Stable self-improvement as an AI safety problem

Paul Christiano Follow

Jan 22, 2015 · 8 min read

"Stable self-improvement" seems to be a primary focus of MIRI's work. As I understand it, the problem is "How do we build an agent which rationally pursues some goal, is willing to modify itself, and with very high probability continues to pursue the same goal after modification?"

The key difficulty is that it is impossible for an agent to formally "trust" its own reasoning, i.e. to believe that "anything that I believe is true." Indeed, even the natural concept of "truth" is logically problematic. But without such a notion of trust, why should an agent even believe that its own continued existence is valuable?

I agree that there are open philosophical questions concerning reasoning under logical uncertainty, and that reflective reasoning highlights some of the difficulties. But I am not yet convinced that stable self-improvement is an especially important problem for AI safety; I think it would be handled correctly by a human-level reasoner as a special case of decision-making under logical uncertainty. This suggests that (1) it will probably be resolved en route to human-level AI, (2) it can probably be "safely" delegated to a human-level AI. I would prefer use energy investigating other aspects of the AI safety problem.

• • •

Consider an agent A which shares our values and is able to reason "as well as we are"—for any particular empirical or mathematical quantity, A's estimate of its expectation is as good as ours. For notational convenience, suppose that A's preferences are the same as "our" preferences, and let U be the associated utility function.

Now suppose that A is thinking about an outcome including the existence of an agent B. (Perhaps B is a new AI that A is considering designing; perhaps B is a version of A that has made some further observations; whatever.) We'd like the agent to evaluate this outcome

on its merits. It should think about how good the existence of B is. If B also maximizes U, then A should correctly understand that B's existence will tend to be good.

The expected value of U conditioned on this outcome is just another empirical quantity. If A is as good at estimation as humans, then it won't predictably over- or under-estimate this quantity. And so it will weigh B's existence correctly when considering the consequences of its actions.

So if we really had a "human-level" reasoner in the sense I assumed at the outset, our problem would be solved. There are a number of reasons to think the problem might be important anyway. I haven't seen any of these arguments fleshed out in much detail, and for the most part I am skeptical.

Self-modification requires high confidence

If we anticipate a long sequence of ever-more-powerful AI's, then we might want to be *very* sure that each change is really an improvement. There are two sides to this concern.

First is the idea that an AI might not exercise sufficient caution when designing a successor. But if the AI has well-calibrated beliefs and shares our values, then by construction it will make the appropriate tradeoffs between reliability and efficiency. So I don't take this concern very seriously.

Second is the concern that, if the required confidence is very high, then it might be very difficult to be confident enough to go ahead with a proposed AI design. In this scenario, an AI might correctly realize that it should not make any risky changes; but this restriction might introduce unacceptable efficiency losses. While the "good guys" proceed cautiously, competitors will race ahead (allowing their systems' values to change over time).

On this view, by working out these issues farther in advance we can save some time for the "good guys," or push research in a direction which makes their task easier.

But this problem can be straightforwardly delegated to machine intelligences. An AI with human-level reasoning is also human-level at assessing the reliability of a system or engineering a highly-reliable

system. We are left with a quantitative question: how useful is doing this work in advance, rather than doing it as its needed?

My intuition is that it's not too valuable, and I don't think anyone has yet made a strong argument for the other side. I think the main disagreement is whether this is an extra-especially hard problem. Briefly, I think:

1. The biggest concerns are from design errors that would conceal themselves (presumably because the resulting AI becomes motivated to conceal them). It's hard to make changes that are bug-free, but it's probably much easier to avoid this very specific kind of bug, which (1) leaves the program still behaving almost exactly as intended, but (2) subtly changes what it's optimizing.

2. You can achieve relatively high confidence using normal empirical methods. Bugs arise in unanticipated and untested situations, but those aren't mostly the kinds of bugs you are worried about (see #1). It is unlikely that an unanticipated situation would cause bugs for several different supervisory systems at once.

3. Previous versions of a system can literally watch their successors operate and ensure that new changes function as intended. New systems can be deployed in such a way that deception is very hard to conceal (even the cost-benefit analysis necessary to decide when to deceive is presumably the result of some calculation which can be monitored).

4. Before AI is doing most of the AI design work, bugs are not catastrophic. Once AI is doing most of the AI design work, you don't have to make many more changes before you have massively increased your AI design capabilities (obsoleting your earlier work). For example, 10 improvements of +30% efficiency (which is a relatively small improvement for a significant software change) results in a >13x total improvement. So talking about "billions" of changes is misleading, in the most extreme case we should maybe talk about "tens" of changes.

5. Applying changes in batches rather than continuously doesn't really matter to productivity. Waiting until you have assembled a +30% productivity increase before deploying a package of changes reduces your speed by less than 25%, probably much less.

6. Just because a change can be decomposed as many smaller changes doesn't mean its probability of failure is overwhelming. Normal software often fails, but each change is composed of a large number of small features. Moreover, the failures of many related changes are highly correlated.

7. If a proposed change results in only a small productivity benefit, then it is particularly easy for the original system to provide end-to-end oversight for the improved version, e.g. listening to its arguments and evaluating its final product. And there should be *very* few changes with big productivity benefits before you have obsoleted your original investment.

8. Bugs are most challenging in a security setting, where adversaries search for improbable corner cases—when your buffer overflows, you don't normally expect it to overwrite your code in a subtle way, unless you are facing an adversary. There is generally no adversary in AI design, and I don't know of any promising approaches to AI safety that involve hardening software against even an implicit adversary. The closest you get is the AI's adversarial desire to maximize its goals; I don't think this is relevantly similar, but I'm happy to have an argument about it.

9. There are many pairs of "nearby" agents such that one is "friendly" and the other "unfriendly." In these cases it might be easy to make a programming misstep that takes you from one category to the other. But this is nearly irrelevant; we get to choose what AI we design, and in particular we can focus on designs that are robustly good (and which are separated from adversarial deceivers by a chasm of non-functional or obviously malignant intermediates).

Standards of reasoning can't be outsourced

When we design an AI, we are (at least implicitly) specifying what kind of reasoning it considers "valid." If we get the answer wrong, for example by leaving out some important pattern of reasoning X that humans accept, then the problem might be permanent: our first AI thinks that accepting X is an error, so it designs successors that also reject X. In principle, the result could be a system which is an effective reasoner but which is not able to reason about its own behavior.

I'm skeptical. An effective reasoner interested in making empirical predictions will tend to (provisionally) accept whatever patterns of reasoning lead to correct predictions. This can include the laws of arithmetic just as well as it can include natural laws. (See my writeup of this view here.) If some pattern of reasoning X is important for making accurate predictions then an effective AI will accept X, at least as suggestive evidence. If this is how human reasoning works, than any sufficiently effective reasoner would recover the same patterns of reasoning.

I would be surprised if, in contrast to this view, human brains were simply constituted to automatically accept certain rules of logic. Certainly the history of logic and mathematics suggests that rules of reasoning are subject to debate, and are developed to fit the empirical facts. And even if human brains are wired to reason logically, they were produced by natural selection (which most definitely wasn't).

There may be a remaining problem in understanding how a system can learn to treat a pattern of reasoning as a useful source of evidence, and more generally where does human logical reasoning come from. I think these are interesting questions, but (1) they are quite distant from the current approach to stable self-improvement, (2) I suspect they have to be resolved to produce human-level reasoners.

Even more clear is that humans don't have any kind of axiomatic "self-trust;" they trust themselves, to the extent they do, based on empirical observations of their own trustworthiness. This brings us to…

Self-improvement highlights open questions about reasoning

In some sense self-modification is just a special case of reasoning under logical uncertainty. But we don't understand reasoning under logical uncertainty in general; reflective reasoning might be a productive challenge problem for thinking about logical uncertainty. I agree with this, but it doesn't seem to be the motivation for MIRI's research program.

For example, I think that the attitude an agent has towards its own judgments should be similar to the attitude it has towards the views of wise peers in general. These views aren't characterized by strong, monotonic forms of trust (such that if a peer says X, I believe X regardless of what other evidence is available). Instead, I view my

peers' judgment as evidence—potentially strong evidence, depending on how much I trust them—which might be overturned by new considerations.

On this perspective, Godelian difficulties don't seem like especially problematic cases. If I learn that I assign the sentence X = "I assign this sentence less than 50% probability" a probability of 49%, then I change my mind and start believing X. Similarly, if I learn "I assign Y a probability of 49%; also Y" then I change my mind and start believing Y. Though I trust myself, if I learn further evidence it can screen off my previous beliefs. Sometimes, knowing exactly what I believe can be further evidence in and of itself, screening off the content of those beliefs. (See here for a discussion of my views.)

It's still unclear how an agent learns that some source is trustworthy; what the content of that knowledge is, how it works, what kind of framework can accommodate it. And it would be great to understand how the output of a reasoning process can constitute evidence (as a deductive fact, rather than an inductive generalization or as an axiom). But these are rather different questions, which one would attack with rather different techniques.

In contrast, techniques for "licensing" the creation of successors are not promising on this perspective unless they correspond to actual improvements in an agent's predictions.