlphaGo won 4-1, took place in Seoul, while tens of millions watched live on internet feeds. Go, usually considered the most intellectually demanding of board games, originated in China but developed into its current form in Japan, enjoying a long golden age from the 17th to the 19th century. Famous contests from the period include the Blood Vomiting game, in which three moves of great subtlety were allegedly revealed to Honinbo Jowa by ghosts, enabling him to defeat his young protégé Intetsu Akaboshi, who after four days of continuous play collapsed and coughed up blood, dying of TB shortly afterwards. Another, the Ear Reddening game, turned on a move of such strength that it caused a discernible flow of blood to the ears of the master Inoue Genan Inseki. That move was, until 13 March this year, probably the most talked about move in the history of Go. That accolade probably now belongs to move 78 in the fourth game between Sedol and AlphaGo, a moment of apparently inexplicable intuition which gave Sedol his only victory in the series. The move, quickly named the Touch of God, has captured the attention not just of fans of Go but of anyone with an interest in what differentiates human from artificial intelligence.

DeepMind, the London-based company behind AlphaGo, was acquired by Google in January 2014. The £400 million price tag seemed large at the time: the company was mainly famous for DQN, a program devised to play old Atari video games from the 1980s. Mastering Space Invaders might not seem, on the face of it, much to boast about compared to beating a champion Go player, but it is the approach DeepMind has taken to both problems that impressed Google. The conventional way of writing, say, a chess program has been to identify and encode the principles underpinning sound play. That isn't the way DeepMind's software works. DQN doesn't know how to repel an invasion. It doesn't know that the electronic signals it is processing depict aliens – they are merely an array of pixels. DeepMind searches the game data for correlations, which it interprets as significant features. It then learns how those features are affected by the choices it makes and uses what it learns to make choices that will, ultimately, bring about a more desirable outcome. After just a few hours of training, the software is, if not unbeatable, then at least uncannily effective. The algorithm is almost completely generic: when presented with a different problem, that of manipulating the parameters controlling the cooling systems at one of Google's data centres with the aim of improving fuel efficiency, it was able to cut the electricity bill by up to 40 per cent.

Demis Hassabis, the CEO of DeepMind, learned to play chess at the age of four. When he was 12 he used his winnings from an international tournament to buy a Sinclair ZX Spectrum computer. At 17 he wrote the software for *Theme Park*, a hugely successful simulation game. He worked in games for ten more years before studying for a PhD in cognitive neuroscience at UCL, then doing research at Harvard and MIT. In 2011 he founded DeepMind with, he has said, a two-step plan to 'solve intelligence, and then use that to solve everything else'.

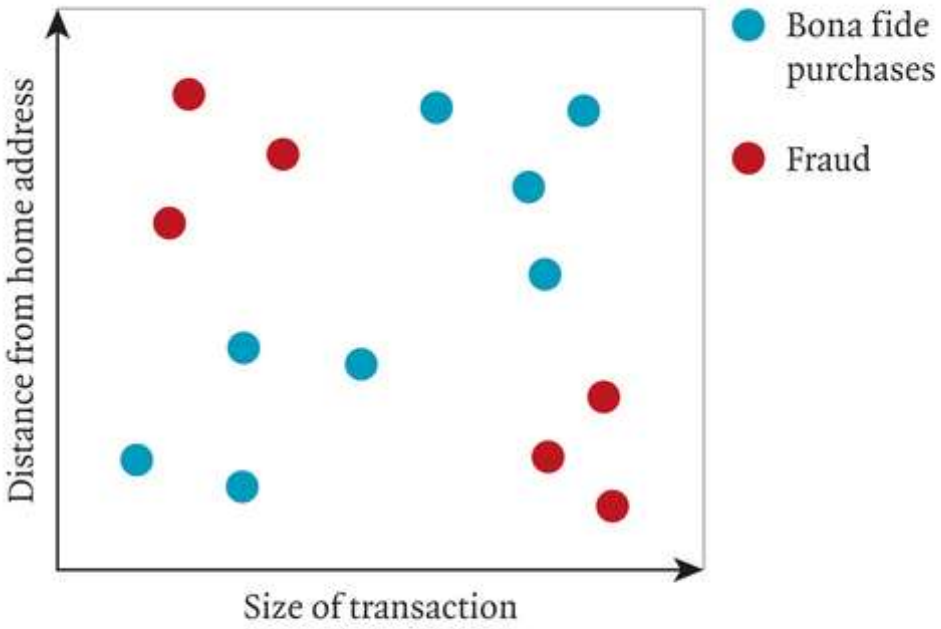
*

In 1965 the philosopher Hubert Dreyfus published a critique of artificial intelligence, later worked up into a book called *What Computers Can't Do*, in which he argued that computers programmed to manipulate symbolic representations would never be able to complete tasks that require intelligence. His thesis was unpopular at the time, but by the turn of the century, decades of disappointment had led many to accept it. One of the differences Dreyfus identified between human intelligence and digital computation is that humans interpret information in contexts that aren't explicitly and exhaustively represented. Someone reading such sentences as 'the girl caught the butterfly with spots,' or 'the girl caught the butterfly with a net,' doesn't register their ambiguity. Our intuitive interpretation in such cases seems to arise from the association of connected ideas, not by logical inference on the basis of known facts about the world. The idea that computers could be programmed to work in a similar way – learning how to interpret data without the programmer's having to provide an explicit representation of all the rules and concepts the interpretation might require – has been around for almost as long as the kind of symbol-based AI that Dreyfus was so scathing about, but it has taken until now to make it work. It is this kind of 'machine learning' that is behind the recent resurgence of interest in AI.

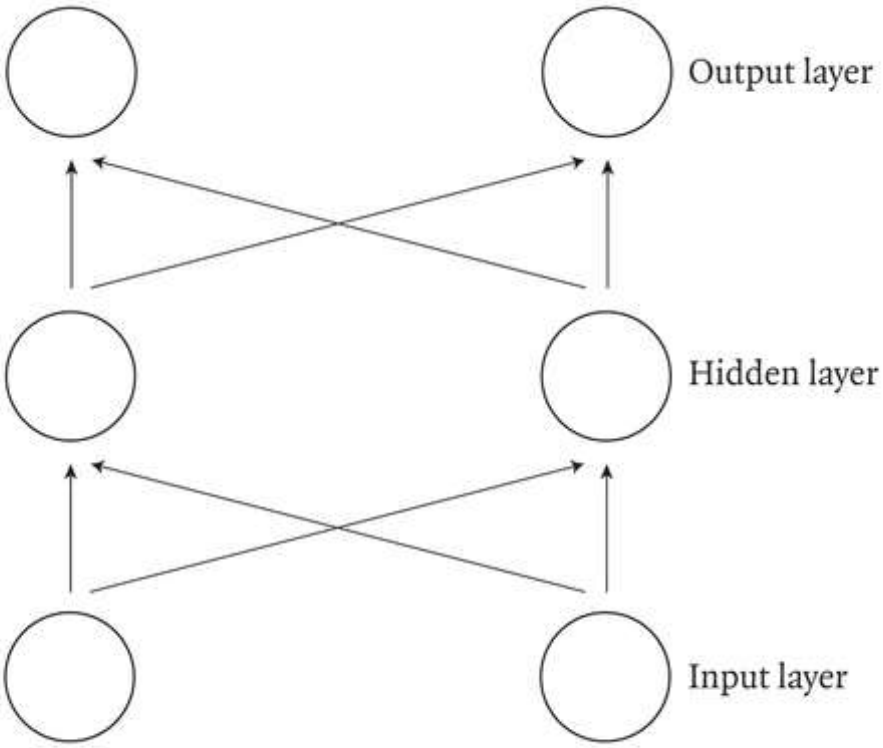
The best-known example of early machine-learning was the Perceptron, built at Cornell in 1957 to simulate a human neuron. Neurons function as simple computational units: each receives multiple inputs but has only a single output – on or off. Given numerical data about examples of a particular phenomenon, the Perceptron could learn a rule and use it to sort further examples into sets. Imagine that the Perceptron was trained using data on credit card transactions, some of which were known to be fraudulent and the rest above board. To begin with, each element of information fed to the Perceptron – it might be the size of the transaction, the time since the previous transaction, the location, or information about the vendor – is assigned a random weight. The machine submits the weighted values of the elements in each case to an algorithm – in the simplest case, it might just add them up. It then classifies the cases (fraud or not fraud) according to whether the total reaches an arbitrary threshold. The results can then be checked to find out whether the machine has assigned the example to the right or wrong side of the threshold. The weights given to the various inputs can then gradually be adjusted to improve the machine's success rate.

Given enough data and a well-structured problem the Perceptron could learn a rule that could then be applied to new examples. Unfortunately, even very simple problems turned out to have a structure that is too complex to be learned in this way. Imagine that only two things are known about credit card transactions: their amount, and where they take place (since

both must be expressed as numbers, let's assume the location is expressed as the distance from the cardholder's home address). If fraud is found to occur only with large purchases or only with distant ones, the Perceptron can be trained to distinguish fraudulent from bona fide transactions. However, if fraud occurs in small distant purchases and in large local ones, as in Figure 1, the task of classification is too complex. The approach only works with problems that are 'linearly separable' and, as should be clear from Figure 1, no single straight line will separate the fraud cases from the rest.



Interest in the approach faded for a while, but at the end of the 1970s people worked out how to tackle more complex classification tasks using networks of artificial neurons arranged in layers, so that the outputs of one layer formed the inputs of the next. Consider the network in Figure 2.



Imagine the two nodes in the input layer are used to store, respectively, the size and location of each credit card transaction. If the left-hand node in the middle layer can be trained to detect just the cases in the top left of Figure 1 (which are linearly separable) and the right-hand node can be trained to detect only the cases to the bottom right, the two inputs to the output layer would measure the extent to which a case is a) small and distant, and b) large and local. Bona fide transactions will score low on both measures, fraud transactions will score highly on one or the other, so the two classes can now be divided by a straight line. The difficult part of all this is that the network has to identify the concepts to be captured in the hidden middle layer on the basis of information about how changing the weights on the links between the middle and output layers affects the final classification of transactions as fraud or bona fide. The problem is solved by computing a measure of how a change in the final set of weights changes the rate of errors in the classification and then propagating that measure backwards through the network.

For a while multi-layer networks were a hot topic, not least because people were excited by the explicit analogy with human perception, which depends on a network of cells that compute features in a hierarchy of increasing abstraction. But, as before, early promise gave way to disappointment. The backwards propagation of errors seemed a hopelessly inefficient training algorithm if more than one or two layers separated the input and output layers. Shallow networks couldn't be programmed to complete challenging tasks in vision or speech recognition, and given simpler tasks they were outperformed by other approaches to machine learning.

The challenge in machine learning is not so much finding a rule that correctly classifies a particular set of data, as finding *the* rule that is most likely to work for future examples. One approach that would work for a linearly separable problem would be to divide the two sets

using the straight line that maximises the distance between the line and the nearest point in each of the two sets. Finding that line is mathematically relatively straightforward. But the most interesting problems tend not to lend themselves to linear separation. A mathematically elegant solution is to project the data into a higher dimensional space where a simple separation can be found by a process of iterative searching. For the data in Figure 1, the search would be for a mathematical function that takes the values of the x and y co-ordinates for each point and uses them to derive a z co-ordinate, so that the red points hover at a greater height than the blue ones.

The representation of credit card transactions as points on a 2D surface or in a 3D volume in this way is, of course, a spatial metaphor. In reality each transaction is just a set of numbers, and in most problems there will be a lot more than three numbers to deal with. A 2D space is divided by a line, a 3D space by a plane. A 'space' of more dimensions than that is divided by a hyperplane. Classification algorithms of the sort I've been describing are known as 'support vector machines'. Their job is to identify the hyperplane that optimally separates points in an n-dimensional space. SVMs dominated machine learning from the 1990s until very recently; they have the sought-after property, not shared by neural networks, that if the computation converges on a solution, it is guaranteed to be the best available one.

Imagine a classifier is to be trained using a hundred images: fifty of them are of the letter 'i', each in different handwriting in shades of grey on a white background; the other fifty, similarly presented, are of 'j's. If each image is 32 pixels high and 32 pixels wide, it can be represented as a single point in a 1024-dimensional space ($32^2 = 1024$), where each dimension corresponds to a pixel, and the value on the dimension ranges from 0, which represents white, to 255, which represents black. The data for the set of images is completely represented as a hundred points in this 1024D space. A support vector machine could attempt to find a hyperplane that divides the space so that, ideally, all the points corresponding to the images of 'i's are on one side and all the 'j's on the other. However, the hundred images will form a diffuse cloud that takes up only a tiny fraction of the total feature space, as it's called, and they will almost certainly be unhelpfully distributed within it. This is a common problem in machine learning: the feature space is only very sparsely populated by the data.

An alternative is to build a new feature space, a system of co-ordinates adapted to the data we are interested in. For example, the origin of the new system of co-ordinates could be placed at the centre of the cloud of points and a line drawn that passes through the origin and goes as close as possible to as many points as possible. A second line through the origin could be drawn at 90° to the first, again as close to as many points as possible; and then a third, and so on until, say, ten dimensions have been defined. Each image could now be given a set of co-ordinates in the new ten-dimensional space; it would no longer be represented by 1024 pixels but by a set of ten numbers that is both a much better characterisation of the data and a more parsimonious input to a support vector machine. Each of these ten numbers corresponds to a value for an abstract feature which has been derived by the computer from an analysis of the data as a whole. This abstract feature will correspond to some way in which

the 'i's and 'j's vary, but it may or may not correspond to an intuitive human interpretation of the data.

Between, roughly, 1990 and 2010 most research in machine learning was focused on statistical techniques such as support vector machines and the attempt to derive feature spaces that made classification easier. As computers became more and more powerful and datasets larger and larger, it became more practical to leave the computers to figure out the right feature space to use. That is what seems magical about software like DeepMind's: computers are abstracting from experience something which can then be applied in reasoning about a problem. It seems right to say that the computer has learned a concept.

*

In 2006 Netflix offered a prize of $1 million to anyone who could improve on the algorithm it used to generate recommendations for its customers. To give contestants something to work with, it released a database of 100,480,507 ratings that 480,189 users had given to 17,770 movies. The prize was awarded in 2009 to a team that used a blend of different algorithms, though most of their success seemed to be down to two of these approaches.

In the first approach a large matrix is created in which each movie is a row and each user a column. In roughly 1 per cent of the cells there will be a number between one and five which indicates the rating a particular user gave to a particular movie. The challenge is to use the data to predict whether a given user would like a movie they haven't seen yet. This corresponds to using the values in the filled cells to predict the rating that should go into the empty cells. The assumption is made that a smallish number of features – thirty, say – determine whether or not a user likes a movie. No assumption is made about what these features are (a happy ending, a big budget, a strong female lead etc), just about how many there are. The problem then reduces to identifying two much smaller matrices. One has a row for each movie and a column for each of thirty features and records the extent to which a given feature is present in the movie. The other has a row for each feature and a column for each user and records the extent to which a user has demonstrated a preference for a given feature. The product of these two matrices will then generate a rating corresponding to each cell in the original large matrix.[*] The problem is that because we didn't start out by making assumptions about what the salient features are, none of the values in either of the smaller matrices is known. The solution, as with other approaches to machine learning, is to start with a guess, see how the generated predictions for filled cells compare with the known ratings and then make repeated adjustments to minimise the average error.

The other approach also assumed that the required predictions could be generated from a small set of salient features, but used a variant of a neural network, known as a Restricted Boltzmann Machine or an autoencoder, to derive the features from the data. A conventional neural network is trained on samples with a known classification until it learns a rule. An autoencoder is trained on samples of unclassified data until it learns how to generate similar patterns of data. The Netflix autoencoder looks just like the neural network in Figure 2, but with many more nodes.

Although Netflix awarded their prize in a blaze of publicity, the winning strategy was never implemented. In part this was because Netflix had already begun to distribute movies via a streaming service. Customers could now pick what they wanted to watch there and then, and somehow this meant they were less likely to pick the kinds of film that earn high ratings (no one wants to watch *Schindler's List* when they've just put the kids to bed on a Tuesday night), so predicting ratings was no longer the best way to make recommendations. (There was the added difficulty that although Netflix believed it had anonymised the data it released, it had included information about films people had watched but not rated and also the dates on which they were watched. If you knew two or three dates on which you and your partner had watched films together, that would probably be enough to allow you to pick out your column in the data, which might make it possible for you to find out what your partner had watched without you. This became known as the '*Brokeback Mountain* problem': in 2009 Netflix was successfully sued for breach of privacy.)

About the same time Netflix launched its competition, Geoffrey Hinton, one of a dwindling number of researchers still working on neural networks, realised that if the lower levels of a neural network could be programmed using autoencoders, then the bottom of a deep neural network could learn a feature space that the top of the network could use to perform a classification. In 2009 two of Hinton's students used this approach to devise a speech recognition system which was, within a few years of development, outperforming competitors that had been refined for thirty years or more. One of the students went on to work for Microsoft, the other for Google, and by 2012 their work was the basis of the algorithm that made it possible for Android phones to respond reasonably reliably to spoken queries and commands.

Google also devoted some space in its massive computing infrastructure to build what was, until last year, the world's biggest artificial neural network. Inception, as it was called, was trained on a thousand machines running in parallel for three days. It analysed still images selected at random from ten million YouTube videos. Whereas earlier neural networks had been used to perform low-level image processing or speech recognition, the much taller stack of layers in this monster network made it possible for it to recognise human faces or (this tells us more about YouTube than it does about AI) cats' faces. If this network had been fed thousands of images labelled as 'contains cats' or 'doesn't contain cats' and trained to work out the difference for itself by iteratively tweeking its 1.7 billion parameters until it had found a classification rule, that would have been impressive enough, given the scale of the task involved in mapping from pixels to low-level image features and then to something as varied and complex as a cat's face. What Google actually achieved is much more extraordinary, and slightly chilling. The input images weren't labelled in any way: the network distilled the concept of 'cat face' out of the data without any guidance.

Last year Hinton, by now a Google employee himself, gave a talk to the Royal Society in which he discussed some of this history and spoke about new developments. One of them is the recurrent neural network, which adds the innovation of weighted links not just between nodes but between instances of the same node at successive steps in the computation. It is as if the network shown in Figure 2 were replicated in 3D, to produce a stack of identical

networks with links rising up out of the page from each node to the node above. Each layer in the stack, however, doesn't represent a part of the network, but the state of the network at a point in time, so the bottom layer is the network at the start of training, the next layer is the network after the first cycle of training and so on. The point is that when a conventional neural network learns how to classify examples, whether they are images or customer ratings, it doesn't matter in what order the training examples are processed. Recurrent networks, in contrast, are ideally suited to analysing data which is inherently sequential, such as speech or language. Researchers at Google have for some years been programming recurrent neural networks to predict the next word in a sentence. Almost as a by-product this work creates a point in a high-dimensional feature space for each word. The features have no human interpretation: they are just the values of the hidden nodes in a network that was trained for a prediction task. But the researchers noticed that words with similar meanings had similar representations in the feature space. Even more astonishing, if you subtracted the features for 'uncle' from those for 'aunt' you got almost the same answer as if you subtracted 'king' from 'queen', suggesting that this abstract, computer-derived space has a dimension that correlates with gender. One practical result of this work is an approach to machine translation that involves mapping between the feature representations of words in different languages. The process is still in its infancy and is currently outperformed by more conventional methods, but it is getting better faster.

*

The solving of problems that until recently seemed insuperable might give the impression that these machines are acquiring capacities usually thought distinctively human. But although what happens in a large recurrent neural network better resembles what takes place in a brain than conventional software does, the similarity is still limited. There is no close analogy between the way neural networks are trained and what we know about the way human learning takes place. It is too early to say whether scaling up networks like Inception will enable computers to identify not only a cat's face but also the general concept 'cat', or even more abstract ideas such as 'two' or 'authenticity'. And powerful though Google's networks are, the features they derive from sequences of words are not built from the experience of human interaction in the way our use of language is: we don't know whether or not they will eventually be able to use language as humans do.

In 2006 Ray Kurzweil wrote a book about what he called the Singularity, the idea that once computers are able to generate improvements to their own intelligence, the rate at which their intelligence improves will accelerate exponentially. Others have aired similar anxieties. The philosopher Nick Bostrom wrote a bestseller, *Superintelligence* (2014), examining the risks associated with uncontrolled artificial intelligence. Stephen Hawking has suggested that building machines more intelligent than we are could lead to the end of the human race. Elon Musk has said much the same. But such dystopian fantasies aren't worth worrying about yet. If there is something to be worried about today, it is the social consequences of the economic transformation computers might bring about – that, and the growing dominance of the small number of corporations that have access to the mammoth quantities of computing power and data the technology requires.

In my field, medical research, applications of machine learning have tended to use datasets with a few thousand or maybe only a few hundred cases. Datasets of 100,000 or 1,000,000 cases are becoming more common, but datasets on the scale of those available to Google or Facebook simply don't exist in many of the fields where AI could be useful. The Royal Free Hospital NHS Trust recently agreed to share data on 1.6 million patients with DeepMind. It's a lot of data but most of it will not concern conditions where deep learning could provide insights – most of the reports talk about a project looking for signs of kidney disease. The quality of data extracted from hospital computer systems is often poor, which will be another challenge. The deal, which came to light when it was leaked to the *New Scientist*, generated a spate of hostile news coverage, with many observers questioning the ethical and legal basis for releasing the data without the patients' consent and for a purpose other than their care. The company has now signed another deal, this time with Moorfields Eye Hospital, giving it access to the retinal scans of a million patients. This new agreement was announced with much greater regard to public anxieties about the uses of confidential data. In truth, the constraints in both deals are pretty robust and the scope for accidental disclosure or inappropriate use is limited. A greater concern is that these research collaborations, entered into by clinicians and academics excited by the potential of the technology, involve transferring a valuable public asset to a private company. Healthcare is one of the largest industries in the world, one where AI could prove transformative. The one scarce resource that AI companies need in order to build their algorithms is data.

Computers haven't entirely outsmarted us yet. To pick the best move in a game, you need to consider every possible move and every possible move that might follow from that move, until the game is won. In Go there are on average 250 possible options to consider for each move, and a typical game between experts might last 150 moves. The number of ways a game can unfurl is so vast that an exhaustive search through every possibility is unfeasible, even for Google. There are two ways to limit the search. One is to have a policy that restricts the search to the most plausible moves; the other is to understand the value of a position, so that there is no need to search for further moves from unpromising positions. AlphaGo contains two 13-layer neural networks: a policy network, which computes the probability that any particular move, out of all possible moves, will be made at any given point in a game; and a value network, which computes the probability of winning from that position. DQN learned to play Atari games from scratch, but the AlphaGo policy network uses data from a server that allows people to play games against each other on the internet – with the result that it now has a database of thirty million game positions. Training the policy network on this database enabled it to predict with 57 per cent accuracy the move that the humans – all of them amateurs – would make. That doesn't sound a great basis from which to improve on human play, but it is enough, and the engineers say that improvements at this stage in the process have huge consequences later on. To move from a passable prediction of human play to a policy that would beat an expert, the system was then programmed to play out games in which the current version of the policy network competed with an adversary selected at random from earlier versions, using the learning algorithms developed in DQN to adjust the weights in the networks to obtain more successful policies. In the final stage of the process the

software played out games starting from any one of the thirty million game positions recorded in the database. Here the program was playing against itself, and adjusting weights to improve its assessment of the value of positions. The whole thing is an incredible engineering achievement.

In its match with Lee Sedol, AlphaGo won the first three games, and with them the series. One hour and 13 minutes into the fourth game, with barely thirty minutes left on his clock, Sedol, playing white, thought for 16 minutes before making an aggressive move into black's territory, on the left edge of the centre. Two or three moves followed fairly quickly, and it was clear that Sedol would need to make two or three forcing moves if he was going to make the attack count. It wasn't obvious, at least to the commentary team, what they could be. Then, after a terrifying six minutes' further thought, Sedol placed a white stone in between two black stones at the right side of the centre. Afterwards the AlphaGo team found that its policy network had rated the chances of an opponent making this move at 1 in 10,000. Michael Redmond, a top-ranked professional player commentating on the game, admitted he hadn't seen it either, but he recognised its significance immediately: if black didn't find a response Sedol would be able to make his forcing moves. AlphaGo, however, didn't seem to realise what was happening. This wasn't something it had encountered in the amateur play on which the policy network was trained, or in the millions and millions of games it had played with itself. At the post-match press conference Sedol was asked what he had been thinking when he played it. It was, he said, the only move he had been able to see.

[*] The conventional approach to multiplying matrices allows a matrix with m rows and n columns to be combined with another with n rows and p columns to create a matrix with m rows and p columns.

Vol. 38 No. 16 · 11 August 2016 » Paul Taylor » The Concept of 'Cat Face'
pages 30-32 | 5101 words

^ Top