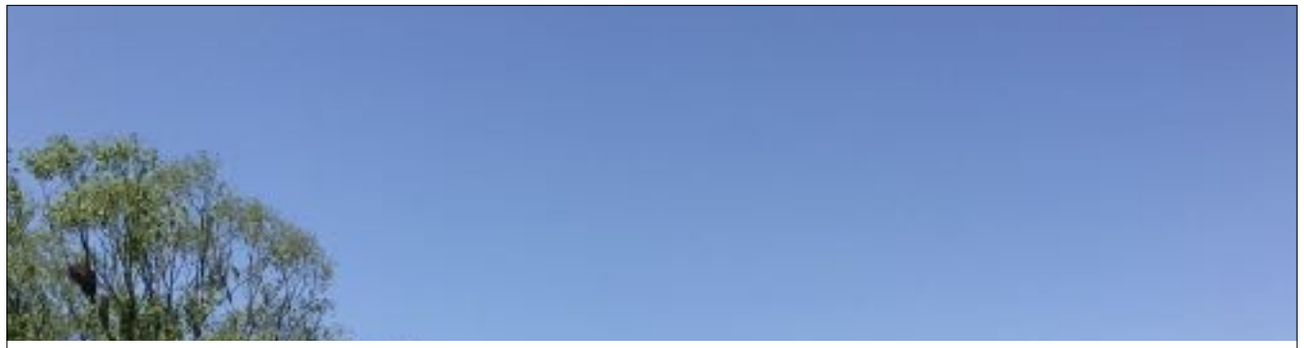


Algebra and Artificial Intelligence

Daniel Murfet

4th May 2018

- GitHub*:* https://github.com/dmurfet/deeplinearlogic
- J. Clift and D. Murfet “Cofree coalgebras and differential linear logic”, *arXiv: https://arxiv.org/abs/1701.01285.*
- J. Clift and D. Murfet “Encodings of Turing machines in linear logic”, *in preparation.*
- J. Clift and D. Murfet “Derivatives of Turing machines in linear logic”, *in preparation.*

In July 2016 I spoke at the annual conference for the Australasian Association for Logic here in Melbourne about the integration of the semantics of linear logic with neural networks. Some of the theoretical aspects of what I spoke about there have been developed further in my joint work with James Clift on differential linear logic. We have two more papers on this subject that we’re putting online soon, and I’ll be speaking about that work next Friday in the pure mathematics seminar.

Unfortunately the experimental aspects of the original project I spoke about in 2016 were not a success. We have opened the Github repository with all our TensorFlow code, including our implementation of DeepMind’s Neural Turing Machine.

Anyway, because of this work I have been paying attention to some of the research within artificial intelligence and machine learning. I think there are some interesting implications of for mathematicians and logicians, and I’d like to share some of my thoughts on that with you today.

Outline

1. State of the art
2. Some history
3. Blind spots

I’m going to start with a summary of some of the recent breakthroughs in AI and machine. Then I’ll rewind the clock a bit and talk about how we got here. Finally I’m going to speculate a bit on what this means for mathematicians and logicians going forward.

I. State of the art

Brief reminder on neural nets

A (*deep) neural network (DNN)* is a particular kind of function

$$F_w : \mathbb{R}^m \longrightarrow \mathbb{R}^n$$

which depends in a “differentiable” way on a vector of weights.

Example: Image classification, as in: cat or dog?

$$F_w\left(\quad \right) = \begin{pmatrix} 0.7 \\ 0.3 \end{pmatrix} \in \mathbb{R}^2$$

$$\cap$$

$$\mathbb{R}^{685 \cdot 685 \cdot 3}$$

$$p_{\text{cat}} = 0.7$$
$$p_{\text{dog}} = 0.3$$

A deep neural network is a particular kind of function F_w from m-tuples of real numbers to n-tuples of real numbers for some m and n, which depends in a differentiable way on a vector of weights w.

An example of a neural network is a function which takes an image as input. An image can be represented as a vector by simply taking the RGB values of each pixel and reading them off into a list. So if this cat picture here was 685 pixels in width and height, with three colour channels, I’d need 685 squared times 3 real numbers to describe it completely. The classification is also a vector, but of probabilities. If there are only two possible classes “dog” and “cat”, then the function outputs a column vector with the probability that the image lies in each class.

A *deep neural network* (DNN) is a particular kind of function

$$F_w : \mathbb{R}^m \longrightarrow \mathbb{R}^n$$

which depends in a “differentiable” way on a vector of weights.

A randomly initialised neural network is **trained** on sample data

$$\{(x_i, y_i)\}_{i \in I}\, , x_i \in \mathbb{R}^m, y_i \in \mathbb{R}^n$$

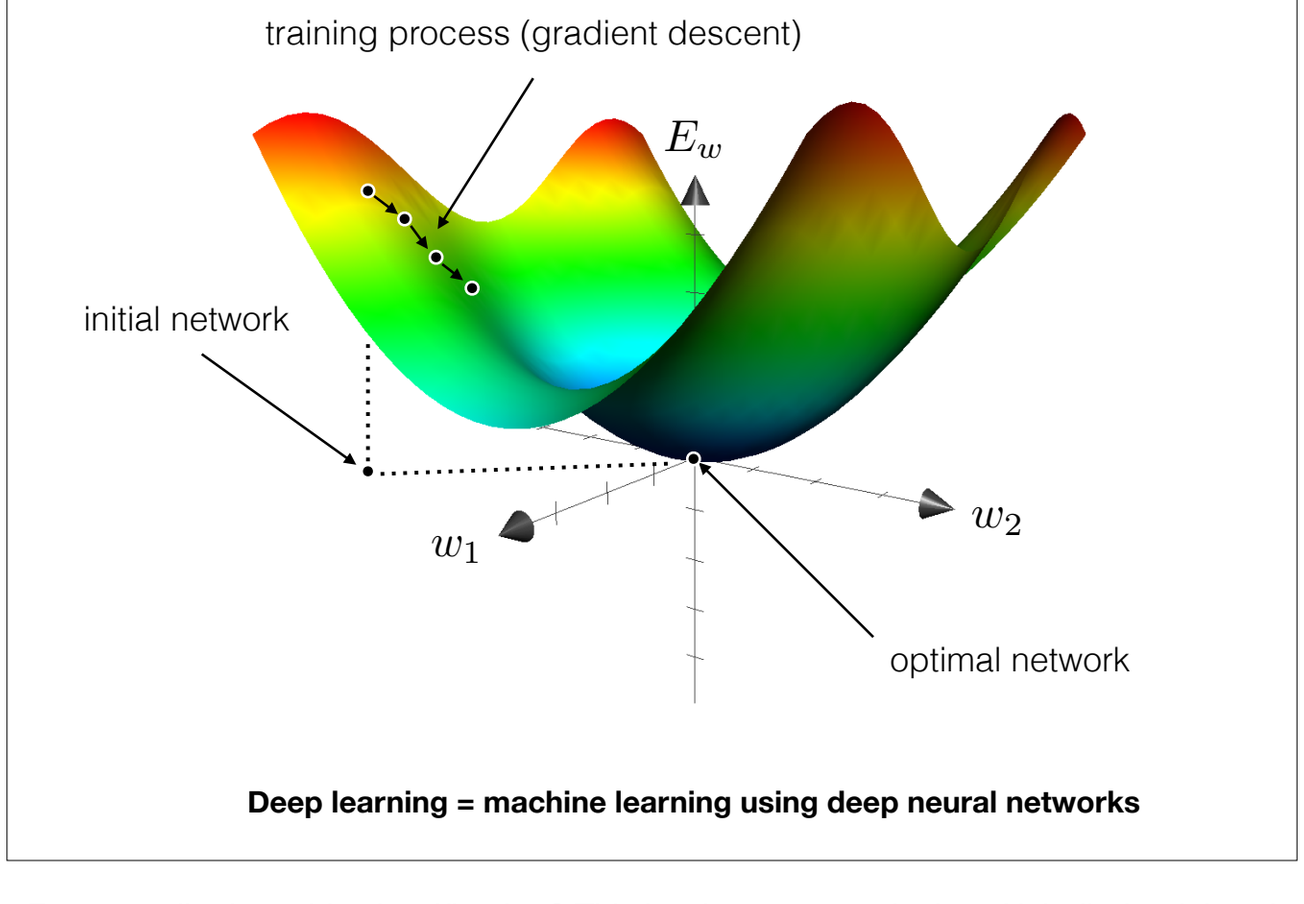
by varying the weight vector in order to minimise the error

$$E_w = \sum_i (F_w(x_i) - y_i)^2$$

Ok, so how does one actually produce a good image classifier?

The idea is that you start with essentially random weights, and therefore a function F_w which is very bad at doing whatever it is supposed to be doing, but then you iteratively improve the quality of the predictions by varying the weight vector w. You vary the weight vector in such a way as to steadily decrease the error of your model on some given training data.

Training data consists of a set of pairs of known values of F_w, that is, x_i and y_i where you know that F_w(x_i) should be equal to y_i. For example, a large corpus of images (that’s the x_i’s) which are labelled by humans as containing either cats and dogs (those labels give you the y_i’s). The error of F_w on this training data is given by the sum shown, which depends only on the weights. By varying the weights, we decrease the error, and if we’re clever in how we do so, the model we train in this way will also be good at classifying cats and dogs in images that are NOT part of the training data.



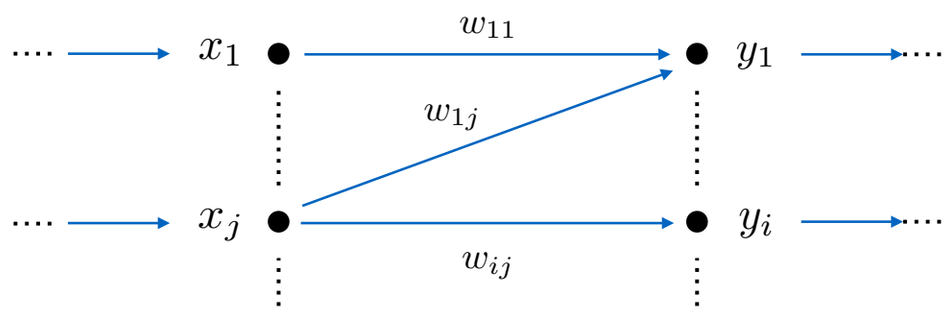
Deep learning = machine learning using deep neural networks

How do we find the parameters w so that F_w actually does this classification? This is where we need a lot of labelled training data to which we apply a learning algorithm called gradient descent, visualised here as flowing down an incline towards a local minimum. The horizontal plane represents the space of weights w_1, w_2 and the surface is the graph of the function which associates to a given choice of weights the error on the training data of F_w with those weights. So a good choice of weights will be a choice with low error, that is, a local minimum on this surface.

The initial network, that is, the random one, is represented by some point in this weight space and above that we have a corresponding point on the error surface. Gradient descent is a standard trick in calculus for moving on that surface steadily towards a local minimum, which is the optimal network that we want to find.

Intro to deep learning

A DNN is made up of many layers of “neurons” connecting the input (on the left) to the output (on the right) as in the following diagram:



$$y_i = \sigma\Big(\sum_j w_{ij} x_j\Big) \qquad \sigma(x) = \tfrac{1}{2}(x + |x|)$$

(omitting biases)

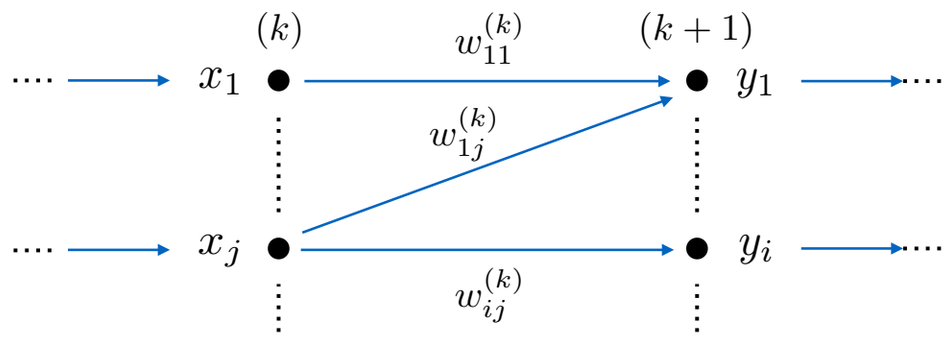
So that’s a general overview of how deep learning works. Now I’d like to tell you the details of how the function F_w is defined, which amounts to telling you what a deep neural network is.

A DNN has multiple layers (that’s why it’s called deep) which are usually presented as going from left to right, with the input (say an image) coming in from the left and the output on the right. Here I’m showing two layers in the middle of the network. The nodes are connected to one another in a particular pattern shown by the arrows, and each arrow is labelled with a weight which is a real number. The connection pattern is part of the design of the neural network architecture, and it varies from application to application, but for present purposes you should imagine that there is an arrow from each node on the left to each node on the right, and no other arrows.

I’d like you to think of each of these nodes as being decorated with a variable, x’s and y’s in this case. The value of each variable is described as a function of the values of the variables in the previous layer and the weights which connect the two layers, by this simple equation. The nonlinear function sigma is sometimes called a rectified linear unit. Perhaps it’s helpful to note that if x is positive, sigma(x) is just x, but if x is negative sigma(x) is zero.

Intro to deep learning

A DNN is made up of many layers of “neurons” connecting the input (on the left) to the output (on the right) as in the following diagram:

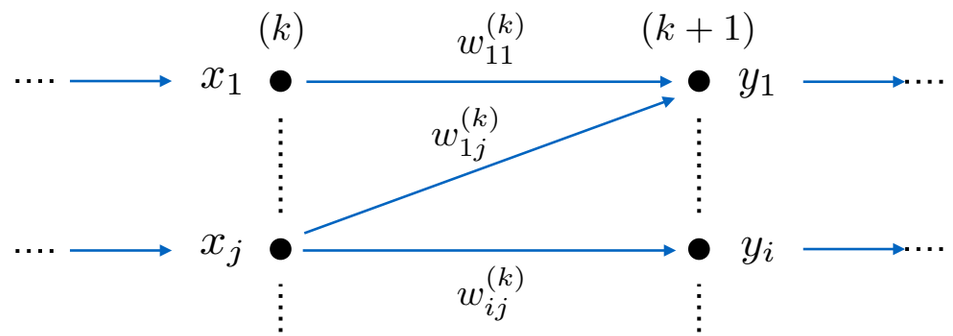


$$y_i = \sigma\Big(\sum_j w_{ij}^{(k)} x_j\Big) \qquad \sigma(x) = \tfrac{1}{2}(x + |x|)$$

I’d like in a moment to give the full equation for F_w, and to that end let me label the layers with integers. So let the layer on the left here be layer k, then the next layer is k+1, and the weights connecting layer k to the next layer are written with a superscript k. But apart from that notational shift, the equations haven’t changed.

Intro to deep learning

A DNN is made up of many layers of “neurons” connecting the input (on the left) to the output (on the right) as in the following diagram:



$$y_i = \sigma\left(\sum_j w_{ij}^{(k)} x_j\right) \qquad \vec{y} = \sigma\left(W^{(k)}\vec{x}\right) \qquad \sigma(x) = \tfrac{1}{2}(x + |x|)$$

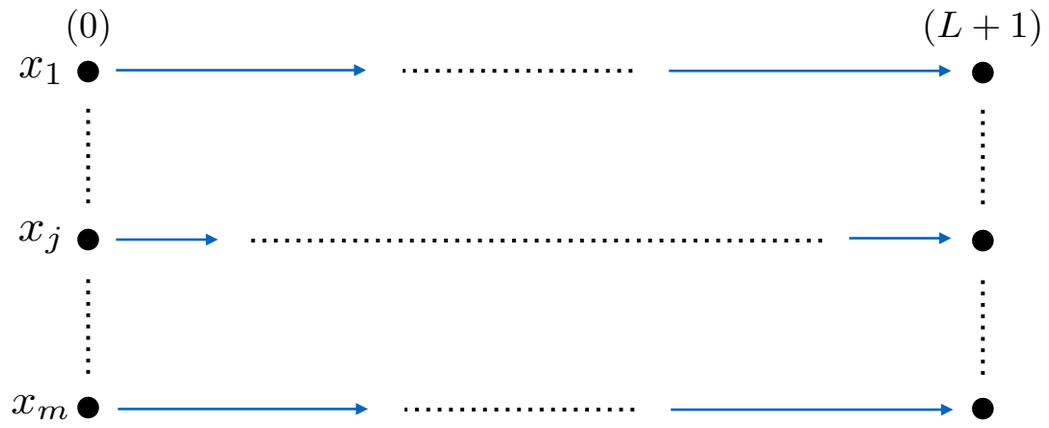
$$\vec{x} = (x_1, x_2, \ldots, x_j, \ldots)^T \qquad W^{(k)} = (w_{ij}^{(k)})_{i,j}$$

Now this sum over j here is just a matrix product, so if I collect all the weights between layers k and k+1 into a matrix W^k, and the values of the nodes into vectors x and y, I can rewrite this equation more compactly as shown here.

This single equation describes the values in layer k + 1 as a function of the values in layer k, and if I iterate it I can describe the values in the final layer, the output layer, as a function of the values in the input layer, just using these functions sigma and multiplication with matrices of weights.

Intro to deep learning

A DNN is made up of many layers of “neurons” connecting the input (on the left) to the output (on the right) as in the following diagram:



$$F_w : \mathbb{R}^m \longrightarrow \mathbb{R}^n \qquad F_w(\vec{x}) = \sigma\Big(W^{(L)} \cdots \sigma\Big(W^{(1)}\sigma\Big(W^{(0)}\vec{x}\Big)\cdots\Big)$$

Here I’ve zoomed out and I’m only showing the input layer, layer 0, and the output layer, which I’m calling layer L + 1. The iteration of the equation on the previous slide gives me the following formula for F_w, the overall function mapping inputs to outputs. Given a vector of input values x, I multiply x by the matrix of weights W^0, then apply sigma, then multiply by W^1, apply sigma, and so on until the final layer.

This function F_w IS the deep neural network. The weights are the real numbers labelling all the edges between all the layers in the neural network. These are also the entries of the matrices W. Varying these weights varies the function F_w, and by varying the weights in a supervised way we hopefully converge to a function F_w with desirable properties.

- **Computer vision** (Kirzhevsky, Sutskever, Hinton 2012)
  - History of ImageNet (Quartz article)
  - Interview with Yann LeCun (YouTube link)
- **Machine translation** (Google 2016)
  - Interview with Jeff Dean (podcast link)
- **Go playing** (DeepMind 2017)
  - AlphaGo movie (homepage)

I'd like to briefly highlight three areas in which there have been breakthroughs using deep learning. The first is in computer vision. At the 2012 NIPS meeting Kirzshevsky, Sutskever, and Hinton won the annual ImageNet image classification competition using a convolutional neural network, and this kicked off the deep learning craze. Computer vision remains the strongest domain of application for deep learning methods.

More recently deep learning based systems have been successful at machine translation of text between languages. Google last year transitioned its Google Translate codebase to a new system based on neural networks. If you use Google translate today you are using a kind of recurrent neural network called a LSTM, invented by Hochreiter and Schmidhuber.

Finally, DeepMind created a Go-playing algorithm called AlphaGo which defeated Lee Sedol, 18-time world champion, in March 2016. The algorithm is a reinforcement learning model based on policies which are deep neural networks.

II. Some history

“Thus logics and mathematics in the central nervous system, when viewed as languages, must structurally be essentially different from those languages to which our common experience refers… when we talk mathematics, we may be discussing a *secondary* language, built on the *primary* language truly used by the central nervous system.”

– John von Neumann, “The Computer and The Brain”, 1958.

In his book “The Computer and the Brain” von Neumann discusses how the brain is a massively parallel but relatively slow computer based on millions of interconnections between neurons, as compared to to our computing machines which are serial and very fast. He also raises the question shown on this slide, which is an antecedent of a debate within the field of artificial intelligence.

The debate is about the following question: if we want to create artificial intelligence, that is, automated reasoning which is as general and flexible as human reasoning, should we focus our engineering and scientific effort on creating an artificial analogue what von Neumann calls the *primary* language, or should we focus instead on the *secondary* language? These two approaches to AI are referred to respectively as *connectionist* and *symbolist*.

“It was John McCarthy who coined the term AI, apparently to distinguish his budding research agenda from that of Norbert Wiener. Wiener had coined “cybernetics” to refer to his own vision of intelligent systems — a vision that was closely tied to operations research, statistics, pattern recognition, information theory and control theory. McCarthy, on the other hand, emphasized the ties to logic. In an interesting reversal, it is Wiener’s intellectual agenda that has come to dominate in the current era, under the banner of McCarthy’s terminology.”

– Michael I. Jordan, 2018 (link)

In the early days AI was largely symbolist. The term “artificial intelligence” was coined in 1956 at the Dartmouth conference organised by Minsky, McCarthy, Shannon and Rochester and this conference is often considered as the starting point of the field. A lot of the early work and successes in AI were closer to the idea of emulating the symbolic layer of human reasoning, which is not surprising given how mature mathematical logic and related ideas had become. As it turned out, the connectionist approach couldn’t really compete until we had much more computational power and a lot more data, which didn’t arrive until the early 2000’s. Since 2012 the connectionist paradigm, under the name of deep learning, has been ascendant.

“At this point there enters an element which occurs repeatedly in the history of cybernetics - the influence of mathematical logic. If I were to choose a patron saint for cybernetics out of the history of science, I should have to choose Leibniz. The philosophy of Leibniz centers about two closely related concepts - that of a universal symbolism and that of a calculus of reasoning… It is therefore not in the least surprising that the same intellectual impulse which has led to the development of mathematical logic has at the same time led to the ideal or actual mechanization of processes of thought”

– Norbert Wiener, “Cybernetics: or Control and Communication in the Animal and the Machine” 1948

Norbert Wiener was a brilliant mathematician who together with Claude Shannon founded the field of information theory, and did important work in analysis. His influence on the field of AI is often underrated. I highly recommend Wiener’s books, not just Cybernetics but also “The human use of human beings” and “God and golem”. He thought very deeply about the influence of automation and AI on society, and wrote very clearly about it.

Algebra

Connectionist	**Symbolist**
Linear algebra	Boolean algebra
Calculus	Programs (e.g. lambda calculus)
Approximate inference	Inductive logic (Bayesianism)

Synthesis

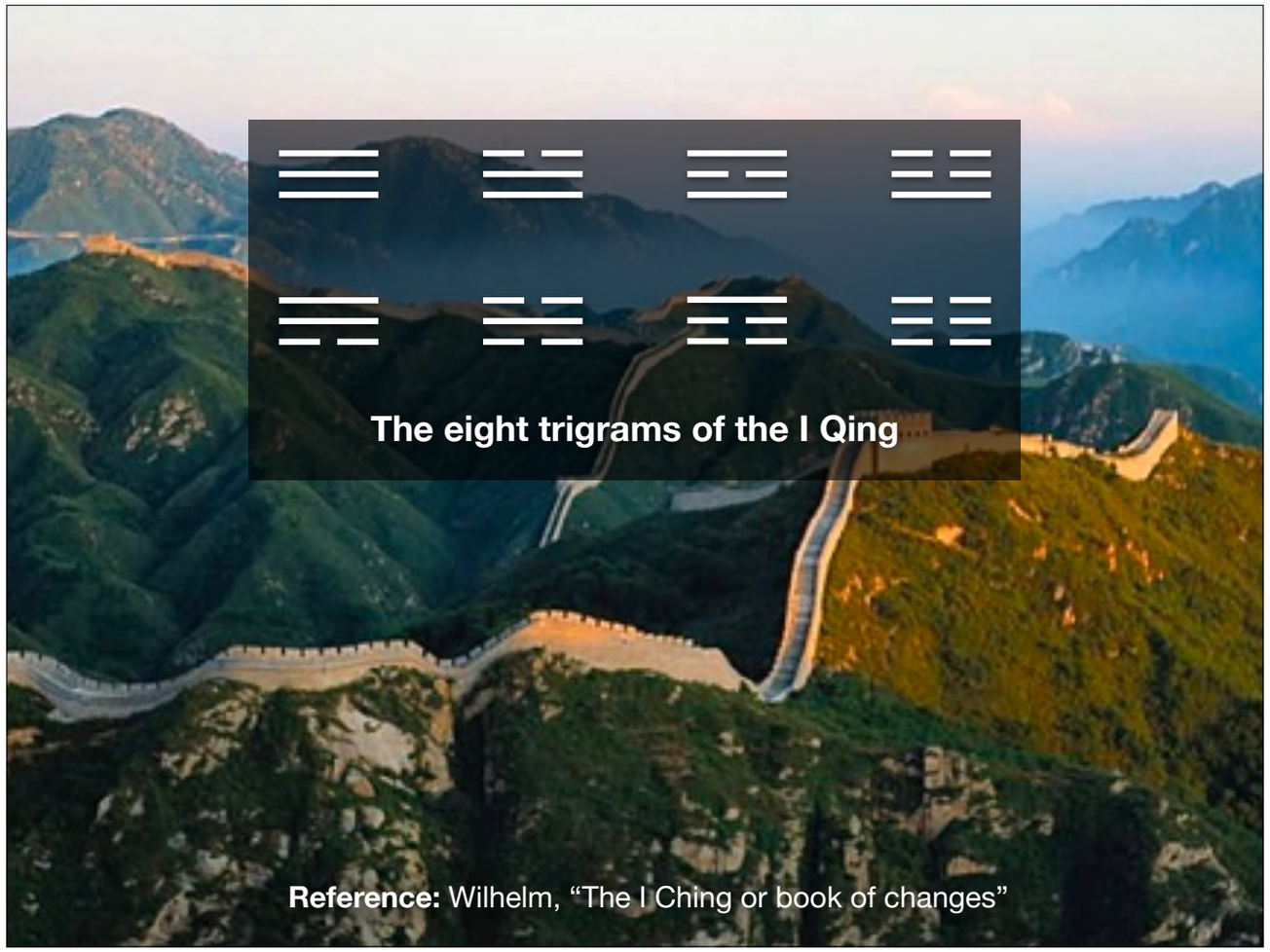
Uber: Pyro, Google: TensorFlow, Facebook: PyTorch

differentiable programming / probabilistic programming

(Tenenbaum talk)

III. Blind spots

Now we are going to change gears completely.



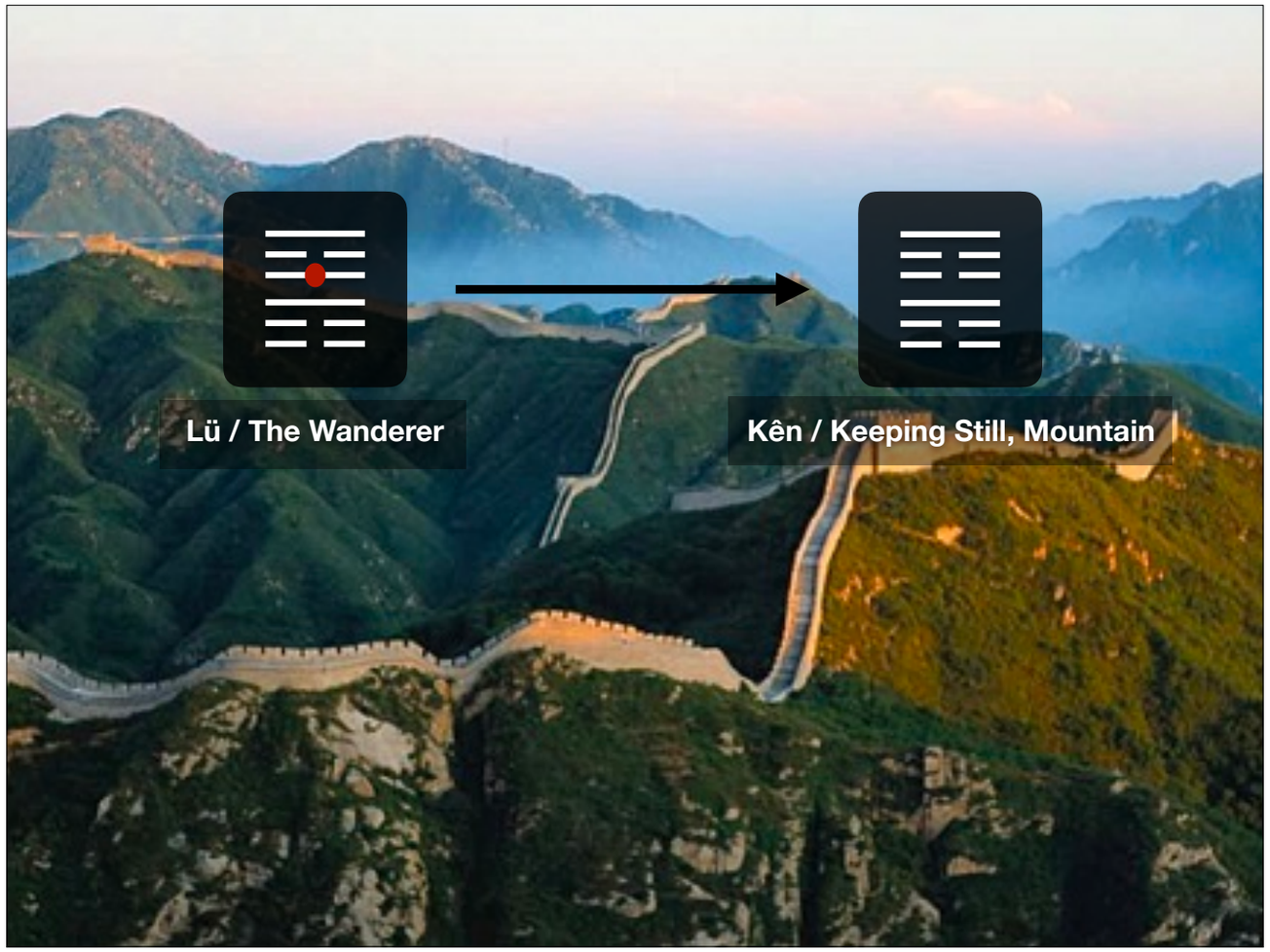
Reference: Wilhelm, “The I Ching or book of changes”

The I Qing, or Book of Changes, dates back to at least 600 BC. It is a foundational text in Chinese philosophy. To explain what the Book of Changes is, let us start by considering the simplest kind of Oracle, which delivers “Yes” or “No” answers to questions that are put to it. If we represent a “Yes” answer by a simple unbroken line, and a “No” answer by a broken line, then we may represent a series of three answers by a group of such lines. Such groups are called trigrams, and there are exactly eight of them.

The idea of the Book of Changes is very simple. It proposes that to identify these trigrams, with their assigned meanings, as the atoms of the world of ideas.

	Name	*Attribute*	*Image*	*Family Relationship*
☰ Ch'ien	the Creative	strong	heaven	father
☷ K'un	the Receptive	devoted, yielding	earth	mother
☳ Chên	the Arousing	inciting movement	thunder	first son
☵ K'an	the Abysmal	dangerous	water	second son
☶ Kên	Keeping Still	resting	mountain	third son
☴ Sun	the Gentle	penetrating	wind, wood	first daughter
☲ Li	the Clinging	light-giving	fire	second daughter
☱ Tui	the Joyous	joyful	lake	third daughter

Here are the meanings assigned to each of the eight trigrams. You'll note that each trigram bears several meanings, applicable to different kinds of questions. By combining two trigrams together to form what is called a hexagram, we can form more complex ideas



For example, here are two hexagrams. These have names and standard meanings, which I won’t elaborate on here. Perhaps what is most fascinating about the I Qing is that it has a dynamical aspect. Obviously if we begin with any hexagram and exchange a single broken line for a solid line, or vice versa, we get a different hexagram with its own meaning. The I Qing assigns narrative meaning to such pairs.

For example, in this case the hexagram Lu would be the starting point of a development leading, by reason of the situation associated with a variation in the third line, to the final situation i.e. the hexagram Ken. By a precise calculus of how some sequences of bits may transform into other sequences, the I Qing sets forth what is intended to be a universal symbolic system for understanding the world (p.336 Needham vol. 2).

“For even though some of the Western methods are different from our own, and may even be an improvement, there is little about them that is new. The principles of mathematics all derive from the *Book of Changes*, and the Western methods are Chinese in origin: this algebra springs from an Eastern word.”

– Emperor Kangxi 1654-1722

Reference: Perkins “Leibniz and China - a commerce in light” p.29
Spence, “Emperor of China”, p. 74.

While the I Qing is quite fascinating in some respects, we obviously cannot believe in it as a system of the world. As Joseph Needham explains in his masterwork about the history of science in China, the deep deference of Chinese intellectuals to the idea of the I Qing was partly responsible for them having a major blind spot when it came to Western mathematics and science. These advances were known to the elite in the Qing dynasty, but not taken sufficiently seriously until military incursions by European powers made the point in a different way.

Blind spot?

Which leads me to ask the following question. Do *we* have a blind spot?

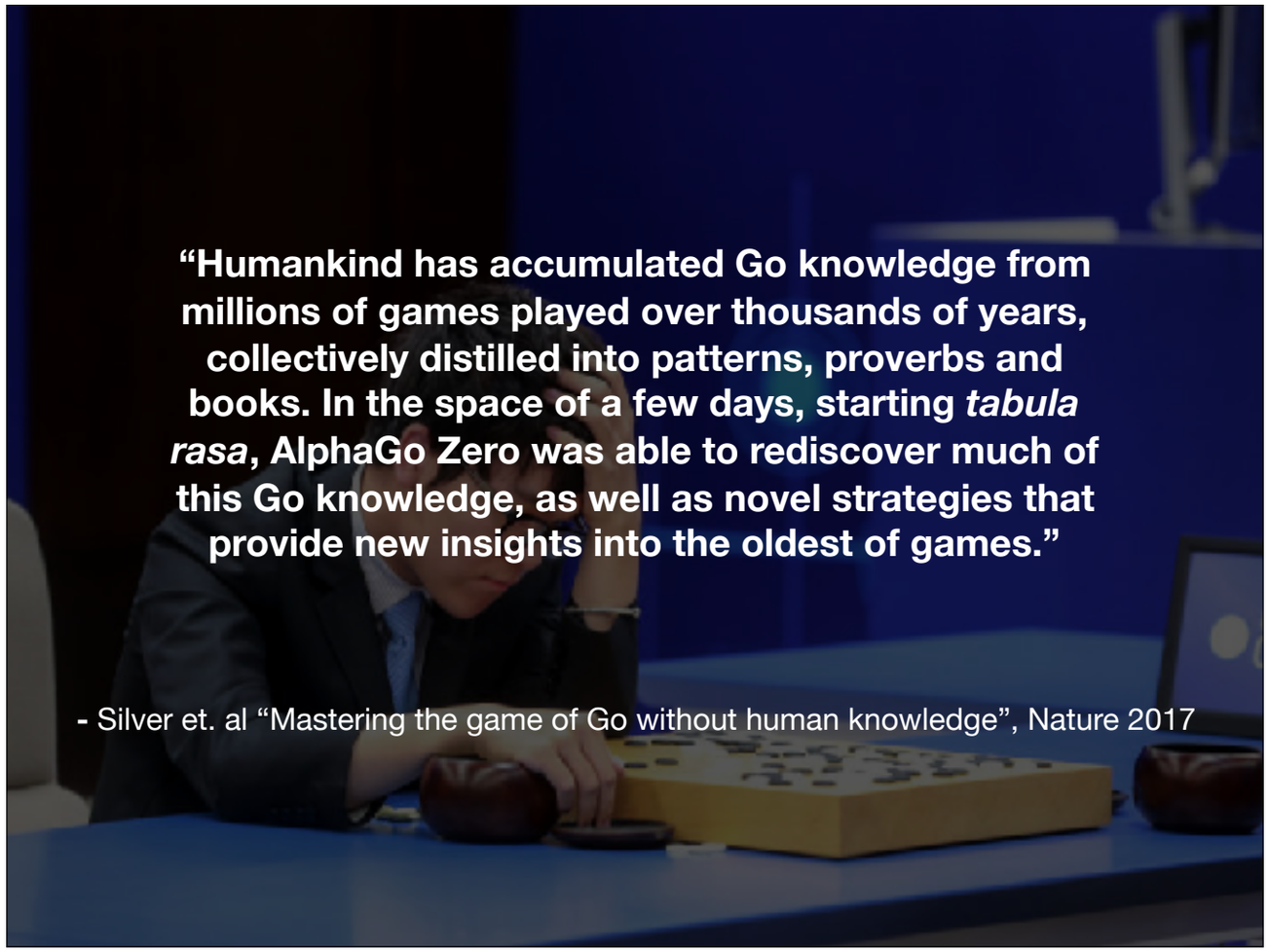
I think we can safely say we don't have *precisely the same* blind spot. Like the Chinese before the 20th century, our natural philosophy is also reliant on a universal symbolic system, namely mathematics. But we have proofs, and our predictions about the world are tested by experiment, so I'm pretty sure we are not making the same kind of error they were with the I Qing. But could there perhaps be some other unexamined aspect of our habits of thought, not necessarily a flaw in the fundamental logic of mathematics, but rather in the way we think about it and use it and relate to it?

How would we know if we had a blind spot? Presumably because one day we will just run headlong into the brick wall of reality, and experience a rude awakening. I think AI is beginning to expose a blind spot of this kind, and to explain let's return to the present day, specifically to March 2016 in Seoul, Korea.

In March 2016 Lee Sedol, 18-time world Go champion, played a series of five matches against a deep reinforcement learning algorithm AlphaGo created by DeepMind. He won one out of the five games.

In June 2017 an updated version of AlphaGo, called AlphaGo zero, teed off in a three game match against Chinese Go prodigy and top-rated player Ke Jie at the “Future of Go” summit in Wuzhen, China. AlphaGo Zero won all three games.


“After my match against AlphaGo, I fundamentally reconsidered the game, and now I can see that this reflection has helped me greatly. I hope all Go players can contemplate AlphaGo’s understanding of the game and style of thinking, all of which is deeply meaningful. Although I lost, I discovered that the possibilities of Go are immense…”
– Ke Jie, June 2017



The original AlphaGo was trained on 160,000 human games as part of the initial phase of creating its policy network. It was then refined by about a million rounds of self-play, using policy gradient reinforcement learning.

However the second version, AlphaGo Zero, which defeated Ke Jie in 2017 was not trained at all on human games. Its policy network was learned purely through self-play, starting only from the rules of Go and some modest observations about the geometry of the game board (for example, symmetries of the rules under rotation). This is a profound distinction to the original AlphaGo.

I have to admit, the Ke Jie match left me a bit shaken, and I know other mathematicians who had the same reaction. But I wasn’t sure at first precisely why I was shaken. It’s not just that machines beat people at Go today, and maybe tomorrow they will be better mathematicians than people too. I know that’s not the case, because I had the reaction to AlphaZero but not so much to AlphaGo. I could feel that there was something deeper than just a worry about asymptotic human inferiority, but I couldn’t articulate it at the time. Now that I’ve had a chance to think about it more, here’s what I think was behind my reaction.

The map is not the territory

- Beginning with a system of rules / axioms, we can
- Generate a **territory** of valid games / true propositions.
- This territory *equals Go* / *mathematics. Right?*
- No. Human mathematics is a conceptual narrative structure for enumerating certain paths in this territory. It is knowledge distilled into “patterns, proverbs and books”. Human mathematics is a **map**.

The AlphaZero victory exposed a blind spot in the worldview of Go players, and I realised that as a mathematician I had been living with exactly the same blind spot. To explain the blind spot, we have to examine what we mean by the “game of Go” or “mathematics”. We could say that we mean the set of rules, or a set of axioms, and the set of valid games or true propositions that follow from those rules or axioms. It is reasonable to think of this as a landscape or territory.

But this territory is not what we mean when we say “Go” or “mathematics”. What we mean is closer to “Human Go” or “Human mathematics”, and it is a historical narrative structure which enumerates in a human comprehensible way some part of this territory. Human Go, or human mathematics, is more like a _map_ than it is like _territory_.

The shock that I think Lee Sedol and Ke Jie had to face was the realisation that Human Go is not all of Go, and maybe it is not even the best parts. The territory was not the map. We haven’t had to really face this yet as mathematicians, but our time is coming too.

So what are we going to do about it?

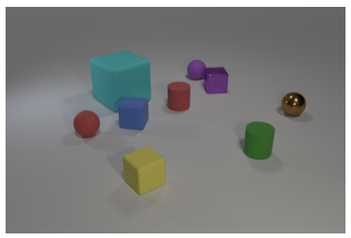
I want to leave this question open, and not answer it directly. But in order to end on a more optimistic note, I’d like to highlight one recent paper that I find interesting.

Making maps

DDRprog: A CLEVR Differentiable Dynamic Reasoning Programmer

Joseph Suarez **Justin Johnson** **Li Fei-Fei**
Department of Computer Science, Stanford University

- Image Index: 2364
- Question: what number of tiny brown shiny things are to the right of the matte sphere that is on the left side of the tiny red object to the right of the small yellow object ?
- Program (label): 1_filter_size_small 1_filter_color_yellow 1_unique 1_relate_right 1_filter_size_small 1_filter_color_red 1_unique 1_relate_left 1_filter_material_rubber 1_filter_shape_sphere 1_unique 1_relate_right 1_filter_size_small 1_filter_color_brown 1_filter_material_metal 1_count
- Answer (predicted, label): 1, 1



Fei-Fei Li was the leader of the original team which put together the ImageNet benchmark, which you will recall was what Kirzshevsky, Sutskever, and Hinton were applying deep learning to back in 2012, in the first big success of these methods. In addition to being a professor at Stanford, she's now chief AI scientist at Google's cloud division.

The ImageNet benchmark was about describing the contents of images. A harder problem is to reason about the contents of an image, and answer questions like the one shown here. In this paper Fei-Fei Li and collaborators show how to use a domain specific language as the target of a deep learning system. The deep learning system learns to synthesise programs in this language which when applied to pairs consisting of an image and a question, returns the answer to the question.

This is good, because in principle we can understand programs. We can't really understand the meaning of the set of weights in a neural network. These programs are a kind of communication between the humans and the machine: the humans specify a problem, the machine communicates its solution in the form of a program, and because the humans are good mathematicians they have invented all sorts of clever ways to reason about programs, and so they are able to trust the machine's solution.

Thanks for your attention.