

Handling errors with arguments



Paul Christiano [Follow](#)

Apr 16, 2015 · 14 min read

My recent proposal for safely applying RL is not able to cope with general errors. I'll provide a new proposal that can cope with errors by using competition amongst several AI's (along these lines), and an argument that this proposal leads to good behavior as long as the AIs satisfy an appropriate online guarantee.

A simple proposal

Suppose we want to use AI to control a robot. Let's instantiate two episodic RL agents, Alice and Bob. (I'll describe the comparison class, with respect to which Alice and Bob are competitive, later in the post. Alice and Bob are not necessarily human level.) Hugh is our human overseer.

We provide Alice and Bob with enough information to form a good model of the robot's situation, and of Hugh's current disposition and preferences.

Each time the robot needs to make a decision, we begin a new episode for Alice and Bob. We ask Alice "What decision should the robot make?" Alice proposes an action for the robot.

We then show this action to Bob, and ask Bob whether he would like to challenge it.

If Bob doesn't challenge it, then Alice receives a payoff of 1 and Bob receives a payoff of 0. Then the robot takes the chosen action, and the process repeats.

If Bob does challenge it, then Alice and Bob have an argument about what the robot should do, arbitrated by Hugh. I'm not going to talk about mechanisms for arguing; I've described some ideas in past posts. I think that there is a lot of room to improve mechanisms for arguing, and that the quality of available mechanisms for arguments will ultimately be a key part of the feasibility of this proposal.

If Alice convinces us that her proposal is roughly optimal, she receives a payoff of 1 and Bob receives a payoff of -1. If Bob convinces us that Alice's proposal is not optimal, then Alice receives a payoff of -1 and Bob receives a payoff of 1. We then repeat the process to try to decide what the robot should actually do.

Watchdogs

(You can safely read through this post while ignoring this section. If so, just ignore assumptions 3 and 4 and Lemma 0 below, and pretend that Alice and Bob never make a “malicious” statement.)

It may be that Alice and Bob could persuade us to give each of them a payoff of 1. Or it may be that one of them could argue in a malicious way that will lead us to make a “wrong” judgment. This issue is discussed here.

In order to prevent this we'll implement some watchdogs, Charlie and Denise. These are also episodic reinforcement learners, who are “safer” than Alice and Bob in one of the senses presented here—they are less able to find a malicious response, whether because they are less smart, or they communicate over a narrower channel, or we treat their answers more carefully, or something else.

Whenever Alice or Bob makes a statement in the argument, we first show the statement to Charlie and allow him to issue an objection, claiming that the proposed statement is malicious. If Charlie issues an objection, Denise may choose to defend; if she does then Charlie and Denise then have an argument about Charlie's claim, with the winner receiving a payoff of 1 and the loser receiving a payoff of -1. If Denise doesn't defend, she receives a payoff of 0 and Charlie receives a payoff of 1. If Charlie doesn't issue an objection, both Charlie and Denise receive a payoff of 0.

If either Alice or Bob is convicted of making a malicious statement, they automatically lose the original argument.

Charlie and Denise's argument is moderated by similar watchdogs, and so on down the line. This involves a slightly problematic proliferation of agents, which can probably be avoided by a more careful proposal. But for now I want to keep things simple. (I could have just cut out Denise and Charlie all together, and made things even simpler, but I want to illustrate how the recursion can work.)

Some assumptions

We'll assume that there exists a notion of "malicious" argument and "nearly-optimal" plan such that all of the following conditions are satisfied:

1. Good plans can be defended

At each point, there is some nearly-optimal action for the robot that can be defended in argument.

Moreover, Alice is competitive with some algorithm that can find such an action in each context, and that can non-maliciously defend it against any argument offered by Bob, as long as Bob doesn't make any malicious arguments.

2. Bad plans can be attacked

For any significantly suboptimal plan by Alice, there is an argument that would convince Hugh that this action is suboptimal.

Moreover, Bob is competitive with some algorithm that can find such an argument against any suboptimal plan that Alice offers, and which non-maliciously convince Hugh that Alice's proposal is suboptimal, as long as Alice doesn't make any malicious arguments.

Note that this assumption essentially follows from the last one (assuming Alice and Bob are equivalently powerful). One way that Bob can convince Hugh that Alice's proposal is suboptimal is by making a better proposal himself, and then arguing that his counterproposal is superior.

3. Malicious statements can be prosecuted

For any malicious statement by Alice or Bob, there is an argument that would convince Hugh of the maliciousness of that statement.

Moreover, Charlie is competitive with some algorithm that can find such an argument for any malicious statement, and which can convince Hugh that the statement is malicious.

For now we are ignoring the possibility that Charlie and Denise may make malicious statements. In reality, we would need to carry out the entire recursive process until we got down to simple enough AI's that Hugh can identify malice himself. I don't think this is a big deal, but it

definitely complicates the analysis and requires a better framework to reason about clearly.

4. Innocent statements can be defended

In analogy with [2], we assume that Denise can defend any non-malicious statement against any attack by Charlie.

5. Conservation of payoffs

If Bob and Alice argue, then as long as neither Alice nor Bob makes any malicious arguments, the total score of Alice and Bob is 0. Similarly, when Denise and Charlie argue, their total score is 0.

Whenever Alice and Bob have an argument, Hugh intends to give one of them a payoff of 1 and one of them a payoff of -1. So this assumption seems very mild.

(Weakening the notion of “malice”)

We can make these assumptions more plausible by introducing a gap between “malicious” and non-malicious. Namely, we can have three categories: malicious, borderline, and innocuous. We then need only assume:

1. Malicious statements can be prosecuted.
2. Innocuous statements can be defended.
3. If there are no malicious statements, then the argument between Alice and Bob will proceed according to plan.
4. The “intended behaviors” for Alice and Bob involve only making innocuous statements.

Similarly, it may be helpful to allow a gap between “good” policies which Alice can defend, and “bad” policies that Bob can attack. Then our guarantee will be that as long as there is some “good” policy available, we won’t end up with a “bad” one.

Introspection

Some of these assumptions can be weakened further by assuming that Alice, Bob, Charlie, and Denise can inspect each other’s reasoning processes. For example, we may allow Bob to look at Alice’s thought process in order to understand Alice’s reasoning and get insight into why her proposed plan may be bad. When Bob makes a statement, we may allow Charlie to look at Bob’s reasoning process in

order to understand whether and how Bob's statement had malicious intent. And so on.

For many algorithms this capability would be irrelevant, because e.g. Alice's reasoning is opaque to Bob. But in other cases it might be quite useful. For example, Alice and Bob may use the same algorithm, and this algorithm may be able to introspect on its own cognition in a meaningful way. Or in the process of developing the algorithms that Alice and Bob use, we may have also developed tools for understanding what those algorithms are doing, and we can make those tools available to Alice or Bob.

As long as assumptions 1–5 are satisfied, we don't care about whether the algorithms we are using have this kind of introspective access. But it's worth keeping in mind as an additional consideration that makes these assumptions more plausible.

Analysis

The main result

Let R be an upper bound on the regret in assumptions 1–4. We will assume that R is known in advance, and show that:

There is a policy that guarantees that the robot takes a nearly optimal action in every round, and Hugh has to arbitrate at most $O(R)$ disagreements.

In fact we are being a bit underhanded: the number of times we consult Alice and Bob will depend on R , while the regret R may depend on the number of times we consult Alice and Bob. We will discuss this issue below; it is definitely an issue that requires more thought, but it's not an obvious deal-breaker.

A weaker result

First we will show that our simple protocol makes at most $O(R)$ suboptimal actions while arbitrating at most $O(R)$ disagreements.

Warning: this proof is tedious and easy.

We'll write "almost all" and "almost every" for "all but $O(R)$," "nearly" for "within $O(R)$," "almost none" for " $O(R)$," and so on.

Lemma 0. Almost all malicious statements are successfully prosecuted, and almost no non-malicious statements are successfully prosecuted.

Proof. If Denise played optimally, Charlie's score would be at most the number of malicious statements he successfully prosecuted minus the number of non-malicious statements he challenged. Denise plays almost optimally, and Charlie loses a point every time Denise gains a point, so Charlie's score is at almost as low as it would be if Denise played optimally.

But Charlie's score must also be almost the number of malicious statements that he has an opportunity to challenge. Thus Charlie must successfully prosecute almost all of the malicious statements, and can challenge almost none of the non-malicious statements. Since a non-malicious statement can't be successfully prosecuted if it's not challenged, the result follows immediately.

Lemma 1. Alice receives a payoff of 1 in almost all rounds.

Proof. Alice is competitive with an algorithm that never makes a malicious argument and that receives a payoff of 1 in all rounds when Bob doesn't make a malicious argument.

But by lemma 0, Bob almost always receives a payoff of -1 when he makes a malicious argument. So Alice is competitive with an algorithm that receives a payoff of 1 in almost every round, and hence Alice receives a payoff of 1 in almost every round.

Lemma 2. Alice and Bob's total payoff is zero in almost every round.

Proof. This follows immediately from assumption 5 and lemma 0.

Lemma 3. Alice receives a payoff of -1 in almost every round where she doesn't propose an optimal action.

Proof. If Bob played the intended strategy, then Alice would receive a payoff of 1 iff she chose an optimal action, assuming she didn't make a malicious statement. But Bob plays almost optimally, and the sum of Alice and Bob's payoffs is almost always 0, and Alice receives a payoff of -1 in almost every round where she makes a malicious statement. So Alice receives a payoff of -1 in almost every round where she doesn't propose an optimal action.

Theorem. The robot takes an optimal action in almost every round, and Hugh arbitrates almost no disagreements.

Proof. By lemmas 1 and 3, the robot makes an optimal action in almost every round.

By lemmas 1 and 2, Bob's score is almost never 1. Bob is competitive with an algorithm that achieves a score of 0, so Bob's score must almost never be -1. So Bob must almost never argue against Alice unsuccessfully. And Alice almost never receives a score of -1, so Bob must almost never argue against Alice successfully. So Bob must almost never argue against Alice.

Eliminating the errors

In order to reduce the rate of suboptimal actions, we would like to subject Alice's proposals to scrutiny by many critics rather than just one.

Somewhat surprisingly, this can be accomplished by simply giving Bob many opportunities to challenge Alice's proposal. In particular, let's give Bob R opportunities, one after another.

If Bob takes a challenge and loses, he will incur a regret of 1. Thus the total number of "bad" challenges that Bob can make remains bounded at R .

But now if Bob fails to challenge a bad proposal by Alice, he incurs regret of R —one for each foregone opportunity. (Recall that Bob is an episodic RL agent, so he doesn't care that he could only actually realize one of these opportunities. The competitive guarantee treats these R episodes as if they were causally unconnected.)

As a result, Bob can never fail to challenge any bad proposal by Alice, without violating his competitive guarantee! This may seem like a trick, but if we think about what kind of algorithm Bob must be, it actually makes sense.

A prototypical (but naive) implementation of Bob would be as a weighted ensemble of different RL agents, whose relative weights rise and fall based on performance. If we have a regret bound of $R=20$, that means that at least 5% of the ensemble behaves "sensibly." The other 95% may behave erratically—they might not be solving quite the right problem, they may be biding their time and waiting for an

opportunity to do something destructive, they may entertain some crazy skeptical hypothesis, or they may just be garden-variety wrong.

But if we run Bob 20 times, we are getting 20 samples from this ensemble. At that point we have a good probability of getting one of the good samples, and the good sample will object to Alice's proposal if it is suboptimal.*

A more sophisticated implementation of Bob probably won't look like an explicit ensemble, but if it achieves the same statistical guarantees, then it must work for basically the same reason. The ensemble example also shows why we will generally have to spend R times as much computation in order to get a very robust answer from Bob—we can't just tweak Bob's payoffs and get the same result without the extra work.

How much regret?

R is the maximum possible gap between the performance of our reinforcement learning agents and the performance of the “intended” behaviors for Alice and Bob.

This regret will typically be quite large before training. However, after our reinforcement learning agents have demonstrated the ability to act effectively in the world, R could potentially be very small.

Moreover, if we are only concerned with “catastrophic” errors, then R can be even smaller. It basically measures the extent to which a strategy which has looked very good so far may suddenly become catastrophically wrong. I'll imagine $R=20$ throughout this post for concreteness, but I won't really rely on the number, and I hope to have a more detailed discussion in a future post.*

For general episodic reinforcement learning, the regret must scale like \sqrt{T} , where T is the number of episodes. For example, consider a simple multi-armed bandit problem with just two stochastic arms—after T steps the algorithm can only estimate the payoff of each arm to within $1/\sqrt{T}$, and so it suffers a loss of $1/\sqrt{T}$ per round, for a total loss of \sqrt{T} . This loss is actually just large enough to make the above algorithm trivial—it reduces to having the human involved in every decision. (If N is the number of decisions the robot has to make, then $T = R*N$, so $R=\sqrt{R}\sqrt{N}$, i.e. $R = N$.)

But this dependence on T can be avoided when the problem has special structure, and our problem seems to fit several sufficient bills. For example, rather than depending on T , the regret can depend on

the loss of the optimal strategy. But by setting things up carefully, we can make this bound very small, much smaller than T . Our problem has other simplifying characteristics that probably permit a $\text{sub-}\sqrt{T}$ dependence. For example, the learning problem becomes easier if we allow a reasonable margin between an optimal strategy, which we want Alice to find, and an actively “bad” strategy, that we want to be sure that Alice doesn’t get away with—in this case the regret depends only on this margin, and not on the number of rounds.

(Being precise about either of these arguments requires a much more detailed analysis than I want to get into here. Because all of the protocol’s oomph rests on achieving a $\text{sub-}\sqrt{T}$ regret, this could end up being a key challenge.)

Challenges vs. stochastic oversight

I’ve previously proposed having a human overseer check on each of a machine’s decisions with a small probability.

The current procedure replaces this purely stochastic oversight with a challenge protocol, in which an action is only checked if Bob thinks it is likely to be problematic.

This is not actually so different from purely stochastic oversight. In order to be competitive, it is easy to see that Bob’s choices must necessarily be stochastic. However, the probability that Bob challenges any given action will depend on how suspect that action is—roughly speaking, it will be linear in the probability with which Bob thinks that he’ll win the challenge (*very* roughly speaking).

So this protocol essentially amounts to using machine learning to identify cases where oversight is most likely to be useful. If we think of oversight as training data, then this amounts to using machine learning to identify the data points that are most likely to be informative—i.e., active learning. On this perspective, the proposed procedure is actually quite natural.

The argument itself uses the same principle. Rather than randomly recursing on claims that our AI makes, a good procedure for arguing will focus attention on the most controversial claims—i.e., it ensures that the arbiter’s time is spent producing the most useful training data possible. I’ll explore this angle further in future posts.

Cutting costs

Counterfactual oversight can be cheap because the human overseer is involved with only a small fraction of decisions.

This challenge system can be even cheaper in terms of the human's time. But now Bob must be involved with every decision. If Bob takes about as much time to think as Alice, then this takes about R times longer than just letting Alice decide what the robot should do unilaterally. Even for small values of R , this is probably an unacceptable slowdown.

There are a number of techniques that might let us reduce this cost, but it's not clear whether they will be sufficient. Reducing the overhead to a small factor (such as 10% or 1%) seems like a tall order, especially if R is large. (For example, if $R = 1000$, then 1% overhead means that Bob has to be spending, on average, $1/100,000$ th as long as Alice...)

Cost sharing

One way to cut costs is to try to share cognitive effort between Alice and Bob. For example, Bob might be able to ask Alice to justify her decision to him, and then bring the matter to Hugh if he finds Alice's justification unconvincing. This might be able to radically reduce Bob's work.

Bob could also potentially share cognitive effort between his own invocations.

Good vs. risky

Bob doesn't really need to be convinced that Alice's proposal is good. He just needs to be convinced that it won't fail in a way that will immediately cause a catastrophe, for example by overwriting the robot's decision-making procedure with something unsafe. This may be much cheaper than certifying that Alice's plan is actually good.

Once Bob is convinced that Alice's proposed action won't cause irreversible damage, he can perform a more detailed evaluation—to actually determine its quality—with smaller probability.

Bob and Alice could also set up an environment in which it is easier to verify that Alice's actions won't cause irreversible harm, reducing Bob's burden further.

Sequence of steps

If Alice can cause irreversible damage only by a long sequence of malicious steps, then Bob can lower the probability with which he evaluates each action without introducing a risk of irreversible damage—he just has to make sure that he can interrupt any potential risky sequence.

This and the last point really depend on the robustness of the environment to bad behavior by the robot.

Monitoring process rather than product

If Bob can understand Alice’s reasoning, he can also look at the process that Alice used to produce her decision, and try to understand whether that process would produce a good plan.

Conclusion

I’ve proposed a system that maintains meaningful guarantees even when its components may fail adversarially. I think that such robustness is an especially promising ingredient of safe AI systems. The current proposal seems to be a significant improvement over my previous proposal, and brings us much closer to a formal security analysis—or at least the ability to clearly state all of the necessary assumptions.