Learning and logic

Paul Christiano Follow
Jan 29, 2016 · 15 min read

In most machine learning tasks, the learner maximizes a concrete, empirical performance measure: in supervised learning the learner maximizes its classification accuracy, in reinforcement learning the learner maximizes its reward. In order to maximize this reward, the learner has to be able to observe or compute it.

But sometimes we want our learner to discover some interesting fact about the world—e.g. to find the mass of the Higgs boson—and we have no external check to tell us whether it has succeeded.

Solving problems where we can't tie success directly to observations seems quite difficult at the moment. And we can't just throw bigger computers and more data at them without doing a bunch of *ad hoc* thinking or finding some new insight. So, relatively speaking, these tasks seem to be getting harder over time.

From an AI control perspective this is an important problem. In the long run, we really want to use machines for tasks where we can't define success as a simple function of observations.

Where logic comes in

Reasoning about logic is one of the simplest possible examples of this challenge. Logic lets us state a goal very precisely, in a very simple language with very simple semantics. Yet, for now, I don't think that we have effective techniques for pursuing symbolically defined goals.

The challenge

As a toy example, consider a program f that is simple to define but prohibitively difficult to evaluate. f takes two arguments, and outputs either 0 or 1. Given an input x, we would like to find an input y such that $f(x, y) = 1$.

This problem is very closely related to estimating the probabilities of the form "$f(x, y) = 1$."

If f is easy to evaluate, then we can treat this as a standard reinforcement learning problem. But as f gets more complicated, this becomes impractical, and we need some new technique.

I don’t know any reasonable algorithm for this problem.

Note that the *goal* is entirely logical, but the process of solving it won’t generally be. Even an agent with a simple logical goal can benefit from having lots of data about the world. For example, you might learn to use a calculator to solve arithmetic problems.

Standards

What do we mean by “reasonable algorithm”?

I’m interested in *frameworks* for symbolic reasoning, that combine available building blocks to solve logical problems. The goal is a framework that can effectively exploit continuing improvements in optimization algorithms, or increased hardware, or conceptual advances AI.

Some desiderata:

- Whatever level of performance on logical tasks we can achieve implicitly in the process of solving RL or supervised learning problems, we ought to be able to achieve similar performance on the logical problems themselves. For example, if our reinforcement learner can form a plan in an environment, then our logical reasoner ought to be able to solve an analogous constraint satisfaction problem. If our reinforcement learner can argue persuasively that a theorem is true in order to win a reward, then our logical reasoner ought to be able to assign high probability to it.
- Similarly, if we can achieve human-level performance on all RL problems, including complex problems requiring the full range of human abilities, we ought to be able to compute probabilities that are as accurate as those assigned by a human.

These standards are very imprecise (e.g. what does it mean for an RL problem to “implicitly” require solving some logical task?), but hopefully it gives a sense of what I am after.

I think that we can’t meet this requirement yet, certainly not in a way that will continue to hold as underlying optimization algorithms and computational hardware improve. (See the next section on inadequate approaches.)

Why logic is especially interesting

Logic isn't just the simplest toy example; it is also an extremely expressive language. With enough additional work I think that we might be able to define a reasonable proxy for our actual preferences as a logical expression. (Though like most people I expect it will be practically easier to use a language that can easily represent things like "the user," which are kind of a mouthful in formal logic.) The problem is "merely" that the logical definition is hopelessly complex.

Whether or not you buy this particular argument, I think that much of the "hard part" of reasoning symbolically already appears in the context of reasoning about very complex logical expressions. Thinking about logic simplifies the general problem of symbolic reasoning, by providing us with semantics "for free." But I think we are still left with a very important problem.

Some inadequate responses to the challenge

Logic as a representation

I can already build a theorem-proving system, that analyzes a sentence φ by searching for proofs of φ. I can maybe even up the ante by assigning probabilities to sets of sentences, and defining procedures for updating these probabilities on "logical observations."

These approaches lag radically behind the current state of the art for supervised learning.

One basic problem is that logic is the language of our problem statement, and logical deduction is indeed powerful, but it is often a *terrible* internal representation. For example, if I am told some facts about a linear order on X, Y, Z, I should probably represent those facts by putting X, Y, Z on a line rather than by explicitly representing every inequality.

We would really like to design algorithms that can efficiently learn whatever internal representation is most effective. Similarly, we'd like to allow our algorithms to learn what approach to logical inference is most appropriate. And in general, approaches which embed logical structure via hand-coded rules (and then lean on those rules to actually do meaningful computational work) look like they may be on the wrong side of the history.

Moreover, if we are searching for a scalable framework, these approaches obviously won't cut if. At best we will end up with a "race" between algorithms for logical reasoning and other AI systems.

(Note that MIRI's conventional research program basically involves running such a race. Most researchers at MIRI don't share my profound skepticism about their prospects.)

Transfer learning

A second approach is to treat logical reasoning as a supervised learning problem. That is, we can sample sentences φ, ask our learner to guess whether they are true, and then adjust the model to assign higher probability to the correct guess (e.g. to maximize log score).

The problem with this approach is that we can only train on sentences φ which are sufficiently simple that we can actually tell whether they are true or false.

In order to apply the learned model to complex sentences, we need to rely on a strong form of transfer learning. Namely, we need to take a model which has had *zero* training on sentences that are too-complex-to-evaluate, and trust it to perform well on such sentences. I am somewhat skeptical about expecting learning algorithms to reliably generalize to a new domain where it is impossible to even tell whether they are generalizing correctly.

Ideally we would be able to train our algorithm on exactly the kinds of sentences that we actually cared about. But this easy vs. hard distinction probably means that we would have to train our system exclusively on much easier toy samples.

I think that this kind of generalization is plausible for simple functions (e.g. multiplication). But assigning probabilities to logical sentences is definitely *not* a simple function; it draws on a wide range of cognitive capabilities, and the actual estimator is extremely complex and messy. I would be completely unsurprised to find that many models which perform well on easy-to-assess sentences have pathological behavior when extended to very challenging sentences.

At some point I might be convinced that AI control inherently needs to rest on assumptions about transfer learning—that we have no hope but to hope that learned functions generalize in the intended way to unforeseen situations. But I haven't yet given up—for now, I still think that we can solve the problem without any leaps of faith.

Pragmatically, if we wanted to train a function to estimate the truth of complex sentences, we might train it on our "best guesses" about the truth of complex sentences that we couldn't answer exactly. But we'll end up with a supervised learning system that estimates our best guesses about logical facts. This doesn't really buy us anything from a control perspective.

A preliminary approach

I'm going to describe an extremely preliminary approach to this problem. It seems far from satisfactory; my purpose is mostly to raise the question and show that we can get at least a little bit of traction on it.

The scheme

We'll train a function *P* to assign probabilities to logical sentences. For simplicity we'll work with a language that has constant, function, and relation symbols, variables, and no quantifiers. Variables are assumed to be universally quantified.

(I'm not really going to talk about how the function is trained or what class of models is used. I just want to use *P* as a black box for the online learning problem I'm going to describe. For concreteness you could imagine training a neural network to recursively build a constant-sized vector representation of formulas or terms by combining representations for each subexpression. Probably an algorithm which could actually handle this problem would need to advance the state of the art in several important ways.)

At the same time we will train a reinforcement learner *A* to produce "challenges" to *P*: *A*'s goal is to identify inconsistencies in *P*'s probabilities.

I'll also assume we have some *observations* φ_i, logical facts which are observed to be true. Over time the set of observations will grow.

I'll allow four kinds of challenges from *A*, corresponding to four kinds of possible inconsistencies.

1. Given any pair of sentences φ, ψ, a consistent assignment of probabilities to sentences should have: $P(\varphi) = P(\varphi \wedge \psi) + P(\varphi \wedge \neg\psi)$.

2. Given any sentence φ with a free variable x, and any term t, we should have $P(\varphi \wedge \varphi[x := t]) = P(\varphi)$.

3. Given any sentence φ and a sentence ψ which is “obviously” equivalent to on φ, we should have $P(\varphi) = P(\psi)$. I won’t define “obviously,” but we could use the notion of *trivial equivalence* from here.

4. Given any observation φ_i, we should have $P(\varphi_i) = 1$.

A’s goal is to produce a pair of sentences, or a sentence and a term, such that *P* violates one of these constraints.

It turns out that these constraints are universal: If *P* doesn’t violate any of these constraints, then we can prove that *P*’s assignments actually correspond to some distribution over models consistent with observation. In reality, *P* will never converge to a distribution that satisfies all of these constraints.

Formally, *A* and *P* play the following game:

- *A* chooses a consistency check from one of categories 1–4 above. We may put some limits on what sentences it can use in a consistency check—for example, to implement curriculum learning, we may initially limit *A* to providing short sentences.

- *P* assigns probabilities to each sentence referenced in the consistency check. (The same program is used to assign a probability to each sentence. Intuitively, separate copies of *P* independently assign a probability to each sentence.)

- If *P*’s probabilities are inconsistent, then we penalize *P* (and reward *A*). A natural choice for penalty is the total KL divergence from *P*’s probabilities to the closest consistent set of probabilities.

A is trained to maximize *P*’s penalty in the next round (i.e. without concern for effects on future rounds), and *P* is trained to minimize its penalty.

Example: Only observations

If *A* only ever makes challenges of type 4—enforcing consistency with an observation—then *P* is free to ignore logical structure. In this case, the procedure corresponds to supervised learning. So at least we have successfully subsumed the simple supervised learning approach.

All of the system's ability to reason about complex sentences is coming from the consistency checks.

The consistency mechanism is more general than the observations. For example, by carrying out the steps of a computation one by one, *A* can force *P* to be correct about the result of that computation. The observations are only relevant if either there are constant symbols in the language, or we are relying on the environment to do interesting computations.

So, even if we left out the observations, as long as *A* followed an appropriate strategy, this system would still subsume the simple supervised learning approach. (*A*'s strategy is obviously very important, see the section "Problem: Relevance" below.)

Example: No structure

P is free to ignore all structure of logical sentences, and only use the constraints implied by *A*'s challenges. For example, *P* could use the following procedure:

Notice that each constraint is linear, so that the set of constraints appearing *A*'s challenges form a polytope (which is simply the whole space [0, 1] in any coordinate that hasn't yet appeared in a constraint). *P* can track each of these constraints, and in each round output the appropriate coordinate of the centroid of this polytope.

(This basically looks like constraint generation, though it's not going to go anywhere good ever because *P* can never converge—see the next section.)

On this model, *P* and *A* together are essentially doing elementary logical inference. The whole definition of the system resides in *A*'s choices about what to explore, which is playing the role of the proposer in a proof search.

Problem: Relevance

There will always be inconsistencies in *P*'s probabilities, and *A* will always be able to find some of them. So *P* can never really win the game, and the training will continuously patch new problems identified by *A* rather than ever converging. Our only guarantee will be that *P* is consistent *for the kinds of questions that A prefers to ask*.

So it really matters that *A* asks relevant questions. But so far we haven't said anything about that, we have just given *A* the goal of identifying inconsistencies in *P*'s view. I think that this is the most important deficiency in the scheme—without correcting it, the whole thing is useless.

For simplicity, let's assume that we are ultimately interested in a particular sentence φ. We would like *A* to focus on questions that are most relevant to φ, such that if *P* is consistent on these sentences then it is especially likely to have a reasonable view about φ.

A crude approach is to simply reward *A* for asking questions which are correlated with φ (according to *P*). For example, when *P* is penalized on some sentence we can reward *A* according to the product [how much *P*'s beliefs about ψ had to move to be consistent] * [the mutual information between ψ and φ, according to *P*]. The hope is that questions which are relevant to φ will be correlated with φ, and so *A* will focus its attention on *P*'s most relevant errors. But there is no real principled reason to think that this would work.

Alternatively, we could train a relevance function *V* in parallel with *A* and *P*. The simplest instantiation of this idea might be to require *V*(φ) = 1, and to require that *V*(ψ) be large whenever ψ is logically related, or perhaps has high mutual information under *P*, to another sentence with a high relevance. (and to otherwise exert downward pressure on *V*). We could then reward *A* for choosing highly relevant sentences. This has a similar intuitive motivation, but it also lacks any principled justification.

Another crude measure is to reward *A* for identifying errors involving simple sentences, so that *P* will be roughly consistent whenever we talk about simple sentences, and it will "notice" any arguments that involve only simple sentences. This can't be a substitute for relevance though, since it requires *P* to notice *all* of these arguments rather than allowing it to focus on the relevant ones.

I don't see any easy way to deal with this problem. That may mean that this approach to logical reasoning is doomed. Or it just might mean that we need a clever idea—I think there is a lot to try.

Problem: going beyond logic

In this scheme, consistency checks are limited to logical consistency conditions. In some sense these conditions are universal if we only care about finite objects. But they may be less powerful than other kinds of inference.

Of course, *A* and *P* can learn strategies that reflect more complex regularities. For example, *P* can learn that probabilistic methods usually work, and thereafter use probabilistic methods to guess whether a sentence is true. And *A* can learn that probabilistic methods usually work, and thereafter use them to identify probable inconsistencies in *P*'s views.

But these other methods of inference can't be used to generate extra constraints on *P*'s beliefs, and that may mean that the resulting beliefs are less accurate than human beliefs (even if *P* is much better at reinforcement learning than a human).

It's not clear whether this is a big problem.

To see an example where this looks like it could be a problem, but actually isn't: consider an agent reasoning about arithmetic in without logical induction. Suppose that *P* assigns a high probability to $\forall n$: $\varphi(n) \rightarrow \varphi(n+1)$, and assigns a high probability to $\varphi(0)$, yet assigns a low probability to $\varphi(1000000)$. At face value, *A* has no way to prove that *P* is inconsistent. Thus *P* might be able to persist in these inconsistent beliefs, even if *P* and *A* are both good enough learners that they would be able to figure out the induction is useful.

But *A* can use induction in order to identify a problem in *P*'s beliefs, by doing a binary search to find a point where *P* has different beliefs about $\varphi(n)$ and $\varphi(n+1)$, even conditioned on $\forall n$: $\varphi(n) \rightarrow \varphi(n+1)$.

So in fact we didn't need to include induction as an inference rule for *A*, it falls naturally out of the rules of the game. (You might complain that the size of *P*'s inconsistency is significantly decreased, but a more clever strategy for *A* can ensure that the reduction is at-most-linear.)

It seems like this is probably a happy coincidence, distinctive to induction. In general, we can probably find axioms such that *P* doesn't lose anything by simply violating them. Even if *P* "can tell" that such an axiom is true, but has no incentive to assign it a high probability. Similarly, using that axiom *A* can identify likely inconsistencies in *P*'s beliefs, but has no way to quickly demonstrate an inconsistency.

And of course, *A* and *P* probably make judgments that don't correspond to any axioms at all, e.g. based on statistical regularities or alternative representations that give probabilistic clues.

In some sense, the problem is that we have a “dumb” arbiter, who only accepts proofs of inconsistency of a very restricted kind. If we want our system to learn to give human-level judgments, we need to either:

- Show that such a “dumb” arbiter is sufficient, and provides essentially the maximal possible pressure on *P*’s beliefs.
- Change the setup to make the arbiter’s job easier, so that a dumb arbiter can do it after all.
- Describe a more clever set of consistency checks which is in some sense universal and lets *A* make any kind of inconsistency argument that *A* might want to make. This is in some sense analogous to specifying *P* directly rather than using learning, but maybe the arbiter’s job is much easier than the learner’s job.
- Allow the arbiter itself to learn without compromising correctness.

A natural way to get around this problem is to use human evaluations to train an arbiter to evaluate consistency. This also allows us to give *P* a much larger class of questions (any questions that are meaningful to a human). In some sense this seems quite promising, but it introduces a few difficulties:

- From a safety perspective, if *A* is very powerful then we have reintroduced the kind of adversarial dynamic that symbolic reasoning may have let us avoid (since now *A* is incentivized to manipulate or deceive the human into judging in its favor). This might not be a critical failure; for example, a weak *A* and *P* can be used to build a more powerful human-aligned agent (which can then play the role of *A* or *P* in a still more powerful system, *etc.*)
- Practically, logic is convenient because consistency is all-or-nothing, and so we don’t have to worry about quantitatively weighing up different partial inconsistencies. Once we move to a more realistic domain, this becomes a critical issue. It looks quite challenging.

This problem is not as clear a deal-breaker as the issue with relevance discussed in the last section. But it seems like a more fundamental problem, and so maybe worth attacking first.

Related work

I am not aware of any existing work which tries to handle logical reasoning in what I'm calling a "scalable" way.

There is a literature on probabilistic logical reasoning, but mostly it fits in the category of "logic as a representation" above. This work mostly isn't aiming to build systems that scale as effectively supervised learning. The flavor of the work ends up being very different.

There is a much smaller literature applying machine learning to this kind of logical problem. What work there is has been very happy to focus on the supervised learning approach, explicitly restricting attention to "easy" sentences where we can easily compute the ground truth or the quality of a proposed solution.

One reason for the lack of practical work is that existing machine learning techniques aren't really strong enough for it to seem worthwhile. My guess is that the situation will change and is already starting to change, but for now there isn't too much.

Researchers at MIRI have thought about these questions at some length, and I thought about them a few years ago, but from a different angle (and with a different motivation). They have instead been focusing on finding improvements to existing intractable or impractical algorithms. Even in the infinite computing case we don't have especially good models of how to solve this problem.

I'm now approaching the problem from a different angle, with a focus on efficacy rather than developing a "clean" theory of reasoning under logical uncertainty, for a few reasons:

1. It's not clear to me there is any clean theory of reasoning under logical uncertainty, and we already have a mediocre theory. It's no longer obvious what additional theorems we want. This seems bad (though certainly not fatal).
2. It is pretty clear that there needs to be a more effective approach to symbolic reasoning, if it is to play any practical role in AI systems. So we know what the problem is.
3. The scalable symbolic reasoning problem looks much more important if AI control becomes a serious issue soon. Trying to solve it also looks like it will yield more useful information (in

particular, this is probably the main uncertainty about the role of logic in practical AI systems).

4. Given that we understand the constraints from efficacy, and don’t understand the constraints from having a clean theory, I think that thinking about efficacy is more likely to improve our thinking about the clean theory than vice versa.