Online guarantees and AI control

Paul Christiano Follow
Apr 14, 2015 · 8 min read

I'm interested in claims of the form: "If we had an AI that could do X well, then we could build an AI which could do Y well."

X may be "prediction," "reinforcement learning," "question-answering," or so on. For concreteness, in this post I'll assume X is prediction.

If we had an AI that could predict well, then we could attack this problem empirically: we could try to build things using our predictor as a building block, and see how far we get.

But if we want to think about the problem in advance, we need some model of what "predict well" means. Normally when people think about AI safety they have an informal model in mind, but I think there are some advantages to precision.

In this post I'll describe a modeling approach based online guarantees. This is the best approach that I'm familiar with, and I think that it's good enough that it's better than a less precise model (or than not doing any theoretical work in advance). But I'm always interested in more suggestions, and I'm not too attached to this approach. (I'm also aware that I may be biased by my own intellectual background.)

Precise thinking about the guarantees of AI is relevant now in order to start doing theoretical work before such AI is available. But this thinking may also be useful in its own right, to reason about the behavior of the systems that we actually build.

In this post I'll focus on stating assumptions about what an AI can do; I won't touch on assumptions about what an AI *can't* do, though those may also be useful.

Comparisons

Suppose that we have a sequence predictor Predict, and we want to articulate the assumption "Predict is a good predictor."

What do we mean by "good"? Normally we'll fix some comparison class C of predictors (which could consist of a single predictor), and say "Predict predicts as well as the best predictor in C."

For example, we might say that Predict predicts as well as the best strategy that any living human could implement, i.e. that Predict is a superhuman predictor. Or we might say that say that Predict predicts as well as the best linear classifier. Or so on.

To make the comparison, we can take our pick from a range of modeling approaches. Some of these assumptions make distributional assumptions on the data—they only apply if the prediction problem is drawn from some particular distribution. These assumptions seem very difficult to work with.

There are two guarantees I am particularly familiar with, that make unusually weak assumptions on the data:

- **Batch.** If you give Predict enough training sequences, and then you give it a new sequence sampled from the same distribution, then in expectation Predict's predictions will be almost as good as the predictions of the best fixed predictor from C. This assumption more or less never applies in a realistic scenario. For example, most distributions aren't perfectly static in time, so if we train our system on May 4, it can't technically be applied to data from May 5.
- **Online**. For any sequence, the total quality of Predict's predictions is close to the total quality of the best fixed predictor from C. I think that this is a good compromise between "attainability" (it is almost as easy as the batch guarantee) and "usefulness" (you can often get actual bounds on realistic systems using this assumption).

If you can't tell, I prefer to use online assumptions. They are the easiest way to get traction on a system's performance without making distributional assumptions on the data. I think this isn't just a pedantic point—these distributional assumptions are actually really important to discussions about AI safety.

Both of these models are pretty crude; even for algorithms designed to have an online guarantee, the online guarantee doesn't very well characterize the algorithm's performance. That said, this is a general problem with precise characterizations. Working with online assumptions can let us have a precise discussion, and algorithms that work with online assumptions are at least promising candidates to think about more deeply or explore empirically.

Regret

Both of these assumptions are parametrized by the “regret,” the gap between the performance of Predict and the performance of the best predictor in C.

The regret may be “small” or “large”, depending on the nature of the task:

- **Small:** Logarithmic in the size of the class C. This normally happens when it is possible to do “off-policy” learning, evaluating the quality of an algorithm even when you aren’t using it.
- **Large**: If you can’t do off-policy learning, you might not be able to notice a “good” algorithm until you try it out. In this case, the regret—the gap between the payoff of our algorithm and the payoff of the best algorithm in C—will typically be linear in the size of C, since we have to try each option.

For prediction, it is possible to do off-policy learning—once you see the next data point, you can tell how well every predictor did—and so the regret is logarithmic. (Though of course efficient algorithms don’t actually consider all of the predictors.) For reinforcement learning, off-policy learning isn’t always possible, and so in general the regret must be linear.

When it is possible to do off-policy learning to a limited extent, you can interpolate between these extremes.

Elaborations

Semi-supervised learning

In semi-supervised learning, the learner is only given feedback on a small amount of data, but is given a larger volume of data in order to help it learn. For example, imagine a classifier which can see billions of images, but only learns the correct classification for a few thousand thousand of them.

This is modeled automatically by online guarantees, if the comparison class C includes algorithms that learn from data. For example, if we compare to the class of humans, then we are implicitly making a guarantee about the system's ability to do semi-supervised learning—since it is competing with humans, who will certainly use all of the data.

Semi-supervised learning still requires some feedback—you can't start labeling data correctly until you've seen some labels. This is an unavoidable limitation of the online learning framework. Of course, sometimes (e.g. for prediction) the "feedback" is inherent in the data itself.

Modifications

A traditional online guarantee is probably not enough for most AI control applications. Instead, it seems more useful to compare an algorithm Predict to a comparison class that is defined in terms of Predict, which contains every "simple modification" of Predict.

To illustrate the problem, assume that Predict is a human-level predictor, i.e. that it is competitive with any strategy that can be implemented by a human. But we make no further assumptions, and so subject to this condition Predict may behave adversarially (for all that we know).

Suppose further that in at least one respect Predict is very superhuman. When asked to make predictions, there is some trivial detail that seems random to a human but which Predict can predict well.

Then Predict can foil most AI control schemes as follows. Every time it predicts the trivial detail well, it can give itself a "point." Whenever we try to use Predict to make an important decision, it can spend a point to make a completely wrong prediction.

If the number of predictions that we care about is smaller than the amount of extraneous detail that Predict can superhumanly predict, then Predict can flub every single important prediction!

It may be possible to cope with this problem, but it seems like we shouldn't have to—this is a very unnatural behavior for Predict.

One way to express this intuition is that while Predict is a human-level predictor, it's a much worse predictor than it could be. That is, a team consisting of Predict and a human predictor can easily obtain a much larger payoff than Predict, by simply defaulting to normal human judgment on the important decisions.

We can formalize this by defining a class C[Predict], as a class of elaborations on Predict. For example, we can consider the class of all strategies that humans could implement if they were able to consult with Predict, or the class of all linear predictors that use Predict's outputs as an additional feature. By saying that Predict is a better predictor than anything from this class, we can express the idea that it's not doing anything really dumb.

Counterfactuals

Consider an agent that plays the following game: in each round it is given an input $x \in X$, then provides an output $y \in Y$, then is told a function $f: X \times Y \rightarrow [0,1]$ and receives a payoff $f(x, y)$.

When we actually apply such a system, the payoff f may be causally related to the agent's choice y. For example, a human user may provide a map f that scores each possible action by the agent. In the meantime the agent's action y might have affected the world, and perhaps also affected the user's judgments.

The agent might play this game in two ways:

1. It predicts what the function f will be, and chooses the action y maximizing $E[f(x, y)]$.
2. For each action y, it predicts what the function f will be if it takes action y. The agent chooses the action maximizing $E[f(x, y) | \text{the agent did } y]$. If the agent has a causal model of the world, it could also use a causal intervention $E[f(x, y) | \text{do}(y)]$.

Doing well according to [1] and doing well according to [2] are different guarantees. We can express each of them as an online guarantee:

1. For any fixed sequence of inputs x and functions f, the agent performs almost as well as the best algorithm in the comparison class.
2. For any fixed sequence of points x and mappings from y to f, the agent performs almost as well as the best algorithm in the comparison class.

These guarantees are achieved by different algorithms, have different difficulties (typically, guarantees like #1 are easier to obtain), and may be useful in different applications (guarantees like #1 seem more useful for AI control). We just need to be clear about which one we are assuming.

In more complex situations there may be more possibilities. A precise statement of the online guarantee will necessarily be completely unambiguous, however.

A note on transfer learning

Many approaches to AI control rest on a form of transfer learning.

For example, suppose that I have a proposal for AI control that requires a powerful question-answering system, e.g. this one. This system requires the question-answerer to answer questions to which we don't know the answer.

I could try to create a question-answering system using a predictor, by giving the predictor a bunch of pairs of the form (Question, Answer).

But if we do this, we are training the predictor only with questions to which we know the answer, and then expecting it to perform well on questions where we don't know the answer.

It seem plausible that this would work. But I don't think you'll be able to state any *sufficient* property of the predictor, without making some unappealing assumptions.

The basic problem is that there are many models that the predictor could learn in order to predict your (Question, Answer) pairs. And you need to know that the predictor is going to choose the "right" one —without being able to really say what the right one is. For example, Solomonoff induction may well learn *your* answer, rather than the "correct" answer (and there many other "failure" modes).

Overall, I am somewhat skeptical about proposals that rest on this kind of transfer learning, because they often make implicit and assumptions about what kind of generalization is the "right" one. I don't think that the kinds of AI we actually build respects these assumptions very reliably, or at all.

Moreover, if you can't ever generate the training data that would distinguish the "right" answer from your answer, then by the same token you can't test whether the learner is doing the right thing—if you can test something, you can train it (at least a bit).

Note that semi-supervised learning is also very similar to transfer learning—the data from one task can be used to help learn how to solve another task, reducing the amount of feedback needed to find a good approach to the new task. But "reducing the amount of feedback needed" is quite different from "eliminating the need for feedback altogether."