AI control and search/planning

Paul Christiano Follow
Oct 21, 2016 · 5 min read

(*Status: sketchy. Follow-up to: not just learning.*)

Consider the following simple algorithm:

- Train a predicted-reward function R(*o*, *a*) and an action-proposer A(*o*).
- Sample a large number of candidate actions from A(*o*), and then output the action *a* for which R(*o*, *a*) is maximal.

This kind of search poses some challenges for AI control:

1. If R predicts something doesn't quite capture our values, like "will the reward button get pressed," then we will get a reward-button-maximizing AI. This raises a familiar set of concerns.
2. Even if R reflects our values most of the time, the search may find the rare points on which R is badly wrong.

This kind of search can also be generalized in many directions; for example, we can use MCTS to search over sequences of actions rather than individual actions (as in AlphaGo). Many of these variants seem to pose a similar set of problems for AI control.

Problem #1

I think that problem #1 should be regarded as a bug in the reward function; if we can solve AI control for the techniques that produced R, then we should be able to address this problem.

For example, if we train R using supervised learning from actual button presses then we will have trouble. But if we have solved AI control for supervised learning, then we could use that scheme to learn an equally-sophisticated reward function which captured what we care about "to the best of its abilities" (the aligned learner could treat data about button presses as a useful signal to help predict goodness). For example, if R is smart enough to predict that action *a* will lead to the user's death, then it's reasonable to expect R to be smart enough to know that action *a* is bad.

If we can't solve AI control for supervised learning then we are in trouble whether or not we perform the search. Either way, no need to blame the search.

Problem #2

I think that problem #2 is much more serious.

In the best case, we can think of the reward function R as itself being an aligned AI, which looks at an action and determines how good it is.

The whole point of doing the search is to turn up actions that are more sophisticated than anything that R could generate itself. If we are running a *really big* search, then we are trying to find actions that are more sophisticated than anything that R has ever seen before.

This is a recipe for R to make a terrible mistake. Maybe you'll just get something like an adversarial example which isn't a meaningful action. Or maybe you will get something more sinister, which does damage when R attempts to evaluate it (e.g. the search could turn up security vulnerabilities). In any case, there isn't much reason to think that you will get an outcome which is actually *good*, while there is reason to think that you might get an outcome which is very *optimized*. I think that should be cause for concern.

Current state of affairs

I think that this is a pretty hard problem, and that most proposed responses are not satisfactory:

- Any response along the lines "don't run a giant brute force search" should explain how to be algorithmically competitive with an alternative that *does* run a giant brute force search (or else accept that aligned AI systems may be at a large competitive disadvantage).
- Any solution that depends on having a very expensive function R should explain how that isn't going to become the bottleneck for the search and drag down performance.
- Any solution that involves crossing our fingers and hoping for the best should probably say something about why we expect R to behave *at all sensibly* on an input that is totally unlike anything it has seen before, and was selected for being a point where R was likely to make an error.

So far this hasn't really been an issue in practical systems, perhaps because extensive search has mostly been effective in domains like games or constraint satisfaction where the reward function is taken as given. But when we eventually design algorithms that effectively combine learned models with extensive search, I think this may become a problem. Optimistically it will be a problem that needs to be solved before giant searches work at all; pessimistically, we may find ad hoc solutions that cause our systems to be more brittle and fail in more spectacular ways.

I have a rough angle of attack in mind, which I hope to flesh out soon. The idea is that *most* proposed actions will be typical enough that R can handle them directly. So if we can identify the extreme actions as being extreme, we can afford to spend significantly more time processing them. If we can get abstraction+adversarial training working well enough, then we could present abstract versions of these inputs to R, which could then take the time to evaluate them slowly—with no individual step of the abstract evaluation being too far from R's training distribution.

Optimism about Bayesianism

Many people (including me) have the intuition that a Bayesian reasoner may be able to implement a more robust/safe planning process. I think that this intuition has merit, but ultimately I think that there is still a lot of work to be done if it is going to be turned into a solution:

- If we apply the kind of search described in this post with a Bayesian reasoner, then we get the a that optimizes our estimate of $\mathbb{E}[U|o, \text{do}(a)]$. We might as well call that estimate $R(o, a)$. Being Bayesian doesn't seem to change the basic problem: R needs to be calculated and trained efficiently, and so it is difficult to ensure that it behaves sensibly on actions a selected from an extremely extensive search.

- So I think that optimism about Bayesian is based on the hope that Bayesian methods will learn very robust models that can reliably generalize outside of the training distribution. I think that this is a plausible hope and a promising direction to explore, but for now I would definitely not want to hang my hat on it—I don't have much optimism that *in general* we will be able to learn very-robust models as efficiently as non-robust models. I think that even if research on robustness succeeds spectacularly

it probably won't be able to establish a sufficiently general claim to let us rest easily.

- A Bayesian approach may allow for a tighter coupling between the representation of the goals and the actual mechanics of the search. For example, rather than considering a bunch of actions a we may reason backwards from a goal; this may pose a much different (and potentially easier) problem for AI control. But I think we need to understand whatever techniques are most useful; that might mean backwards-chaining, but it could also mean the kind of more straightforward approach applied here. So I don't think it's yet reasonable to restrict attention to search strategies that have a particular kind of internal structure.

Conclusion

I think that powerful AI systems will probably make use of large-scale search, and we will probably need new techniques in order to perform them safely. I expect this to be a challenging problem, which may be quite different from the kind of difficulties addressed by a scheme like ALBA.