The reward engineering problem

Paul Christiano Follow
May 30, 2016 · 9 min read

Today we usually train reinforcement learning agents to perform narrow tasks with simple goals. We may eventually want to train RL agents to behave "well" in open-ended environments where there is no simple goal.

Suppose that we are trying to train an RL agent **A**. In each episode, **A** interacts with an environment, producing a transcript τ. We then *evaluate* that transcript, producing a reward $r \in [0, 1]$. **A** is trained is to maximize its reward.

We would like to set up the rewards so that **A** will learn to behave well—that is, such that *if* **A** learns to receive a high reward, *then* we will be happy with **A**'s behavior.

To make the problem feasible, we assume that we have access to another agent **H** which

1. is "smarter" than **A**, and
2. makes "good" decisions.

In order to evaluate transcript τ, we allow ourselves to make any number of calls to **H**, and to use any other tools that are available. The question is: how do we carry out the evaluation, so that the optimal strategy for **A** is to also make "good" decisions?

Following Daniel Dewey, I'll call this the *reward engineering problem*.

Note that our evaluation process may be quite expensive, and actually implementing it may be infeasible. To build a working system, we would need to combine this evaluation with semi-supervised RL and learning with catastrophes.

Possible approaches and remaining problems

I know of 3 basic approaches to reward engineering:

1. **Direct supervision**. Use **H** to evaluate **A**’s behavior, and train **A** to maximize **H**’s evaluations. In some contexts we could compare two behaviors instead of evaluating one in isolation.
2. **Imitation learning**. Use **H** to generate a bunch of transcripts, and train **A** to produce similar-looking transcripts. For example, we could train a model to distinguish **A**’s behavior from **H**’s behavior, and reward **A** when it fools the distinguisher.
3. **Inverse reinforcement learning**. Use **H** to generate a bunch of transcripts, and then infer a reward function which is being approximately optimized by **H**. Use this reward function to evaluate **A**’s behavior.

All of these approaches are promising but face significant challenges. I’ll describe some of these problems in the next 3 sections.

1. Direct supervision

In direct supervision, **H** looks at a transcript of **A**’s behavior, and estimates how good that transcript is.

To see the problem with this scheme, suppose that **A** has been asked to draw a picture, and **A** does it by copying an existing picture with some modifications. If originality is especially important, then this may be a very “bad” policy. But even if **H** is much smarter than **A**, it may be hard to tell that the picture is not original—creating a derivative work only requires looking at a single existing picture, while checking if a work is derivative requires considering *every* picture.

More formally: in order for direct supervision to be effective, **H** needs to be better-informed than **A** about what is “good.” If this condition is satisfied, then from **A**’s perspective, estimating **H**’s estimate of goodness is equivalent to estimating actual goodness. This condition is superficially plausible—after all, we did assume that **H** is smarter than **A**.

The problem is that when **A** picks an action, **A** is especially well-informed about that action—the computation which produced the action provides evidence about it, and **H** may not have access to that evidence.

Transparency

One response is to let **H** see *how* **A** computed its action. If **H** can understand that process, then **H** may be able to effectively evaluate the action.

Sometimes this is straightforward: for example, if **A** uses an attention mechanism to look at a particular painting and copy it, we can simply tell **H** what **A** looked at.

In other contexts it might be much more challenging. For example, **A** may copy an image by "memorizing" the image in its weights, rather than by "looking at" the image at test time.

One approach to these challenging cases is to train **A** to produce actions *and* to produce explanations that are maximally helpful for evaluating those actions. For example, we may train **A** to produce images *and* to point out similarities between its output and training images.

This is a challenging problem for several reasons. One issue is that producing these explanations involves a huge "action space" and a very complex objective. A more subtle problem is that there are two conflicting objectives: **A** wants to produce actions that **H** evaluates as "good," but providing useful information will sometimes lead **H** to produce a lower evaluation. Training **A** to do both tasks requires a new approach.

Other problems

We can imagine other failure modes of direct supervision. For example, **A** may find an action that exploits one of **H**'s biases or blind spots in order to receive a high rating.

We hope that these "attacks" can only succeed if **H** is ignorant about the process that produced a given action, and so can be resolved by whatever form of transparency allows **H** to accurately evaluate **A**'s actions in general.

That is, if **A** carefully explains to **H** how an action was chosen to exploit **H**'s biases, then **H** can hopefully avoid being exploited. This seems especially plausible given that **H** is smarter than **A**.

2. Imitation learning

Imitation learning has two conceptual problems:

- If **H** is more competent than **A**, then **A** will generally be unable to imitate **H**'s behavior.
- We don't have a totally satisfactory framework for reducing imitation learning to an optimization problem.

What if A can't imitate H?

Suppose that **A** has been asked to build a block tower. **H** can quickly stack the blocks, and 99% of the time the tower stays standing; 1% of the time **H** messes up and the tower falls down. **A** is not as capable as **H**, and so if it tries to stack the blocks quickly the tower falls down 100% of the time.

The "best" behavior for **A** may be to stack the blocks more slowly, so that the tower can stay standing. But this behavior is hard to induce with imitation learning, because **H** *never* stacks the blocks slowly. Instead, an imitation learner is more likely to try to stack the blocks quickly and fail (since at least **H** does this 1% of the time).

One response to this problem is to have **H** "dumb down" its behavior so that it can be copied by **A**.

However, this process may be challenging for **H**. Finding a way to do a task which is within **A**'s abilities may be much harder than simply doing the task—for example, it may require a deep understanding of **A**'s limitations and capabilities.

I've proposed a procedure, "meeting halfway," for addressing this problem. The idea is that we train a discriminator to distinguish **H**'s behavior from **A**'s behavior, and use the discriminator's output to help **H** behave in an "**A**-like" way. This proposal faces many challenges, and it's not at all clear if it can work.

How do you train an imitator?

The plagiarism example from the last section is also a challenge for imitation learning. Suppose that **A** has been asked to draw a picture. **H** would draw a completely original picture. How can we train **A** to draw an original picture?

The most plausible existing approach is probably generative adversarial networks. In this approach, a discriminator is trained to distinguish **A**'s behavior from **H**'s behavior, and **A** is trained to fool the discriminator.

But suppose that **A** draws a picture by copying an existing image. It may be hard for the discriminator to learn to distinguish "original image" from "derivative of existing image," for exactly the same reasons discussed before. And so **A** may receive just as high a reward by copying an existing image as by drawing a novel picture.

Unfortunately, solving this problem seems even more difficult for reinforcement learning. We can't give the discriminator any access to **A**'s internal state, since the discriminator isn't supposed to know whether it is looking at data that came from **A** or from **H**.

Instead, it might be easier to use an alternative to the generative adversarial networks framework. There are some plausible contenders, but nothing is currently known that could plausibly scale to general behavior in complex environments. (Though the obstacles are not always obvious.)

3. Inverse reinforcement learning

In IRL, we try to infer a reward function that **H** is approximately maximizing. We can then use that reward function to train **A**.

This approach is closely connected to imitation learning, and faces exactly analogous difficulties:

- **H**'s behavior does not give much information about the reward function in regions far from **H**'s trajectory.
- If we first learn a reward function and then use it to train **A,** then the reward function is essentially a direct supervisor and faces exactly the same difficulties.

The second problem seems to be the most serious: unless we find a resolution to that problem, then direct supervision seems more promising than IRL. (Though IRL may still be a useful technique for the resulting RL problem—understanding the supervisor's values is a critical subtask of solving an RL problem defined by direct supervision.)

H's behavior is not sufficiently informative

Consider the block tower example from the last section. If **H** always quickly builds a perfect block tower, then **H**'s behavior does not give any evidence about tradeoffs between different imperfections: how much should **A** be willing to compromise on quality to get the job done faster? If the tower can only be tall *or* stable, which is preferred?

To get around this difficulty, we would like to elicit information from **H** other than trajectories. For example, we might ask **H** questions, and use those questions as evidence about **H**'s reward function.

Incorporating this information is much less straightforward than incorporating information from **H**'s behavior. For example, updating on **H**'s statements require an explicit model of how **H** believes its statements relate to its goals, even though we can't directly observe that relationship. This is much more complex than existing approaches like MaxEnt IRL, which fit a simple model directly to **H**'s behavior.

These issues are central in "cooperative IRL." For now there are many open problems.

The major difficulties of direct supervision still apply

The bigger problem for IRL is how to represent the reward function:

- If the reward function is represented by a concrete, learned function from trajectories to rewards, then we are back in the situation of direct supervision.
- Instead the reward function may act on an abstract space of "possible worlds." This approach potentially avoids the difficulties of direct supervision, but it seems to require a particular form of model-based RL. It's not clear if this constraint will be compatible with the most effective approaches to reinforcement learning.

Ideally we would find a better representation that incorporates the best of both worlds—avoiding the difficulties of direct supervision, without seriously restricting the form of **A**.

Alternatively, we could hope that powerful RL agents have an appropriate model-based architecture. Or we could do research on appropriate forms of model-based RL to increase the probability that they are competitive.

Research directions

Each of the problems discussed in this post is a possible direction for research. I think that three problems are especially promising:

- Training ML systems to produce the kind of auxiliary information that could make direct supervision reliable. There are open theoretical questions about how this training should be done—and huge practical obstacles to actually making it work.
- Developing alternative objectives for imitation learning or generative modeling. There has been a lot of recent progress in this area, and it is probably worth doing more conceptual work to see if we can find new frameworks.
- Experimenting with “meeting halfway,” or with other practical approaches for producing imitable demonstrations.

Conclusion

If we cannot solve the reward engineering problem in practice, it seems unlikely that we will be able to train robustly beneficial RL agents.

Conversely, if we can solve the reward engineering problem, then I believe that solution could be leveraged into an attack on the whole value alignment problem (along these lines—I will discuss this in more detail over my next few posts).

Reward engineering is not only an important question for AI control, but also appears to be tractable today; there are both theoretical and experimental lines of attack. I’m optimistic that we will understand this problem much better over the coming years, and I think that will be very good news for AI control.

(This research was supported as part of the Future of Life Institute FLI-RFP-AI1 program, grant #2015–143898.)