Why ChatGPT cannot learn to reason deductively

Eric Johannesson

Department of Philosophy
Ashoka University

March 27, 2023

Deductive reasoning

What does it mean to be able to reason deductively?

Roughly:

Definition

To be able to reason deductively is to be able to answer questions of the following type:

Suppose that $P_1, \ldots, P_n$. Does it follows that C?

Clearly, this is not an all-or-nothing matter!

But the more questions you can answer of that type, the better you are at deductive reasoning.

An example

Example

Suppose that Anna is looking at Bob, and that Bob is looking at Carol. Furthermore, suppose that Anna is happy, and that Carol is not happy. Does it follow that someone who is happy is looking at someone who is not happy?

Solution.

Yes, it does follow. Bob is either happy or not happy. If he is happy, then someone who is happy (Bob) is looking at someone who is not (Carol). If he is not happy, then someone who is happy (Anna) is looking at someone who is not (Bob). In either case, someone who is happy is looking at someone who is not. □

What does ChatGPT say?

New chat

Suppose that Anna is looking at Bob, and that Bob is looking at Carol. Furthermore, suppose that Anna is happy, and that Carol is not happy. Does it follow that someone who is happy is looking at someone who is not happy?

No, it does not follow that someone who is happy is looking at someone who is not happy.

In this scenario, Anna is happy and looking at Bob, who is looking at Carol, who is not happy. However, we don't know if Bob is happy or not. It's possible that Bob is also happy, which means that someone who is happy (Bob) is looking at someone who is not happy (Carol).

Therefore, we cannot make a definitive conclusion about whether someone who is happy is looking at someone who is not happy based on the information given.

Thinking fast and slow

Kahneman: humans use two modes of reasoning, one which is fast and instinctive, and one which is slow and deliberative.

ChatGPT is like someone who can only think fast.

But some logical problems require more time, even for an extremely good deductive reasoner.

The existence of such problems essentially follows from the so-called *time hierarchy theorem* in computational complexity theory.

Decidable and undecidable problems

Let Σ^* be the set of all finite strings of symbols taken from some finite alphabet Σ.

Definition (Decidability)

A set $S \subseteq \Sigma^*$ is *decidable* just in case there is a Turing machine M that *decides* S, in the following sense: for any $x \in \Sigma^*$,

- if $x \in S$, then M *accepts* x (outputs 'yes' on input x), and
- if $x \notin S$, then M *rejects* x (outputs 'no' on input x).

Theorem (The undecidability of first-order logic)

Suppose that Σ^* contains all sentences of some first-order language that has at least one binary predicate. The set $S \subseteq \Sigma^*$ of logical truths in this language is not decidable.

The undecidability of first-order logic

C follows from $P_1, \ldots, P_n$ just in case $P_1 \wedge \ldots \wedge P_n \rightarrow C$ is a logial truth.

Hence, there is no chatbot such that, for any first-order sentences $P_1, ..., P_n$ and C, when asked whether C follows from $P_1, \ldots, P_n$, the chatbot answers

- 'yes' if C follows from $P_1, \ldots, P_n$, and
- 'no' if C does not follow from $P_1, \ldots, P_n$.

So, it would be unfair to accuse ChatGPT of not being able to perform this impossible task!

Computational problems as logical problems

We say that a set $R \subseteq \mathbb{N}$ is decidable just in case there is a Turing machine that, when given a sequence representing a number n as input,

- outputs a sequence representing 1 if $n \in R$, and
- outputs a sequence representing 0 if $n \notin R$.

Theorem (Representability)

There is a finite first-order theory Q of arithmetic such that, for any decidable set $R \subseteq \mathbb{N}$, there is a formula $\varphi(x)$ such that, for any number n,

- $Q \vdash \varphi(\underline{n})$ *if* $n \in R$, *and*
- $Q \vdash \neg\varphi(\underline{n})$ *if* $n \notin R$.

where $\underline{n}$ is the numeral corresponding to n.

Learning algorithms

For any characteristic function $c : \mathbb{N} \to \{0, 1\}$ and number n, a learning algorithm A takes a sequence $s = \langle c(0), \ldots, c(n) \rangle$ of answers as input, and outputs a description $A(s)$ of a Turing machine.

Definition (Success in the limit)

A learning algorithm A is *succesful in the limit* with respect to a set of characteristic functions $F \subseteq \{0, 1\}^{\mathbb{N}}$ just in case, for any $c \in F$, there is a number k such that $A(\langle c(0), \ldots, c(n) \rangle) = M_c$ for all $n \geq k$, where M_c is a description of a Turing machine computing c.

On the existence of successful learning algorithms

Theorem

A set $F \subseteq \{0,1\}^{\mathbb{N}}$ has an in the limit successful learning algorithm just in case there is a recursively enumerable set of (descriptions of) Turing machines that compute all and only functions in F.

Corollary

No learning algorithm is successful in the limit with respect to the set of all recursive characteristic functions.

But:

Corollary

There is a learning algorithm that is successful in the limit with respect to the set of all primitive recursive characteristic functions.

Slow AI

Let $F \subseteq \{0,1\}^{\mathbb{N}}$ be the set of all primitive recursive characteristic functions.

We construct a learning algorithm called *SlowAI* for F as follows. Let $M_0, M_1, M_2, \ldots$ be a recursive enumeration of descriptions of Turing machines computing all and only functions in F.

For any sequence of answers, let SlowAI return the first description of a Turing machine in the enumeration that is consistent with those answers.

With sufficient training, SlowAI can learn to decide any primitive recursive set.

As we shall see, ChatGPT cannot.

Time complexity

For each total function $f : \mathbb{N} \to \mathbb{N}$, we define the set $\mathsf{C}_f \subseteq \mathcal{P}(\mathbb{N})$ as follows: for any $R \subseteq \mathbb{N}$, we have $R \in \mathsf{C}_f$ just in case there is a Turing machine M deciding R such that, for any $n \in \mathbb{N}$, M returns an answer to input n in less than $f(n)$ steps.

A set $R \subseteq \mathbb{N}$ is said to be decided in

- **linear time** just in case $R \in \mathsf{C}_f$ for some linear function $f(n) = a + bn$.
- **polynomial time** just in case $R \in \mathsf{C}_f$ for some polynomial function $f(n) = a_0 + a_1 n + \ldots + a_k n^k$.
- **exponential time** just in case $R \in \mathsf{C}_f$ for some exponential function $f(n) = 2^{g(n)}$, where g is a polynomial function.

Fact: for every polynomial function f, there is k such that, for any $n \geq k$, we have $2^n > f(n)$.

Example: propositional logic

Any problem of propositional logic can be solved in exponential time: a problem of length n contains at most n atomic propositions, for which there are 2^n different interpretations (assignments of truth-values).

Evaluating a formula under an interpretation can be done in linear time (with respect to the length of the formula). Hence, there is an algorithm A and a number k such that, for any propositional logic problem of length n, we have that A solves the problem in less than $kn2^n$ steps.

It is not known whether problems of propositional logic can be solved in polynomial time.

This is (literally) a million dollar question in computer science!

A time hierarchy theorem for primitive recursive sets

For any function $f : \mathbb{N} \to \mathbb{N}$, let $\mathrm{PR}_f \subseteq \{0,1\}^{\mathbb{N}}$ be the set of characteristic functions computed by a Turing machine in less than $f(n)$ steps for all n.

Theorem

Let $f : \mathbb{N} \to \mathbb{N}$ be a primitive recursive time-constructible function, and define $g(n) = f(2n)^3 + 1$. Then we have $\mathrm{PR}_f \subset \mathrm{PR}_g$.

Corollary

There is a primitive recursive set $R \subseteq \mathbb{N}$ represented by a formula $\varphi_R(x)$ in Q such that some Turing machine decides whether $Q \vdash \varphi_R(n)$ in less than $2^{2n \cdot 3} + 1$ steps for all n, but no Turing machine decides whether $Q \vdash \varphi_R(n)$ in less than 2^n steps for all n.

The time complexity of ChatGPT

As far as I have been able to ascertain, for any dimension d, and for any input string of length $n \leq d$, ChatGPT runs in polynomial time as a function of d and n.[1]

Hence, ChatGPT cannot learn to decide sets that are only decidable in exponential time.

So there are infinitely many formulas $\varphi(x)$ such that SlowAI can learn to decide whether $Q \vdash \varphi(\underline{n})$ for any n, but ChatGPT cannot.

[1] It returns an answer in less than something like $a + bdn^2 + cnd^2$ steps, https://stackoverflow.com/questions/65703260/computational-complexity-of-self-attention-in-the-transformer-model

Caveats

- In a way, we already knew there would be decidable problems that ChatGPT must fail to decide, since it only takes inputs up to a certain length.
- Presumably, any physical computational device – or human, for that matter – can really only decide a finite set.
- There is a number k such that, for any finite set R of natural numbers, there is a Turing machine deciding whether $n \in R$ in less than kn steps for each natural number n.

  (But perhaps one can establish lower bounds on the length of the description of such a machine?)
- In any case, by the time complexity of ChatGPT alone, we cannot rule out the possibility that it solves every logical problem that humans can solve.

Caveats

Lastly, suppose we do not require ChatGPT to only answer 'yes' or 'no', but allow it to provide answers of arbitrary length. Furthermore, suppose that we choose to interpret an answer as 'yes' ('no') just in case the last token is 'yes' ('no').

In other words, we allow ChatGPT to "think out loud".

Then it may no longer be the case that ChatGPT always answers our questions in polynomial time (even though each token is produced in polynomial time).