Active learning for opaque, powerful predictors

Paul Christiano Follow
Jan 3, 2016 · 6 min read

(An open theoretical question relevant to AI control. **Note**: the problem as stated here is known to be impossible; see the response for some additional assumptions that might make the problem possible.)

Suppose that I have a very powerful prediction algorithm Predict, and I'd like to train it to imitate human behavior. Can I leverage the prediction algorithm itself to minimize the amount of data required?

More precisely, I'm interested in training the predictor to map a question to a sample from the distribution over answers a human would provide. Each day I will use my system to answer 100,000 (=N) questions. I have time to actually elicit a human's response to exactly one of those questions. On each day, if all 100,000 predictions are "acceptably good," then the day goes well. If not, the day goes badly. I would like to minimize the number of bad days.

I know that with a reasonable amount of appropriately-chosen training data, my predictor will converge to making acceptably good predictions of human behavior. More precisely:

- I know there exists a particular prediction algorithm GoodEnough that makes acceptably good predictions. Moreover, if a predictor's expected log loss in a given day is within <1 bit of GoodEnough, then it must also be making acceptably good predictions.
- I know that over any sequence of data points, Predict will make predictions almost as good GoodEnough. Say, the total log loss of Predict will be at most 10 (=R) bits higher than GoodEnough.

For simplicity, we'll assume that the set of 100,000 questions is iid on each day—the distribution over tasks is perfectly static.

The question is: how can I use Predict to choose which data points to get human feedback on, so as to minimize the number of bad days?

As a function of R and N, how few bad days can we guarantee?

The naive strategy of picking a random question each day leads to 1,000,000 (=N·R) bad days. I don't think there is any strategy with less than 1000 (=min{2^R, N}) bad days in the worst case, when we have a bunch of equiprobable models each of which makes exactly one idiosyncratic prediction. I would be very happy with the 2^R bound, or indeed any bound that is independent of N.

Clarifications

I'm happy to make relatively strong assumptions on GoodEnough (and hence Predict), short of assuming that it can predict the human perfectly. For example, I'm happy to assume that GoodEnough is able to accurately predict what questions Predict will answer poorly. Even though my question-answering task takes place over a series of disconnected days, I'm happy to assume that GoodEnough can make predictions that span many days. And I'm happy to apply Predict to prediction problems where we condition on observations out-of-order, e.g. to ask "what will we see today, given that tomorrow we see X?", and to assume that GoodEnough makes reasonable predictions in these cases.

(For concreteness' sake, I'll suggest a precise model for Predict in the final section.)

I'm also willing to make arbitrary changes to the prediction procedure, as long as it remains feasible. As a silly example, rather than directly consulting Predict to answer a question Q, I would be willing to use Predict to predict Predict's answer to Q. Likewise, I'm willing to produce training data points in any way that works, rather than simply picking one of the questions from the day. For example, I would be willing to use Predict to sample a question from some other distribution.

The solution must work when the errors made by Predict are adversarial.

Opacity

If Predict works by maintaining an explicit distribution over hypotheses, at least 1/1000th of which are competitive with GoodEnough, then we can use standard techniques to obtain ≤1000 bad days: on each day, we get feedback on any maximally contentious training point. But it's not clear how to extend this approach to general prediction algorithms.

One tempting approach is to use Predict to generate questions that Predict will answer poorly. Then if Predict really does answer poorly, we make rapid progress on training. And if Predict doesn't answer poorly, but there actually was some question that it *would have* answered poorly, then we also make progress on training (though progress may be exponentially slower in this case, which is where the 2^R comes from).

The problem with this approach is that it is hard to tell whether a question was "answered poorly" in a way that is useful for active learning, without having explicit access to the model class. What we really care about is the gap between Predict's performance and GoodEnough's performance, but there is no clear way to use Predict to estimate that gap, or to pick a question that has a large gap. There are a number of standard notions used to guide active learning, but none of them seem to work in the worst case.

Prior art?

This problem seems pretty natural, but I'm not aware of any existing work that speaks directly to it. The most distinctive characteristics of this problem are:

- Making minimal assumptions about the model-to-be-learned, and about the learning procedure itself.
- Getting worst-case guarantees. (Most work with worst-case guarantees makes very strong assumptions about the model-to-be-learned, e.g. that it's a linear threshold function.)
- Focusing on very powerful learners that can make reasonable predictions about its own future behavior. (This one makes the problem easier; you need to do something to make the problem easier if you want to handle the other two properties.)

It may well be that this problem is thoroughly impossible, or that any solution requires some additional assumptions. Impossibility results, or positive results under extra assumptions, would be great.

Tree-structured predictions

Rather than considering an unstructured list of N predictions, I am especially interested in what I'll call the "tree-structured" setting, where N may be extremely large.

Each day there is a distinct prediction problem for each string of ≤100 (=k) bits. Of course we won't actually use all of these exponentially many predictions in a single day, but we'll assume that we can get a bad outcome for a day just because our predictor *would* make one of these predictions badly if it came up.

For a simple example where this might happen, suppose that the user is estimating how many 100 bit primes there are. The prediction problem associated to the string s is "how many 100 bit primes are there whose highest-order bits are s?"

When faced with this question, the human might ask the system "how many 100 bit primes are there whose highest-order bits are $s \circ b$?" for b = 0, 1. E.g. in order to answer the question "how many 100 bit primes are there that start with 10?" the human might ask "how many 100 bit primes are there that start with 101?" And so on.

Suppose that our predictor behaves badly when asked "how many 100 bit primes are there that start with 100100?", outputting an unreasonably large number. If the predictor not only predicts badly in this case, but can *predict* that it will predict badly in this case, then that error will propagate up and lead to a bad answer to "how many 100 bit primes are there?"

We can pose our original active learning problem in the tree-structured case. The goal and assumptions are essentially the same: we assume that if our predictor would make acceptably good predictions at every question in the tree, then we will have a good day. We try to minimize the number of bad days by cleverly choosing questions on which to train our predictor.

This version of the problem seems harder to formalize well, and seems much harder to solve well. If the problem is impossible, I think this represents a real challenge to this bootstrapping scheme. If the problem has a clean solution, I think that solution is likely to be interesting to a much broader audience.

Models for good predictors

We could imagine that Predict is a Solomonoff inductor and GoodEnough is a physical model of the human and their environment.

Obviously such strong prediction is implausible, but an approach that worked with very good predictors may well work under some more plausible assumptions. A positive result for very powerful predictors

would also be a “lower bound for lower bounds”—it would rule out a realistic impossibility result.

Solomonoff induction has roughly the right properties, but it is not generally able to make predictions about itself or about larger systems of which it is a part. Such self-prediction seems essential to solving the above problem, so it would be good to use a model that was capable of reasoning about itself.

We can get a better (though still way-too-powerful) model by considering an analogous “reflective oracle machine,” which is able to obtain the same guarantees as Solomonoff induction while making predictions about environments that are computationally rich enough to implement the reflective oracle machine itself. The formal details aren’t especially important, and you wouldn’t go far wrong by just pretending that Solomonoff induction could make good (probabilistic) predictions about environments containing Solomonoff inductors.