Paper: AGI from the perspectives of categorical logic and algebraic geometry

YKY

August 13, 2024

Contents

My paper was not very well-written, nor easy to understand. That's not usually my style, but if I don't embellish the paper with some pretensions, it may not get published ☺ In this presentation I try to explain my ideas more easily:

I In the first part I try to explain how the theory of "**No Free Lunch**" and inductive biases is a basis for accelerating AGI (in particular the learning process). This is based on an informal argument which I have difficulty making rigorous.

II In the second part I explain **categorical logic** (as the structure of logic), in particular the notion of "fibrations" which describe predicate logics. But my conclusion is that fibrations are not very useful for accelerating learning, because it is actually isomorphic to the "boring" concatenation of tokens that we currently use in LLMs.

III The third part contains an idea that I discovered after submitting the paper, a kind of "**algebraic-logic**" neural network. Algebra, algebraic geometry, and polynomials are powerful tools in mathematics with a long history, much older than Curry-Howard isomorphism and category theory. They may offer more techniques to accelerate learning.

Part I

How does “No Free Lunch” guide us to accelerate AGI?

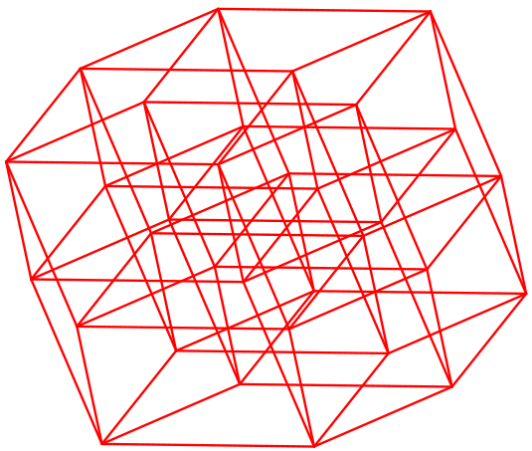
The basic argument

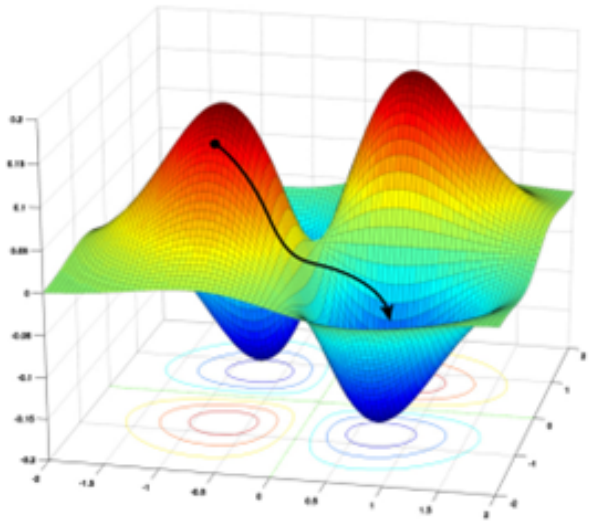
- We know very little about how **AGI solutions** are distributed in the hypothesis space (ie parameter space of neural networks), but exploring this space is very expensive (= training large models)
- We know with some certainty that at least one AGI solution exists in the structural **sub-space** conforming to formal symbolic logic (the kind that humans are familiar with).
- By a probabilistic argument based on "**ignorance**" [1], it seems reasonable to confine our search within this sub-space rather than in other "unknown" regions. But I'm not sure if this argument is sound, perhaps the reader can help validate or disprove it?

[1] similar to the argument we make to assign 1/6 to the rolling of a die

The hypothesis space, ie. the space of AGIs

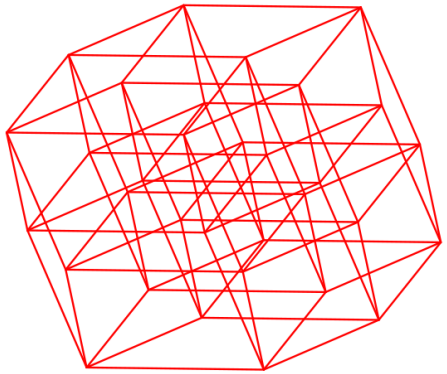
- In deep learning, our hypothesis space Θ = neural networks = parameter space = weight space = $\mathbb{R}^n$ or $[0,1]^n$ if truncated.
- It looks like this (left):

$\Theta =$ 



- Imagine a loss function as sitting above the hypercube; We seek to minimize it by **gradient descent**.
- When people visualize gradient descent, they tend to think of the above figure (right), but in practice, the "ups and downs" along one dimension is *dominated* (overshadowed) by the sheer number of dimensions.

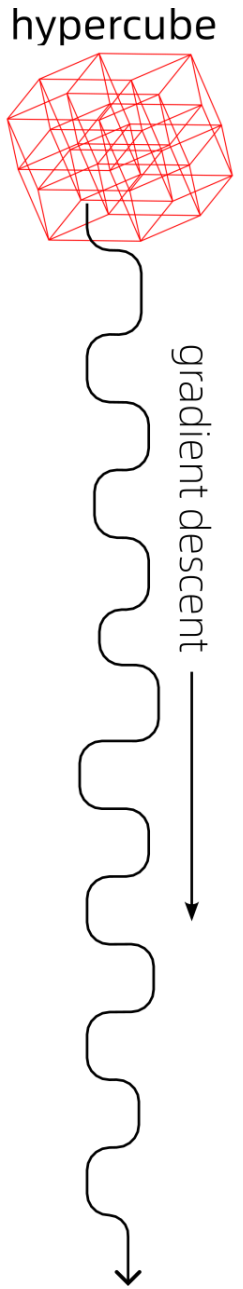
The distance metric in gradient descent

$\Theta =$ 

- Training large models is costly. We want to measure the cost of training neural networks by measuring the **size** of the hypothesis space Θ that we're searching.
- Gradient descent works by $\nabla_\Theta = \sum \frac{\partial}{\partial \theta_i}$. Each step is a "delta" move in the parameter space Θ.
- The distance metric we can use in the space Θ is the **Euclidean** one given by $ds^2 = d\theta_1^2 + d\theta_2^2 + ...$
- So the **area** of parameter space measured by Euclidean metric is an estimation of how long it may take to converge to an AGI solution.

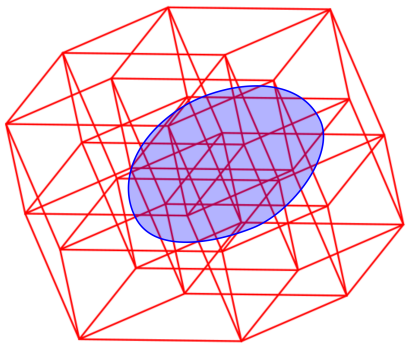
Unreasonable effectiveness of gradient descent

- Gradient descent can keep going down *for a long time* as there are millions of dimensions.
- The "unreasonable" effectiveness of gradient descent in deep learning is currently our most powerful weapon to solve the AGI problem.
  Its explanation is likely very difficult, as it hinges on the P =? NP problem.

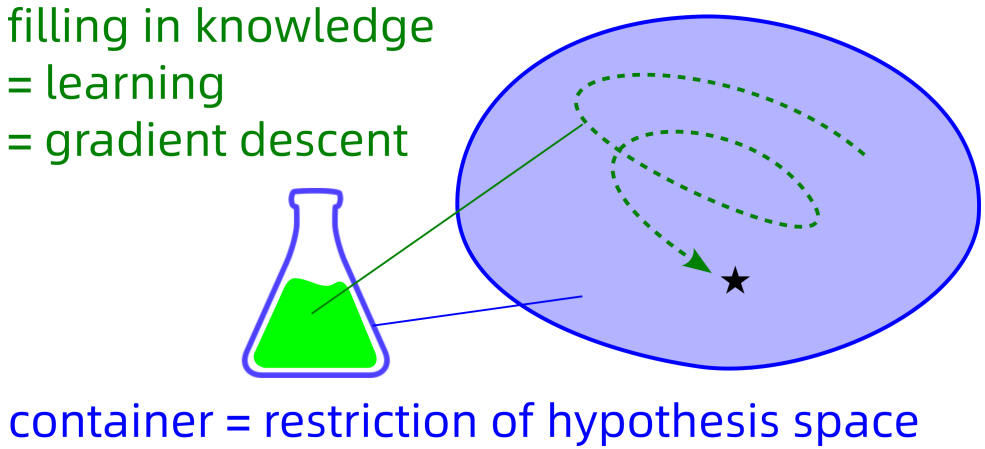


Finding an intelligent structure and filling it with intelligent knowledge

- Symmetry allows us to restrict searching to a particular **sub-space** of the parameter space:



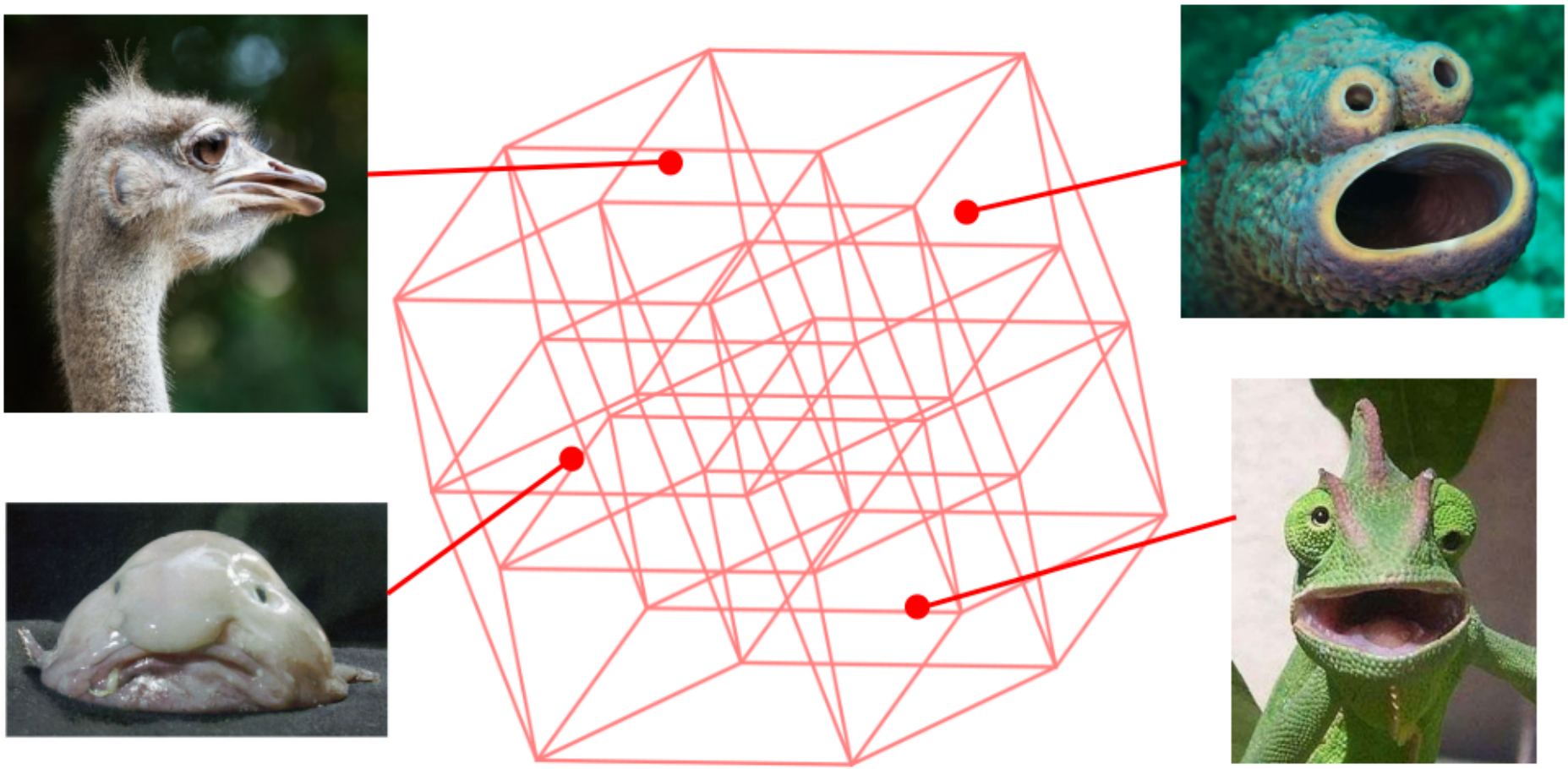
- After restricting to the right **structure**, we still need gradient descent to search for the right **knowledge**:



The parameter space represents both structure and knowledge.

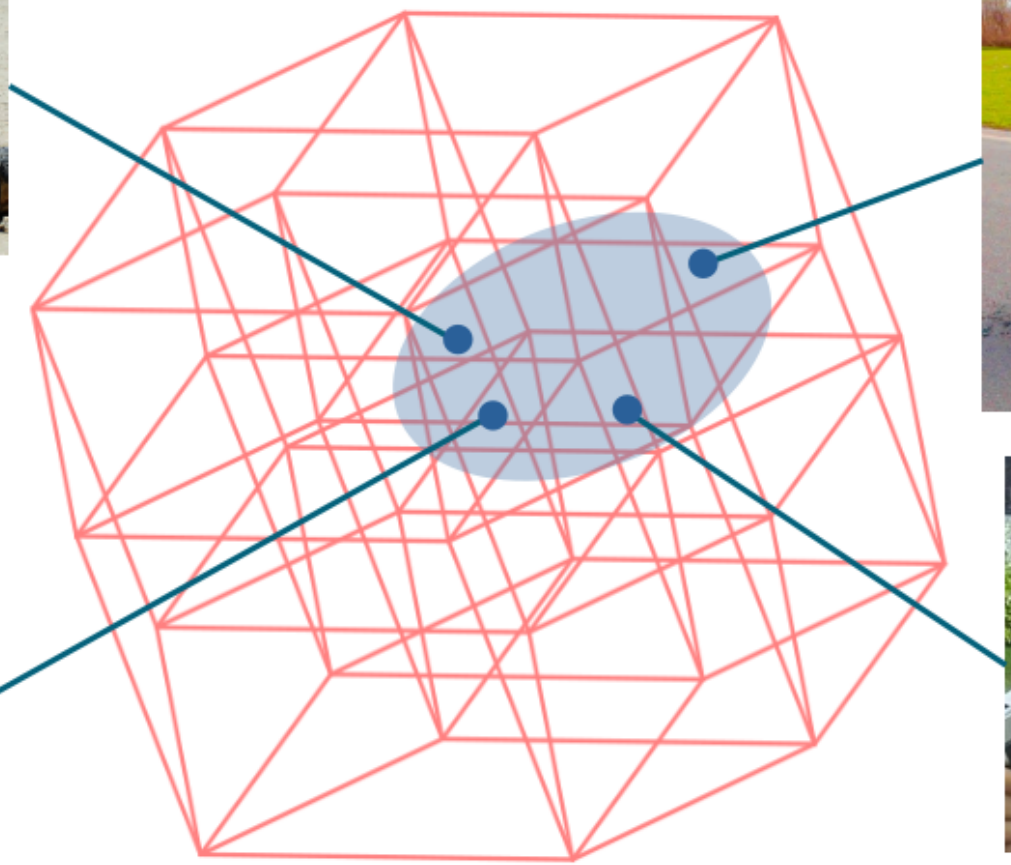
The hypothesis space has plenty of "dumb structures"

- It seems that dumb learning-machine structures (animals) are more common than smart ones



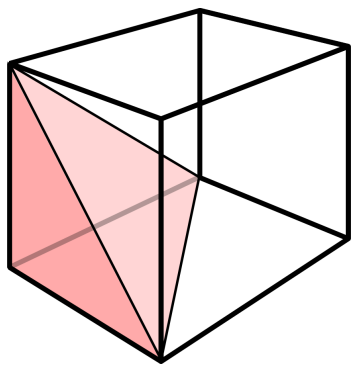
Even if we have found an intelligent structure, it may still contain lots of "dumb knowledge" in that sub-space

- It also seems that dumb knowledge is more common than smart knowledge



Motivating example: symmetric neural networks

- Now we turn to an artificial example where the No Free Lunch strategy can be proven to work.
- Permutation symmetry or commutativity ($ab = ba$) is the best-known symmetry in mathematics
- The input space, as a hypercube, is symmetric under exchange of vertices. The **fundamental domain** of this symmetry is one *corner* of the hypercube:



(Note that this hypercube is the *input* space, not the mapping / hypothesis space)

- Building a symmetry into a neural network may seem a daunting task, but recent research shows that merely **projecting** all data points to the fundamental domain has the same effect as achieving symmetry [2].

Motivating example: symmetric neural networks (2)

- If we restrict to binary logic, the total number of Boolean functions in n variables is 2^{2^n} whereas the number of **symmetric** Boolean functions is 2^{n+1}. So we clearly see an *exponential* reduction in the size of the hypothesis space.
- Tantalizingly, the **Transformer** also has this permutation symmetry, and researchers have proposed that Transformers are capable of performing *syntactic* manipulations similar to those in proofs of symbolic / predicate logic.
- We can further relax the restriction of binary logic to k-ary logic, ie, increase the number of lattice points in the hypercube, and the above analysis still holds qualitatively. In the limit, this seems to apply to **smooth functions** as well.

Part II

Categorical logic, and fibrations in particular

Curry-Howard isomorphism

Many readers are already familiar with this, so I'll just highlight some interesting points...

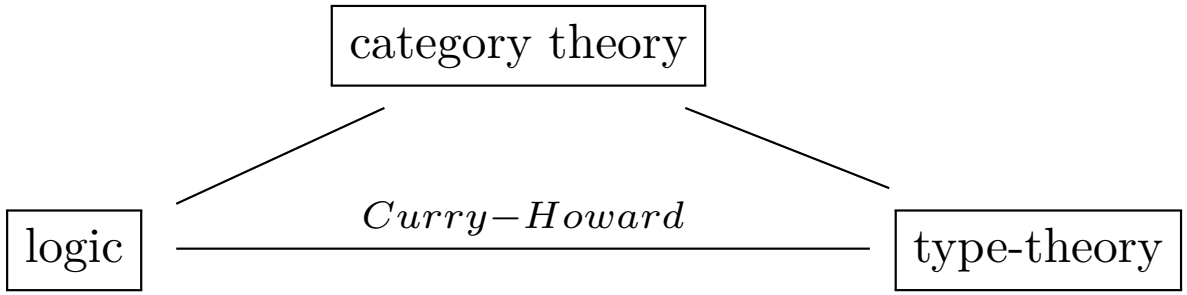
- The starting point of the Curry-Howard isomorphism is the following correspondence:

$$\boxed{\text{implication in logic}}\ A \to B \qquad A \xrightarrow{f} B\ \boxed{\text{function in type theory}}$$

- An example in type theory: "**currying**" means to convert a 2-argument function into a single-argument function which returns another function, ie. $f(x, y) = (g(x))(y)$.
  The types of f and g are equivalent, ie,
  $X \to (Y \to Z) \simeq (X \times Y) \to Z$.
  If we treat the above as logic it becomes:
  $X \to (Y \to Z) \equiv (X \wedge Y) \to Z$. One can verify this is a valid logic formula.
- This and other amazing coincidences led us to believe that CHI has some deep truth in it.

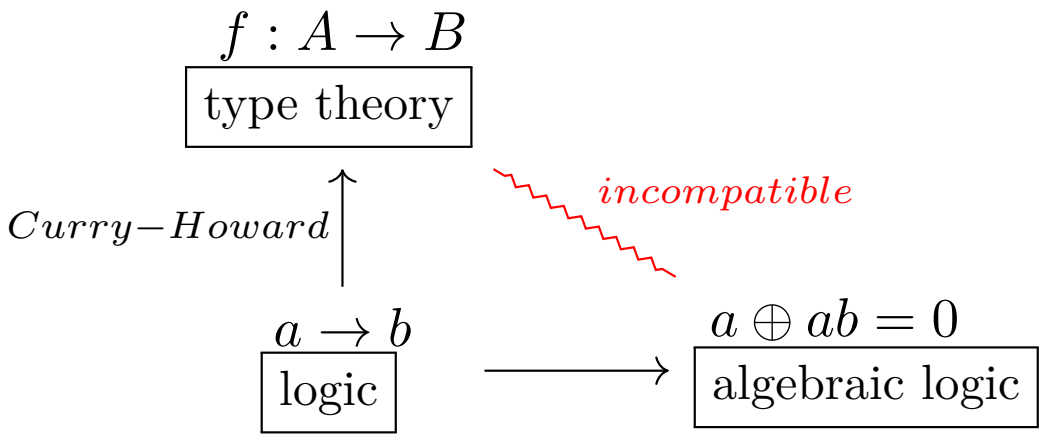
Categorical logic **clashes** with algebraic logic

- The late Joachim Lambek proposed a "trinity" between category theory, logic, and type theory [3]:



(1)

- But the Curry-Howard approach seems, at least on the surface, *incompatible* with the **algebraic logic** approach, for example:



- I have been working in the Curry-Howard direction because it feels more "modern" but lately I began to think the algebraic logic approach may have more potential for computational acceleration.

Definition of a **fiber bundle**

A **fiber bundle** is a tuple $\xi = (E, p, B, F)$:

(i) $E =$ **total space**

(ii) $B =$ **base space**

(iii) $F =$ a topological space called the **fiber** of ξ

(iv) $\begin{matrix} E \\ \downarrow p \\ B \end{matrix}$ is a continuous surjective map, called the **projection**

(v) for each point $b \in B$, the inverse image $p^{-1}(b) = F_b$, called the **fiber over** b, is homeomorphic to F

(vi) B has an opening covering $\{U_a\}_{a \in A}$ such that for each $a \in A$, there is a homeomorphism: $\psi_a : U_a \times F \to p^{-1}(U_a)$.
If F is a discrete space, then the structure $F \hookrightarrow E \xrightarrow{p} B$ is called a **covering** of B.

The condition (vi) is just a re-phrase of (v) in the form of open sets, similar to the "gluing" together of charts in differential manifolds. So the essential condition is (v).

The "boring" cross product

- It was an anti-climax when I realized that fibrations do not offer a symmetry that reduces the size of the hypothesis space.
- An example of fibration for predicate logic is a relation $r(a, b)$ sitting above two entities a, b.
- If we treat the relation predicate and the entities as homogeneous atoms, ie. $r, a, b \in A$, then the term's type is just $A \times A \times A$, though it looks like $\begin{array}{c} A \\ \downarrow \\ A \times A \end{array}$.
- Putting an atom above the other two does not change the fact that the whole type is isomorphic to $A \times A \times A$, and this Cartesian product of types is "boring" because it is basically the same as a sequence of "tokens" in LLM terminology:

  which we have been using all the time.
- If we do not endow the fibration with more structure (such as a structure group G, giving rise to a G-bundle), then we are left with the boring Cartesian product, which does not offer any reduction / acceleration.

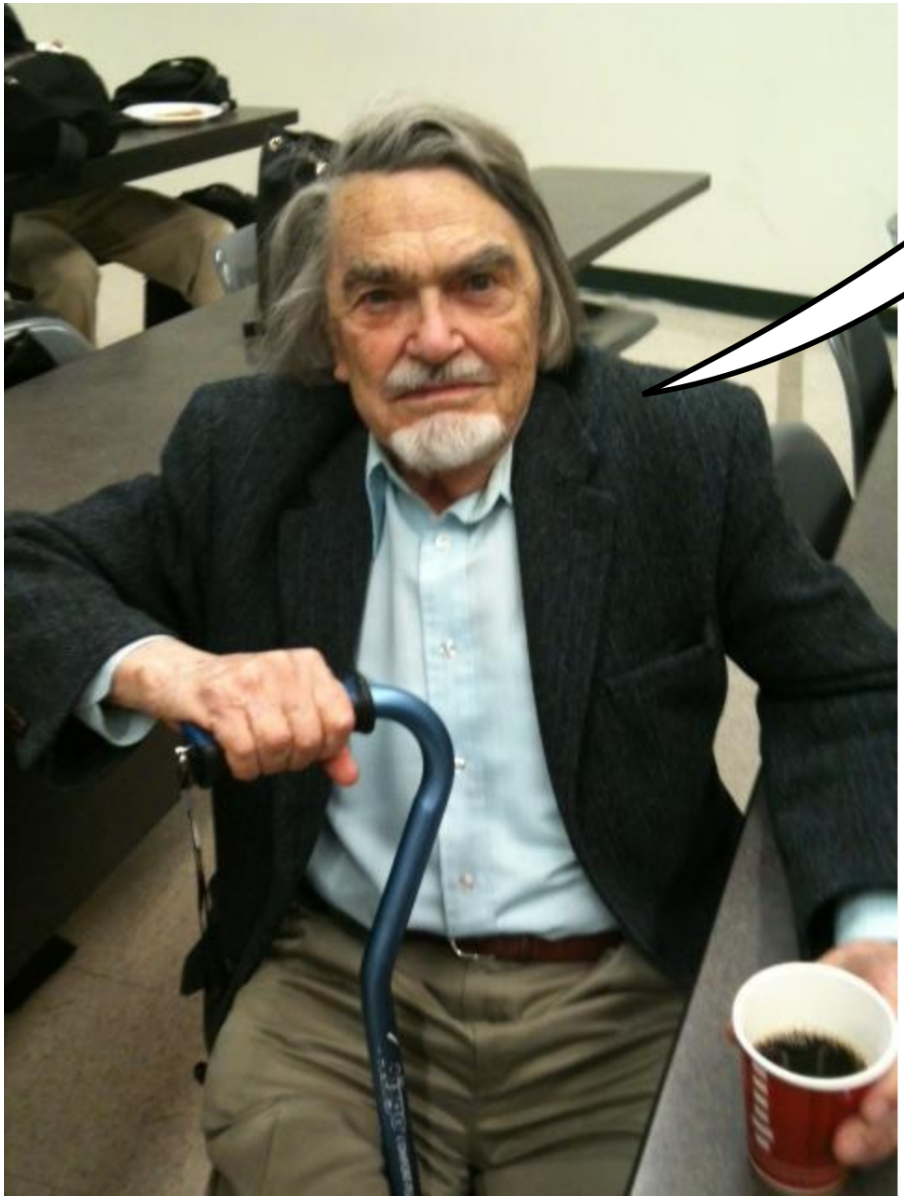
Part III

"Algebraic-logical" neural networks

Algebraic logic

- Traditionally, algebra (and algebraic geometry) is concerned with systems of **polynomials** over the "classical" number fields $\mathbb{R}$ or $\mathbb{C}$. The polynomials are built upon addition and multiplication of numbers, that we're all familiar with.
- Propositional logic $(B, \wedge, \vee, \neg, \top, \bot)$ can be equated with **Boolean rings** $(B, \cdot, \oplus, -(), 0, 1)$, with ring multiplication identified as $\wedge$ and ring addition identified as symmetric difference ("XOR").
- As predicate logic involves more operators (eg. $\forall$ and $\exists$), the classical algebra of numbers seems inadequate to represent them.
- For example, **cylindric algebra**[2] has additional operations c_i (cylindrification) and Id_{ij} (diagonal).
- Notice that algebra does not appear in the "trinity" (1). The Curry-Howard approach is not directly related to algebraic logic.

[2] see Wikipedia for a quick intro



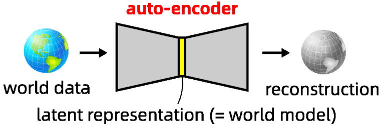
Algebraic logic, perhaps because of its failure to arrive at the foundations of mathematics quickly, poses many hard problems and attracts people who are good at solving problems with great ingenuity. Categorists, on the other hand, are more interested in making definitions than in solving problems.

Joachim Lambek

"Categorical versus algebraic logic" in Algebraic Logic, Budapest (Hungary), 1988

Neural networks with polynomial activation functions

- This is the basic setup of an auto-encoder which is the common setting for all current LLM models:

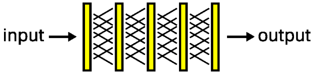


- The predictor / model can be construed as a logical **knowledge-base** (KB) making inferences:

$$\boxed{\text{input state}}\ \Gamma \ \xmapsto{\ \text{KB}\ }\ \Delta\ \boxed{\text{new conclusion}}$$

$$\Gamma \xrightarrow[\text{LLM / GPT / etc}]{\text{deep NN}} \Delta$$

- A neural network is composed of multiple layers:



- We can replace the activation functions with **polynomials**, creating "**algebraic**" neural networks. Pushing **Curry-Howard** to its extreme, perhaps each layer of algebraic NN can be identified as a logical KB?

One layer of neural network as a polynomial function

- Andréka-Németi [1] developed a **general framework** for studying algebraic logic, based on model-theoretic semantics. A logic is $\langle F, M, mng, \vdash, \Vdash \rangle$ where F = set of formulas, M = class of models or possible worlds, mng = meaning function: $F \times M \to$ Sets, $\vdash$ = syntactic provability, $\Vdash$ = semantic consequence, usually defined via mng.
- For example, in first-order logic, $mng(\psi)$ = set of all **evaluations** of variables $\vec{x}$ such that the formula $\psi(\vec{x})$ is satisfied in $\mathfrak{M} \in M$.
- For our purpose, let's start with a simple logic formula: $\forall x.P(x) \to Q(x)$. It can be implemented in **vector-matrix form**:

$$\begin{matrix} \sigma \\ \sigma \end{matrix} \begin{pmatrix} P \\ x \end{pmatrix} \begin{pmatrix} M \\ \mathrm{Id} \end{pmatrix} = \begin{pmatrix} Q \\ x \end{pmatrix}$$

  where σ = element-wise activation function (polynomial), M = transformation matrix.
- This is a bit tricky because our algebraic logic include terms that appear as input-output objects, as well as **maps** between such objects. But the Andréka-Németi framework is still applicable in essence.

How does it help to accelerate learning?

- I just newly discovered this and need more time to explore.
- What we have here is quite interesting: on the one hand, a deep neural network capable of **gradient descent**; on the other hand it is also an algebraic logic.
- Our goal remains the same: to accelerate learning.

References

Thanks for watching ☺️

[1] H. Andréka, I. Németi, and I. Sain.
Universal Algebraic Logic: Dedicated to the Unity of Science.
Studies in Universal Logic. Springer Basel, 2021.

[2] Aslan, Platt, and Sheard. "Group invariant machine learning by fundamental domain projections". In:
Proceedings of Machine Learning Research 197 (2023),
pp. 181–218.

[3] Lambek and Scott. Introduction to higher order categorical logic.
Cambridge, 1986.