Benign model-free RL

Paul Christiano Follow
Mar 19, 2017 · 10 min read

In my last post, I described three research areas in AI control that I see as central: reward learning, robustness, and deliberation.

In this post I argue that these three pieces may be *sufficient* to get a benign and competitive version of model-free reinforcement learning. I think this is an important intermediate goal of solving AI control.

This post doesn't discuss benign model-based RL at all, which I think is another key obstacle for prosaic AI control.

(*This post overlaps extensively with my post on ALBA, but I hope this one will be much clearer. Technically, ALBA is an implementation of the general strategy outlined in this post. I think the general strategy is much more important than that particular implementation.*)

Ingredients

Reward learning and robustness

Given a benign agent H, reward learning allows us to construct a reward function r that can be used to train a weaker benign agent A. If our training process is robust, the resulting agent A will remain benign off of the training distribution (though it may be *incompetent* off of the training distribution).

Schematically, we can think of reward learning + robustness as a widget which takes a slow, benign process H and produces a fast, benign process A:

Reward learning

A's capabilities should be roughly the "intersection" of H's capabilities and our RL algorithms' competence. That is, A should be able to perform a task whenever *both* H can perform that task and our RL algorithms can learn to perform that task.

In these pictures, the vertical axis corresponds intuitively to "capability," with higher agents being more capable. But in reality I'm thinking of the possible capabilities as forming a complete lattice. That is, a generic pair of levels of capabilities is incomparable, with neither strictly dominating the other.

Amplification

If we iteratively apply reward learning and robustness, we will obtain a sequence of weaker and weaker agents. To get anywhere, we need some mechanism that lets us produce a *stronger* agent.

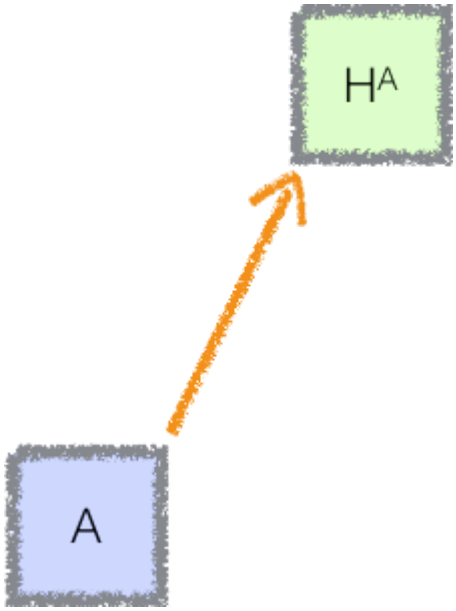
The capability amplification problem is to start with a weak agent A and a human expert H, and to produce a significantly more capable agent H^A. The more capable agent can take a lot longer to think, all we care about is that it *eventually* arrives at better decisions than A. The key challenge is ensuring that H^A remains benign, i.e. that the system doesn't acquire new preferences as it becomes more capable.

An example approach is to provide A as an assistant to H. We can give H an hour to deliberate, and let it consult A thousands of times during that hour. H^A's output is then whatever H outputs at the end

of that process. Because H is consulting A a large number of times, we can hope that the resulting system will be much smarter than A. Of course, the resulting system will be thousands of times more computationally expensive than A, but that's fine.

In general, meta-execution is my current preferred approach to capability amplification.

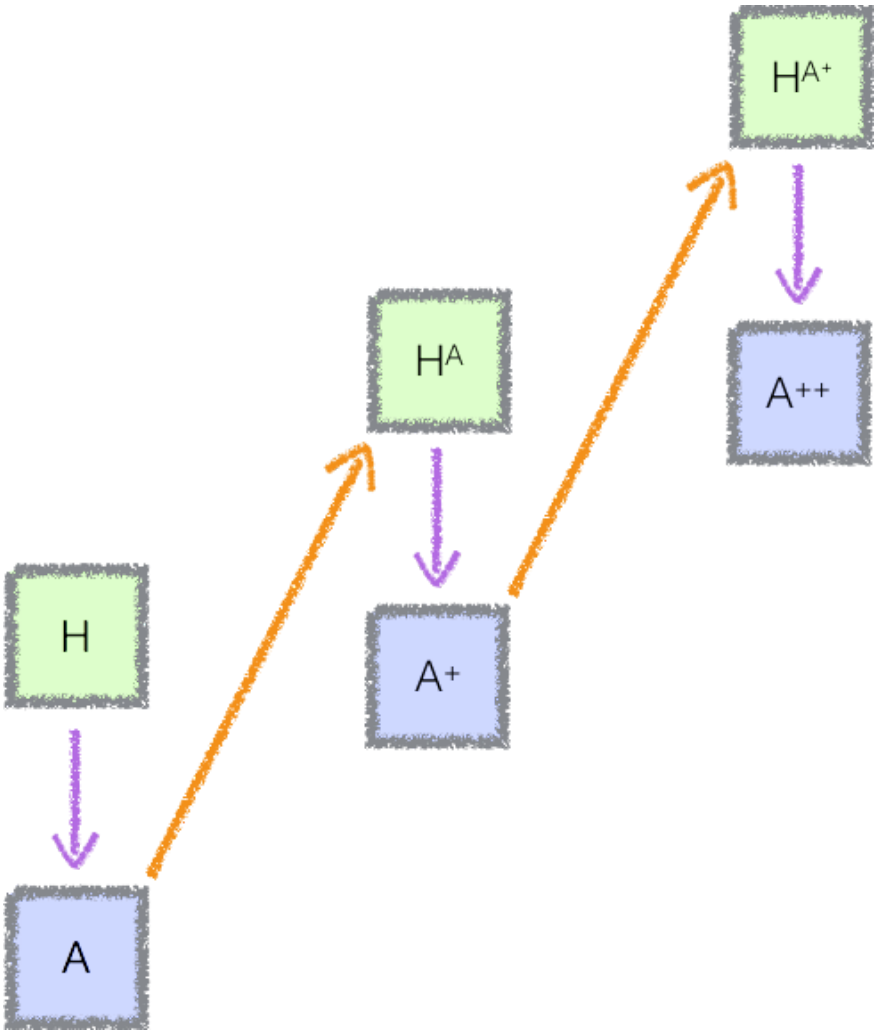
Schematically, we can think of amplification as a widget which takes a fast, benign process A and produces a slow, benign process H^A:



Amplification

Putting it together

With these two widgets in hand, we can iteratively produce a sequence of increasingly competent agents:



Iterated amplification + reward learning

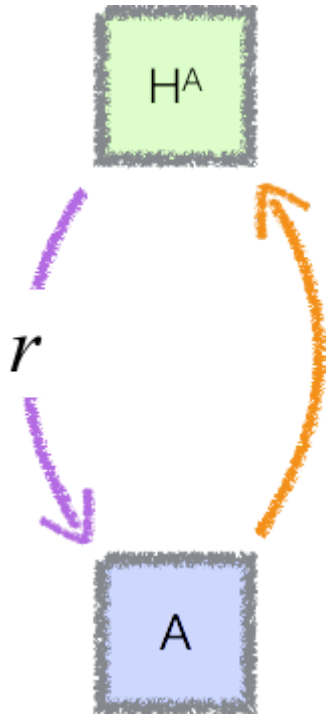
That is, we start with our benign expert H. We then learn a reward function and train an agent A, which is less capable than H but can run much faster. By running many instances of A, we obtain a more powerful agent H^A, which is approximately as expensive as H.

We can then repeat the process, using H^A to train an agent A^+ which runs as fast as A but is more capable. By running A^+ for a long time we obtain a still more capable agent H^{A^+}, and the cycle repeats.

Collapsing the recursion

I've described an explicit sequence of increasingly capable agents. This is the most convenient framework for analysis, but actually implementing a sequence of distinct agents might introduce significant overhead. It also feels at odds with current practice, such that I would be intuitively surprised to actually see it work out.

Instead, we can collapse the entire sequence to a single agent:



An agent defines its own reward function

In this version there is a single agent A which is simultaneously being trained and being used to define a reward function.

Alternatively, we can view this as a sequential scheme with a strong initialization: there is a separate agent at each time t, who oversees the agent at time t+1, but each agent is initialized using the previous one's state.

This version of the scheme is more likely to be efficient, and it feels much closer to a practical framework for RL. (I originally suggested a similar scheme here.)

However, in addition to complicating the analysis, it also introduces additional challenges and risks. For example, if H^A actually consults A, then there are unattractive equilibria in which A manipulates the reward function, and the manipulated reward function rewards manipulation. Averting this problem either requires H to sometimes avoid depending on A, or else requires us to sometimes run against an old version of A (a trick sometimes used to stabilize self-play). Both of these techniques implicitly reintroduce the iterative structure of the original scheme, though they may do so with lower computational overhead.

We will have an even more serious problem if our approach to reward learning relied on throttling the learning algorithm. When we work with an explicit sequence of agents, we can ensure that their

capabilities improve gradually. It's not straightforward to do something analogous in the single agent case.

Overall I think this version of the scheme is more likely to be practical. But it introduces several additional complications, and I think it's reasonable to start by considering the explicit sequential form until we have a solid grasp of it.

Analysis

I'll make two critical claims about this construction. Neither claim has yet been formalized, and it's not clear whether it will be possible to formalize them completely.

Claim #1: All of these agents are benign.

This is plausible by induction:

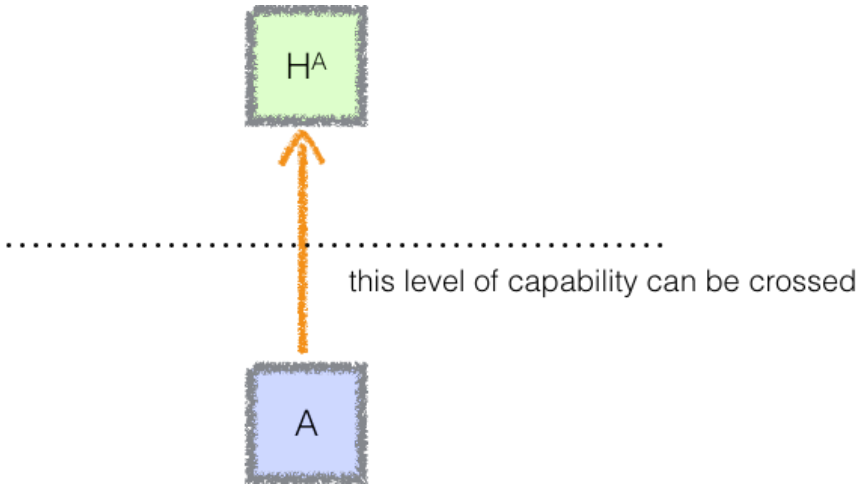
- The original expert H is benign by definition.
- If we start with a benign overseer H, and have working solutions to reward learning + robustness, then the trained agent A is benign.
- If we start with a benign agent A, and have a woking solution to capability amplification, then the amplified agent H^A will be benign.

There are important subtleties in this argument; for example, an agent may be benign with high probability, and the error probability may increase exponentially as we proceed through the induction. Dealing with these subtleties will require careful definitions, and in some cases adjustments to the algorithm. For example, in the case of increasing failure probabilities, we need to strengthen the statement of amplification to avoid the problem.

Claim #2: The final agent has state-of-the-art performance.

This is plausible if our building blocks satisfy several desirable properties.

First, capability amplification should be able to cross every level non-maximal level of capability. That is, for every level of capability, it is possible to start with an agent A who is below that level, and end up with an agent H^A which is above that level:



For every possible place we could put the dotted line — every possible capability level — there must be some agent A for whom the orange arrow crosses that dotted line. Otherwise we would never be able to get to the other side of that dotted line, i.e. we would never be able to surpass that level of capability.

Second, capability amplification should be monotonic (if A is at least as capable as B then H^A should be at least as capable as H^B).

Third, reward learning should yield an agent whose capabilities are at least the infimum of our RL algorithm's capabilities and the overseer's capabilities, even if we train robustly.

Now given a sequence of increasingly powerful fast agents we can take the supremum of their capabilities. Those agents will all be weaker than our RL algorithms and so the supremum is not the maximal capability, so we can consider a starting point from which capability amplification would cross that supremum. By hypothesis the sequence must eventually cross this starting point, and at that point amplification will push it above the supremum (and reward learning will keep it above the supremum). Making this argument carefully shows that the supremum is the state of the art for RL algorithms and that we attain the supremum after some finite number of steps. (Though all of this is based on a leaky abstraction of "capabilities.")

Cost

I think this proposal will be most helpful if it imposes minimal additional overhead. My main goal is to develop algorithms with sublinear overhead, i.e. for which the fraction of overhead converges to 0 as the underlying algorithms become stronger.

The cost of this scheme depends on the quantitative properties of our basic building blocks:

Factor #1: How much do reward learning and robustness slow down training?

During RL, we need to evaluate the agent A many times. If we want to use a learned reward function we may need to evaluate A more times. And if we want to train a policy which remains benign off of the training distribution, we may need to evaluate A more times (e.g. since we may need to do adversarial training). Ideally that overhead will shrink as our algorithms become more powerful.

I think this is plausible but far from certain (for now it is uncertain whether reward learning and robustness are even plausible). Some reassuring factors:

- Reward learning / adversarial training can actually improve the performance of our system—the computational time spent on them might actually be well-spent even from a capabilities perspective
- The difficulty of the "additional learning problem" we are trying to solve in each case (e.g. the concept of "defer to human control") may not scale up linearly with the complexity of the underlying domain.

Factor #2: how many times do we have to invoke the overseer during training?

In addition to calling the agent A, we will need to call the overseer H in order to get information about the reward function. Because the overseer is much more expensive than the agent, we would like to minimize the number of times we call the overseer. This can be quantified by the ratio between the number of calls to H and the number of calls to A. For example, we may need to call H once for every hundred calls to A.

Factor #3: how expensive is capability amplification?

Capability amplification is possible only because we allow the agent H^A to think for much longer than A. But "much longer" could represent a range of values: is H^A a hundred times more expensive to evaluate than A? A thousand? A million?

Roughly speaking, factors #2 and #3 should be multiplied together to get the overhead from reward learning: factor #2 tells us how many

times we have to call the overseer, while factor #3 tells us how expensive the overseer is.

The total overhead is thus (Factor #1) + (Factor #2) * (Factor #3). As an example, I'd be happy with values like 10% + 0.01% × 1000 = 20%.

Factor #4: do we need to train many separate agents?

If we need to use a sequence of N increasingly capable agents, then we would naively increase our training time by a factor of N. Naively, this would dominate the overhead, and in order for the scheme to be workable I think we would need to avoid it. I see a few plausible approaches:

- We could use the collapsed version with a single agent.
- We could use some other initialization or parameter-sharing scheme to effectively reuse the computational work done in training earlier agents.
- The earlier agents could require significantly less training time than the final agent, e.g. because they are less capable. For example, if each agent takes only 20% as long to train as the following one, then the total overhead is only 25%.

These mechanisms can work together; for example, each agent may require some amount of non-reusable computation, but that amount may be reduced by a clever initialization scheme.

Conclusion

I've outlined an approach to AI control for model-free RL. I think there is a very good chance, perhaps as high as 50%, that this basic strategy can eventually be used to train benign state-of-the-art model-free RL agents. Note that this strategy also applies to techniques like evolution that have historically been considered really bad news for control.

That said, the scheme in this post is still extremely incomplete. I have recently prioritized building a practical implementation of these ideas, rather than continuing to work out conceptual issues. That does not mean that I think the conceptual issues are worked out conclusively, but it does mean that I think we're at the point where

we'd benefit from empirical information about what works in practice (which is a long way from how I felt about AI control 3 years ago!)

I think the largest technical uncertainty with this scheme is whether we can achieve enough robustness to avoid malign behavior in general.

This scheme does not apply to any components of our system which aren't learned end-to-end. The idea is to use this training strategy for any internal components of our system which use model-free RL. In parallel, we need to develop aligned variants of each other algorithmic technique that plays a role in our AI systems. In particular, I think that model-based RL with extensive planning is a likely sticking point for this program, and so is a natural topic for further conceptual research.