Flattening debate between RL agents

Paul Christiano Follow
Apr 16, 2015 · 6 min read

(Status: exploratory, not very important.)

Let Alice be an effective episodic RL agent, and suppose that a human Hugh wants to use Alice's help to implement (an approximation of) Hugh's considered judgment.

I proposed a simple protocol in this post. In that protocol, we use a fixed recursion depth, and recurse on a random question at each level.

In a more recent post I suggest a big improvement: use another reinforcement learner, Bob, in order to choose which question to recurse on. Bob's goal is to make Alice do as poorly as possible. This protocol behaves much better if Alice makes errors, by ensuring that Alice learns from her errors. (If we recurse on a random question, we may happen to never provide feedback on any of the questions where Alice is wrong.)

(This post won't make much sense without the context from my previous posts.)

This proposal still leaves a lot to be desired. For example, if we want to have a recursion depth of N, then Hugh has to actually implement all N rounds of recursion. It is possible to remove this requirement when a powerful predictor is available. Can we do something analogous using RL agents?

This basically amounts to the question: can we reduce the number of rounds of interaction in an argument between two AIs?

A simple solution

The simplest solution is to allow Bob to accept Alice's answer, if it looks right, and to charge Bob a price for making an unsuccessful challenge. If Alice has already converged to a state where she is giving good answers, then this is enough to make the argument short—most of the time.

This approach seem good enough for most purposes, and in particular it makes the average time per argument low. But it has the disadvantage that the behavior is entirely determined by what Hugh "would have" done if there had been an extensive recursion. In some contexts this may be problematic.

For example, in a rapidly moving world, it may be extremely problematic for an AI system to pause for a week while Hugh implements a very extensive deliberative process. If doing so is merely costly, that's not a problem: the process is only actually carried out very rarely. But it may be that doing so is actually impossible, or extreme enough that it would substantially distort Hugh's judgment. For example, it may be that the world is changing very rapidly, and Hugh cannot guarantee his own safety for a week if a poorly-chosen set of AIs decides to wait on the outcome of an extensive process of deliberation.

If Hugh wants to pursue a much longer process of deliberation—for example, one whose "depth" is a few years—this problem becomes much more extreme. It would be nice to simulate these protocols without any dependence on what would actually happen if Hugh had to drop everything and reflect for years.

A remark on complexity theory

"Flat" proof systems, with few rounds of interaction between the verifier Hugh and the provers Alice and Bob, are generally not as powerful as proof systems which involve many rounds of interaction. (Unless the polynomial hierarchy collapses.) This holds even though the proof systems with "many rounds of interaction" never actually have to use more than one round if the two participants are honest.

But in this case our provers are software agents. This means that we can copy one agent and seperate the two parts, or equivalently ask them several questions and wipe their memory in between.

With this ability, the power of O(1) round interactive protocols is MIP = NEXP, a massively larger complexity class. Of course we won't really be able to solve problems in NEXP, since our AIs aren't that clever. But it does suggest that there is no complexity-theoretic limit on the AIs' ability to argue for any conclusion that they are capable of reaching.

Anyway, back to the real world.

A flatter protocol

Our protocol:

1. Bob chooses a question Q.
2. We give Alice input Q, and she provides answer A. We don't append (Q,A) to Alice's log.
3. Hugh evaluates Alice's answer, giving it a score $p \in [-1, +1]$. In the process, Hugh may ask Alice any number of questions, which she answers in exactly the same way that she answered Q (they also aren't added to her log). Critically, Alice doesn't "know" whether she is answering Hugh's questions, or Bob's.
4. We then append (Q, A, p) to Alice's log, i.e. we give Alice feedback of p for the answer A.
5. We then append (None, Q, $-p$) to Bob's log, i.e. we give Bob feedback of $-p$ for the question Q.

Under this scheme, Bob will do his best to find a question that will stump Alice. So if Alice manages to get high payoffs anyway, she would have gotten high payoffs for any other question that Bob could think of. In particular, she should also get high payoffs for each question posed by Hugh in step 3—assuming that Bob could also think of these questions.

After playing this game many times, Hugh has a good sense of how good Alice is at it. This makes it possible for him to interpret the results more sensibly.

Problems

Tricky questions

A basic problem with this game is that Bob can choose tricky questions that we just don't care about. If Alice answers these questions well, then that gives us some evidence that she can answer "real" questions well. But it may be that Bob can always find some clever question that trips up Alice, even if Hugh doesn't really care about Alice's ability to answer that question.

Hugh could try to address this by first checking whether Bob's question seems like something we care about; if it doesn't, Hugh can penalize Bob.

This introduces the risk Hugh doesn't understand what questions he should care about. There are a number of easy patches, though as usual it's hard to know how well they would work:

- Hugh could ask Bob to explain to us why we might care about the question. Of course, if Bob's answer is "because you'd need to know the answer to Q in order to evaluate Q*" then Hugh will need to know why he should care about Q*. This introduces a new problem with recursion, which we would have to address. Hopefully Bob could find a more direct explanation for relevance.
- We can start off being more conservative about what seems "relevant." As we see what kinds of questions come up when evaluating "relevant" questions, we can gradually expand our view of relevance. This would require some time for us to approach equilibrium.
- We can use a new reinforcement learner Charlie whose goal is to guess whether Hugh's question was generated by Hugh, or was generated by us in the process of evaluating a question generated by Hugh. This requires a number of tricks in order to be useful, but it might help when combined with the earlier ideas.

Hugh could also just deal with this concern, and know that Bob's questions are especially tricky.

Bob's questions don't "look right"

If Alice can tell the difference between Hugh's questions and Bob's questions, then Alice might answer Bob's questions well while answering Hugh's questions poorly. In this case, Bob will try to make his questions look like Hugh's, but he may fail.

If Bob is as smart as Alice, then we might hope that Bob can imitate Hugh well enough that Alice can't tell the difference. But that's not how it works: identifying an imitation is generally easier than making one.

We could try to address this by having Hugh help Bob with the question-generation step. For example, Bob ask Hugh: "If you wanted to know the population of Spain, how would you phrase the question?" Or they could work collaboratively in other ways to help find hard questions. For example, Bob could pose a question Q* to Alice, get Alice's answer A*, ask Hugh to evaluate Alice's answer, and

then let Q be the hardest question that Hugh asked in the process of evaluating A*.

Likewise, when Hugh actually asks questions, he could use Bob's help to try to make them look like Bob's questions. In this case we would need to introduce some additional incentives for Bob to be helpful, but that seems straightforward. The hope is that Hugh and Bob can somehow meet halfway, since both of them are incentivized to do so.

Overall this kind of solution seems kind of plausible, but unsatisfying and not that promising. I suspect this problem is really a sign that this is fundamentally not the right approach.

Conclusion

I think there is probably a lot of room to develop better mechanisms for running "arguments" between competing AIs, and that the concept of argument is much deeper than it at first appears. This post explores one possible technique to significantly speed up arguments. This technique has some interesting properties but doesn't yet seem especially promising. The biggest difficulty seems to be that it's hard for a realistic reinforcement learner to effectively imitate a human, and thi protocol relies on the imitation being compelling enough to fool another reinforcement learner.