Ignoring computational limitations with reflective oracles

Paul Christiano Follow
Oct 4, 2016 · 5 min read

It feels like a major difficulty in AI control is *defining what we want to do*, in a sense that is nearly independent of the algorithmic problem of *doing it efficiently*. It might be useful to explicitly separate out these two parts of the problem.

Coping with computational limitations seems to be the "hard part" of AI, and we don't yet know what a solution will look like. If we are able to isolate the part of AI control that *doesn't* depend on computational limitations, it might be easier to work on it long before we know what practical AI systems will look like.

To this end, it would be useful to have a natural and theoretically clean model of computation in which there aren't meaningful computational limitations. We could then try to address AI control in that model, or to understand the properties of optimal algorithms in that model.

(This post is largely a review of these papers.)

Defining reflective [evaluation] oracles

Reflective oracles are introduced in this paper. The model in that paper is a little bit complicated for technical reasons, but basically:

- You give a reflective oracle a (randomized) program F with binary outputs.
- It gives you a (randomized) output Eval(F) immediately.
- If the program F always returns a value, then Eval(F) samples from the same distribution as F. That is, Eval( ) can instantly run an arbitrarily slow program F.
- If F doesn't always return a value (say sometimes it gets stuck in an infinite loop), then the probability that Eval(F) outputs any bit *b* is at least as high as the probability that F outputs that bit. For example, if F returns **0** with probability 50%, returns **1** with

probability 30%, and hangs with probability 20%, then Eval(F) can return **0** with any probability between 50% and 70%.

- **The program F can itself make calls to Eval( ).**

In the paper, we prove that such an oracle exists, despite apparent problems with self-reference.

For example, if we consider the program Q := "Run Eval(Q) then return the opposite," then Eval(Q) simply returns a uniform distribution over {0, 1}, and hence Q returns a uniform distribution over {1, 0}.

Of course there is (probably) no physical way to implement such a reflective oracle. But it might nevertheless be a useful model to consider.

Reflective oracles as a model of unlimited resources

Writing programs that can use reflective oracles is a way of ignoring computational limitations.

Reflective oracles have many nice properties:

- Any boolean function that can be implemented using a reflective oracle can be implemented with a single application of a reflective oracle. So in a world with reflective oracles, there is no such thing as "not having enough time to find the answer." Note that this isn't true for machines with a halting oracle, or any other non-trivial model of computation that had been previously proposed: calling a halting oracle twice lets you compute more stuff than calling a halting oracle once, and hypercomputers face their own hypercomputational limits.
- Even if the laws of physics were strange enough to permit reflective oracles, reflective oracles would still suffice for doing perfect physical simulations. This also isn't true for a halting oracle: if the universe could contain halting oracles, then the physical prediction problem must be so difficult that it can't be solved using *merely* a halting oracle. If we try to write "really slow algorithms" we can run into a similar difficulty: by assuming that our algorithms can spend more computing time than anything else in the environment, then we can inadvertently rule out the case where our algorithms themselves are an important object to reason about.

- Reflective oracles are strictly weaker than a halting oracle. This isn't a big deal, but it's still nice that we are moving in the right direction.

Example: game theory

Game theory is another example of a domain where it is convenient to ignore computational limitations—in the real world playing games is incredibly messy, but if we ignore computational limitations (and irrationality and etc.) we get a very clean and simple theory.

The usual approach is to completely abstract away the computations performed by the agents themselves, and just reason about the properties that we feel the agents' strategies *should have*, if they were rational and had no computational limitations. In some cases this is unsatisfying: it treats the players separately from the rest of nature, forcing game theory into a separate magisterium. It makes it hard to reason about alternative decision procedures. It prevents us from considering worlds that contain both agents and other comparably powerful computational processes. And so on.

Intuitively, game theory should fall out naturally from causal decision theory when we ignore computational limitations. And indeed, in the paper that introduces reflective oracles we show that this occurs.

Example: RL

Reinforcement learning in the real world is extremely complicated. Again, much of this complication is due to computational limits.

Without computational limits, we can implement model-based RL by starting with a broad prior, doing Bayesian reasoning, and then planning with a brute-force search.

AIXI is an implementation of this idea using a halting oracle. I think that AIXI has helped the field reason about powerful RL at least somewhat: it refines our sense of what we are trying to do, lets us make concrete statements about how powerful RL agents might behave, and lets us examine potential limitations of model-based RL agents that would persist even if we solved all of the hard computational problems in AI. (I also think that AIXI has delivered good bang for the buck—even if it hasn't improved our understanding that much in absolute terms, it also represents a very small fraction of cognitive effort spent on understanding AI.)

Using reflective oracles, researchers at MIRI have defined a "reflective" version of AIXI. Because it uses only on reflective oracles this algorithm is weaker than AIXI itself. But by the same token, it is also easier to reason about, and in particular it can reason successfully about itself or about environments that contain itself. Since it doesn't need to treat itself as a distinguished part of the environment, it can: (1) reason about events that might change its own code or memories, (2) reason about other comparably powerful agents (and in that setting learns to play Nash equilibria), (3) potentially be generalized to different algorithms that better handle the fact that they are embodied within the environment rather than separate from it.

Conclusion

If we are interested in reasoning about the behavior of powerful agents that make very effective use of computational resources, it may be useful to consider simplified models in which there are no meaningful computational limits. I believe that reflective oracles are a pretty good model for these purposes, and certainly that they are the most natural model currently available.