Steps towards safe AI from online learning

Paul Christiano Follow
Apr 8, 2015 · 9 min read

Suppose that we have a good algorithm for episodic reinforcement learning, and would like to use it to build a safe AI. (A more precise problem statement.)

In this post I'll describe a very lightweight approach, whose success mostly rests on behavior of the human overseers. I'll describe some techniques that the overseers could use to get good behavior. In the next post, I'll discuss some problems with this approach.

Supervised learning

(ETA: this is a contextual bandits problem, not a supervised learning problem. It's much more like RL than supervised learning.)

Rather than working directly with an RL agent, we will consider a supervised online learner. This will simplify the presentation and make the discussion slightly more general.

In each round the learner is given a string $X \in \{0,1\}^n$, outputs a label $Y \in \{0,1\}^n$, and then receives a payoff $p(Y) \in [-1,1]$. The goal of the learner is to maximize their total payoff, for any fixed sequence of pairs X, *p*. It is easy to see that this is a special case of episodic reinforcement learning.

Let Predict be a good algorithm for this supervised learning problem, perhaps obtained by the reduction to episodic RL.

We'll leave the log L of past triples (X,Y,*p*(Y)) implicit, and write Predict(X') for the value that Predict outputs when it has seen the log L and the new input X'. We may also include other information in the log in order to help the algorithm perform well, even when the predictor is not being scored on that information. This could include e.g. information from sensors that Predict could use to build a good model of its environment.

Our mechanism

Conceptually our mechanism is very simple, and closely analogous to the definition of considered judgment from this previous post. Let Hugh be a designated human overseer for our AI.

To answer a question Q, we use Predict to find an answer which Hugh will score highly. Rather than making Hugh score answers on his own, we allow him to use Predict to recursively answer any sub-questions that arise. We require that the recursion depth be bounded (and limited by Hugh's patience). It is probably possible to remove this restriction with a more clever design.

Formally, we define Answer[N](Q) = Predict((N,Q)). Let M be some fixed, large integer. Then we define Answer(Q) = Answer[M](Q).

What is Predict predicting? Each time we call Answer(Q), we apply the following procedure with some non-negligible probability. If we do, we wait to return the answer until this procedure is finished.

1. Set Q* = Q, N* = M, A* = Answer[N*](Q*) = Answer(Q).
2. Print (Q*, A*).
3. If N* > 0, let Hugh adaptively query Answer[N*-1] at any number of questions Q1, Q2, …
4. Hugh outputs a score *p*(A*) ∈ [-1,1].
5. With probability 1/2, append ((N*,Q*), A*, *p*(A*)*)* to Predict's log.
6. Let Q* be one of the Qi chosen at random, let N* = N*-1, and let A* = Answer[N*](Q*). Go back to step 2.

We then build an AI that takes the action Answer("What should the AI do?"). We may also allow Hugh to give the robot instructions during step 3, for example he may decide to instruct the robot to take action Answer(Q) before he has finished evaluating Q.

Techniques

The behavior of Answer depends on the strategy employed by Hugh. Here are some techniques that Hugh can use to try to get good answers. All of these techniques fit within steps 3 and 4 of the mechanism above, though I will describe some of them in the (equivalent) language of consulting a "different" AI to help verify one AI's work.

Expensive counterfactual oversight

As described here: because Hugh only needs to provide oversight with small probability, he can do a much more thorough job than would otherwise be possible (while still not incurring a meaningful performance penalty).

This is a necessary precondition for most of the other techniques in this section, which would otherwise be unacceptably inefficient for most applications.

The concept of counterfactual oversight may seem somewhat arcane (and the rest of the techniques in this post therefore somewhat impractical). But we can also think of it as a robust form of training data. It may be that the actual counterfactual involvement is unnecessary, and that we can just follow the contemporary paradigm of training and then testing. Or perhaps some entirely different procedure is possible. But in any case, we shoul be able to find something that works at least as well as counterfactual oversight.

Verification

Some actions have immediately observable consequences, e.g. if someone builds a building, I can tell whether it serves my needs, whether it collapses, and so on.

In these cases, we can ask our AI to design a building, and after the fact decide how well it performed and assign a reward appropriately. In the procedure above, Hugh can allow a long time to elapse between receiving the proposed answer A* and actually replying with a score *p*.

In some cases no evaluation is feasible in the short term, but one will eventually become possible. For example, we may be able to evaluate a plan after we observe its longer-term consequences, or after we have become smarter or learned some important further facts. In principle this is not a problem; Hugh can just allow a *very* long time to elapse before making a judgment.

As the feedback is delayed further into the future it becomes particularly likely that a “good” plan will be one which compromises Hugh’s ability to evaluate it (for example by replacing him with a more agreeable rater). I’ll discuss this issue in more detail in a future post, but one basic precaution is to use a division of labor where one AI devises a plan and another AI helps Hugh ensure that the proposed plan is unlikely to compromise his ability to evaluate it. If

Hugh can determine that a plan is “safe” early, then he can safely wait a long time to determine whether it is “effective.”

Another simple precaution is to ensure that an AI making a long-term plan receives a near-maximal payoff by behaving honestly, but receives a very low payoff if we detect a manipulation attempt (see this talk for a discussion of this idea). This ensures that manipulation isn’t worthwhile unless it is very likely to succeed. The section **comparisons** below provides an observation that seems helpful for ensuring the payoff for honest behavior is high.

Bootstrapping

Verification on its own may often be prohibitively challenging. For example, imagine that we have commissioned some code and would like to confirm that it conforms to our specification. Even if we only do this confirmation with small probability, it may be intractable for humans to verify a code base that is too complex for them to have written themselves.

But this verification can itself be performed by an AI who will be rewarded for noticing possible problems (and perhaps who is able to inspect the entire cognitive process of the AI that produced the program, to help guess where a bug is most likely lurk). If the verification is performed only with small probability, then it can be performed much more carefully than the task itself (though this would involve a modification to the setup above, where Hugh can elect to use more powerful predictors for some queries).

Of course the work of this evaluator can itself be evaluated with the help of another AI. The key question is whether this bootstrapping ever “bottoms out,” with an AI whose behavior is sufficiently simple that a human can actually understand and evaluate it. This issue is discussed in these two posts; the other techniques on this list just need to be good enough that they let us evaluate each answer using a slightly less complicated process than the one that produced it.

Comparisons

Assigning an action a score between -1 and 1 may be challenging, since it involves an implicit comparison between that action and an unknown alternative. But Hugh can instead ask AI questions like “What alternative actions would have been better?” and then try to make pairwise comparisons between the action that was taken and the alternative that was proposed, giving a high reward iff the original seems better or at least nearly as good. We can make these comparisons in advance or after the fact, we need only make them with small probability, and we can make them with the assistance of the same AI that proposed the original action.

Arguments

More generally, most plans that are good seem to be good for comprehensible reasons.

That is, suppose an AI proposes an action which will be very hard to evaluate after the fact (or that an AI claims that a given plan is no good, or that a given program has a subtle bug, or so on). If we understood the consequences of this action, the reasons why it seemed good to the AI, the possible problems with the action, and so on, then it seems like we should typically be able to tell whether the proposal is good or bad.

The AI could provide an argument for its action, which we could evaluate to the best of our abilities. We could then ask the AI to find the plan that is supported by the most compelling argument, and take that one. But it seems much more effective to have an adversarial process in which another comparably intelligent AI helps us find holes.

Any techniques for arbitrating arguments can be applied to this situation. For examples, see past posts about arguments here and here. As described in those posts, a human arbiter can implicitly examine very large issues, by breaking them down into smaller questions and focusing on subquestions were the disputants disagree.

Reasoning about complex arguments may sometimes be harder than making a decision, and so an AI may be able to find a good plan without being able to justify it. But (1) we can structure arguments in such a way that intermediate claims are just empirical predictions, and such that "tell the truth" is a simple and effective rhetorical strategy, (2) we can carry out arguments only with very small probability, so it's fine if having arguments is merely slower or more expensive than making decisions, as long as it's possible at all, (3) we can make efforts to closely match the intellectual abilities of the two arguers (e.g. by using the same AI in both roles), so that differences in rhetorical ability are unlikely to have a big effect.

Verify process, not product

If a machine vision system tells me "This is a picture of a horse," it's not clear that they can give any compelling argument for their conclusion. In the case of images we could of course just look and see if we agree. But the problem is harder when machines are making analogous judgments that are beyond human abilities. For a simple analogy, imagine justifying “This is a picture of a horse” to a society composed entirely of blind people.

In this case it is possible to make a different sort of argument. Namely, the learner can claim “The parameters of <this> model were adjusted so that it reliably outputs ‘horse’ on pictures labeled ‘horse,’ and not pictures with other labels.” The learner can also justify each of: <this> model outputs ‘horse’ on this picture, the training images labeled ‘horse’ were really horses, the current image is drawn from a distribution sufficiently similar to the training data, and so on. Altogether, this constitues a compelling argument that the image in question probably depicts a horse, even to a society that has never before encountered a picture of anything.

Of course this argument could fail for many subtle reasons, but we can rely on a comparably sophisticated AI to identify possible problems in a proposed argument.

Note that we can potentially verify even very complex processes in this way (for example, processes that take billions of steps), by having one AI describe what each step is doing, and having another AI point out the aspects of this description that we are most likely to find problematic upon deeper investigation.

In the worst case, this amounts to simply solving the AI control problem on a case-by-case basis: rather than producing an AI which makes human-friendly decisions in general, we just need to find a way to certify that the process used to make a *particular* decision was human-friendly. And this certification can itself be assisted by another AI using a process which is nearly as complex as the original.

The unknown

This is a short list of approaches that Hugh could use to get good performance out of his AI. It’s clearly not a complete list, and Hugh would be at a significant advantage in coming up with new items once he knows more detail about the AI’s reasoning and has the opportunity to interact with it at length.

The main reason to have such a list now is to better understand whether these kinds of techniques are likely to be adequate, or whether a fundamentally different approach is needed.