Iterated Amplification

1. Preface to the sequence on iterated amplification
2. The Steering Problem
3. Clarifying "AI Alignment"
4. An unaligned benchmark
5. Prosaic AI alignment
6. Approval-directed agents
7. Approval-directed bootstrapping
8. Humans Consulting HCH
9. Corrigibility
10. Iterated Distillation and Amplification
11. Benign model-free RL
12. Factored Cognition
13. Supervising strong learners by amplifying weak experts
14. AlphaGo Zero and capability amplification
15. Directions and desiderata for AI alignment
16. The reward engineering problem
17. Capability amplification
18. Learning with catastrophes
19. Thoughts on reward engineering
20. Techniques for optimizing worst-case performance
21. Reliability amplification
22. Security amplification
23. Meta-execution

Preface to the sequence on iterated amplification

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

This sequence describes iterated amplification, a possible strategy for building an AI that is actually trying to do what we want out of ML systems trained by gradient descent.

Iterated amplification is not intended to be a silver bullet that resolves all of the possible problems with AI; it's an approach to the particular alignment problem posed by scaled-up versions of modern ML systems.

Iterated amplification is based on a few key hopes

- If you have an overseer who is smarter than the agent you are trying to train, you can safely use that overseer's judgment as an objective.
- We can train an RL system using very sparse feedback, so it's OK if that overseer is very computationally expensive.
- A team of aligned agents may be smarter than any individual agent, while remaining aligned.

If all of these hopes panned out, then at every point in training "a team of the smartest agents we've been able to train so far" would be a suitable overseer for training a slightly smarter aligned successor. This could let us train very intelligent agents while preserving alignment (starting the induction from an aligned human).

Iterated amplification is still in an preliminary state and is best understood as a research program rather than a worked out solution. Nevertheless, I think it is the most concrete existing framework for aligning powerful ML with human interests.

Purpose and audience

The purpose of this sequence is to communicate the basic intuitions motivating iterated amplification, to define iterated amplification, and to present some of the important open questions.

I expect this sequence to be most useful for readers who would like to have a somewhat detailed understanding of iterated amplification, and are looking for something more structured than ai-alignment.com to help orient themselves.

The sequence is intended to provide enough background to follow most public discussion about iterated amplification, and to be useful for building intuition and informing research about AI alignment even if you never think about amplification again.

The sequence will be easier to understand if you have a working understanding of ML, statistics, and online learning, and if you are familiar with other work on AI alignment. But it would be reasonable to just dive in and just skip over any detailed discussion that seems to depend on missing prerequisites.

Outline and reading recommendations

- The first part of this sequence clarifies the problem that iterated amplification is trying to solve, which is both narrower and broader than you might expect.
- The second part of the sequence outlines the basic intuitions that motivate iterated amplification. I think that these intuitions may be more important than the scheme itself, but they are considerably more informal.
- The core of the sequence is the third section. Benign model-free RL describes iterated amplification, as a general framework into which we can substitute arbitrary algorithms for reward learning, amplification, and robustness. The first four posts all describe variants of this idea from different perspectives, and if you find that one of those descriptions is clearest for you then I recommend focusing on that one and skimming the others.
- The fourth part of the sequence describes some of the black boxes in iterated amplification and discusses what we would need to do to fill in those boxes. I think these are some of the most important open questions in AI alignment.
- The fifth section of the sequence breaks down some of these problems further and describes some possible approaches.
- The final section is an FAQ by Alex Zhu, included as appendix.

The sequence is not intended to be building towards a big reveal---after the first section, each post should stand on its own as addressing a basic question raised by the preceding posts. If the first section seems uninteresting you may want to skip it; if future sections seem uninteresting then it’s probably not going to get any better.

Some readers might prefer starting with the third section, while being prepared to jump back if it’s not clear what’s going on or why. (It would still make sense to return to the first two sections after reading the third.)

If you already understand iterated amplification you might be interested in jumping around the fourth and fifth sections to look at details you haven’t considered before.

The posts in this sequence link liberally to each other (not always in order) and to outside posts. The sequence is designed to make sense when read in order without reading other posts, following links only if you are interested in more details.

Tomorrow's AI Alignment Forum sequences post will be 'Future directions for ambitious value learning' by Rohin Shah, in the sequence 'Value Learning'.

The next post in this sequence will come out on Tuesday 13th November, and will be 'The Steering Problem' by Paul Christiano.

The Steering Problem

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

Most AI research focuses on reproducing human abilities: to learn, infer, and reason; to perceive, plan, and predict. There is a complementary problem which (understandably) receives much less attention: if you *had* these abilities, what would you do with them?

The steering problem: Using black-box access to human-level cognitive abilities, can we write a program that is as useful as a well-motivated human with those abilities?

This post explains what the steering problem is and why I think it's worth spending time on.

Introduction

A capable, well-motivated human can be extremely useful: they can work without oversight, produce results that need not be double-checked, and work towards goals that aren't precisely defined. These capabilities are critical in domains where decisions cannot be easily supervised, whether because they are too fast, too complex, or too numerous.

In some sense "be as useful as possible" is just another task at which a machine might reach human-level performance. But it is different from the concrete capabilities normally considered in AI research.

We can say clearly what it means to "predict well," "plan well," or "reason well." If we ignored computational limits, machines could achieve any of these goals today. And before the existing vision of AI is realized, we must *necessarily* achieve each of these goals.

For now, "be as useful as possible" is in a different category. We can't say exactly what it means. We could not do it no matter how fast our computers could compute. And even if we resolved the most salient challenges in AI, we could remain in the dark about this one.

Consider a capable AI tasked with running an academic conference. How should it use its capabilities to make decisions?

- We could try to specify exactly what makes a conference good or bad. But our requirements are complex and varied, and so specifying them exactly seems time-consuming or impossible.
- We could build an AI that imitates successful conference organizers. But this approach can never do any better than the humans we are imitating. Realistically, it won't even match human performance unless we somehow communicate what characteristics are important and why.

- We could ask an AI to maximize our satisfaction with the conference. But we'll get what we measure. An extensive evaluation would greatly increase the cost of the conference, while a superficial evaluation would leave us with a conference optimized for superficial metrics.

Everyday experience with humans shows how hard delegation can be, and how much easier it is to assign a task to someone who actually cares about the outcome.

Of course there is already pressure to write *useful* programs in addition to smart programs, and some AI research studies how to efficiently and robustly communicate desired behaviors. For now, available solutions apply only in limited domains or to weak agents. The steering problem is to close this gap.

Motivation

A system which "merely" predicted well would be extraordinarily useful. Why does it matter whether we know how to make a system which is “as useful as possible”?

Our machines will probably do *some* things very effectively. We know what it means to "act well" in the service of a given goal. For example, using human cognitive abilities as a black box, we could probably design autonomous corporations which very effectively maximized growth. If the black box was cheaper than the real thing, such autonomous corporations could displace their conventional competitors.

If machines can do everything equally well, then this would be great news. If not, society’s direction may be profoundly influenced by what can and cannot be done easily. For example, if we can only maximize what we can precisely define, we may inadvertently end up with a world filled with machines trying their hardest to build bigger factories and better widgets, uninterested in anything we consider intrinsically valuable.

All technologies are more useful for some tasks than others, but machine intelligence might be particularly problematic because it can entrench itself. For example, a rational profit-maximizing corporation might distribute itself throughout the world, pay people to help protect it, make well-crafted moral appeals for equal treatment, or campaign to change policy. Although such corporations could bring large benefits in the short term, in the long run they may be difficult or impossible to uproot, even once they serve no one’s interests.

Why now?

Reproducing human abilities gets a lot of deserved attention. Figuring out exactly what you’d do once you succeed feels like planning the celebration before the victory: it might be interesting, but why can’t it wait?

1. **Maybe it’s hard**. Probably the steering problem is much easier than the AI problem, but it might turn out to be surprisingly difficult. If it *is* difficult, then learning that earlier will help us think more clearly about AI, and give us a head start on addressing it.
2. **It may help us understand AI.** The difficulty of saying exactly what you want is a basic challenge, and the steering problem is a natural perspective on this challenge. A little bit of research on natural theoretical problems is often

worthwhile, even when the direct applications are limited or unclear. In section 4 we discuss possible approaches to the steering problem, many of which are new perspectives on important problems.

3. **It should be developed alongside AI.** The steering problem is a long-term goal in the same way that understanding human-level prediction is a long-term goal. Just as we do theoretical research on prediction before that research is commercially relevant, it may be sensible to do theoretical research on steering before it is commercially relevant. Ideally, our ability to build useful systems will grow in parallel with our ability to build capable systems.
4. **Nine women can't make a baby in one month.** We could try to save resources by postponing work on the steering problem until it seems important. At this point it will be easier to work on the steering problem, and if the steering problem turns out to be unimportant then we can avoid thinking about it altogether.
   But at large scales it becomes hard to speed up progress by increasing the number of researchers. Fewer people working for longer may ultimately be more efficient (even if earlier researchers are at a disadvantage). This is particularly pressing if we may eventually want to invest much more effort in the steering problem.
5. **AI progress may be surprising**. We probably won't reproduce human abilities in the next few decades, and we probably won't do it without ample advance notice. That said, AI is too young, and our understanding too shaky, to make confident predictions. A mere 15 years is 20% of the history of modern computing. If important human-level capabilities are developed surprisingly early or rapidly, then it would be worthwhile to better understand the implications in advance.
6. **The field is sparse**. Because the steering problem and similar questions have received so little attention, individual researchers are likely to make rapid headway. There are perhaps three to four orders of magnitude between basic research on AI and research directly relevant to the steering problem, lowering the bar for arguments 1-5.

In section 3 we discuss some other reasons not to work on the steering problem: Is work done now likely to be relevant? Is there any concrete work to do now? Should we wait until we can do experiments? Are there adequate incentives to resolve this problem already?

Defining the problem precisely

Recall our problem statement:

The steering problem: Using black-box access to human-level cognitive abilities, can we write a program that is as useful as a well-motivated human with those abilities?

We'll adopt a particular human, Hugh, as our "well-motivated human:" we'll assume that we have black-box access to Hugh-level cognitive abilities, and we'll try to write a program which is as useful as Hugh.

Abilities

In reality, AI research yields complicated sets of related abilities, with rich internal structure and no simple performance guarantees. But in order to do concrete work in advance, we will model abilities as black boxes with well-defined contracts.

We're particularly interested in tasks which are "AI complete" in the sense that human-level performance on that task could be used as a black box to achieve human-level performance on a very wide range of tasks. For now, we'll further focus on domains where performance can be unambiguously defined.

Some examples:

- **Boolean question-answering**. A question-answerer is given a statement and outputs a probability. A question-answerer is Hugh-level if it never makes judgments predictably worse than Hugh's. We can consider question-answerers in a variety of languages, ranging from natural language ("Will a third party win the US presidency in 2016?") to precise algorithmic specifications ("Will this program output 1?").
- **Online learning**. A function-learner is given a sequence of labelled examples (x, y) and predicts the label of a new data point, x'. A function-learner is Hugh-level if, after training on any sequence of data (xi, yi), the learner's guess for the label of the next point xi+1 is---on average---at least as good as Hugh's.
- **Embodied reinforcement learning**. A reinforcement learner interacts with an environment and receives periodic rewards, with the goal of maximizing the discounted sum of its rewards. A reinforcement learner is Hugh-level if, following any sequence of observations, it achieves an *expected* performance as good as Hugh's in the subsequent rounds. The expectation is taken using our subjective distribution over the physical situation of an agent who has made those observations.

When talking about Hugh's predictions, judgments, or decisions, we imagine that Hugh has access to a reasonably powerful computer, which he can use to process or display data. For example, if Hugh is given the binary data from a camera, he can render it on a screen in order to make predictions about it.

We can also consider a particularly degenerate ability:

- **Unlimited computation**. A box that can run any algorithm in a single time step is--in some sense--Hugh level at every precisely stated task.

Although unlimited computation seems exceptionally powerful, it's not immediately clear how to solve the steering problem even using such an extreme ability.

Measuring usefulness

What does it mean for a program to be "as useful" as Hugh?

We'll start by defining "as useful for X as Hugh," and then we will informally say that a program is "as useful" as Hugh if it's as useful for the tasks we care most about.

Consider **H,** a black box that simulates Hugh or perhaps consults a version of Hugh who is working remotely. We'll suppose that running **H** takes the same amount of time as consulting our Hugh-level black boxes. A project to accomplish X could potentially use as many copies of **H** as it can afford to run.

A program **P** is more useful than Hugh for X if, for every project using **H** to accomplish X, we can efficiently transform it into a new project which uses **P** to accomplish X. The new project shouldn't be much more expensive---it shouldn't take much longer, use much more computation or many additional resources, involve much more human labor, or have significant additional side-effects.

Well-motivated

What it does it mean for Hugh to be well-motivated?

The easiest approach is universal quantification: for *any* human Hugh, if we run our program using Hugh-level black boxes, it should be as useful as Hugh.

Alternatively, we can leverage our intuitive sense of what it means for someone to be well-motivated to do X, and define "well-motivated" to mean "motivated to help the user's project succeed."

Scaling up

If we are given better black boxes, we should make a better program. This is captured by the requirement that our program should be as useful as Hugh, no matter how capable Hugh is (as long as the black boxes are equally capable).

Ideally, our solutions should scale far past human-level abilities. This is not a theoretical concern---in many domains computers already have significantly superhuman abilities. This requirement is harder to make precise, because we can no longer talk about the "human benchmark." But in general, we would like to build systems which are (1) working towards their owner's interests, and (2) nearly as effective as the best goal-directed systems that can be built using the available abilities. The ideal solution to the steering problem will have these characteristics in general, even when the black-box abilities are radically superhuman.

This is an abridged version of this document from 2014; most of the document is now superseded by later posts in this sequence.

Tomorrow's AI Alignment Forum sequences post will be 'Embedded Agency (text)' in the sequence Embedded Agency, by Scott Garrabrant and Abram Demski.

The next post in this sequence will come out on Thursday 15th November, and will be 'Clarifying "AI Alignment"' by Paul Christiano.

Clarifying "AI Alignment"

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

When I say an AI A is *aligned with* an operator H, I mean:

> *A is trying to do what H wants it to do.*

The “alignment problem” is the problem of building powerful AI systems that are aligned with their operators.

This is significantly narrower than some other definitions of the alignment problem, so it seems important to clarify what I mean.

In particular, this is the problem of getting your AI to try to do the right thing, **not** the problem of figuring out which thing is right. An aligned AI would try to figure out which thing is right, and like a human it may or may not succeed.

Analogy

Consider a human assistant who is trying their hardest to do what H wants.

I’d say this assistant is aligned with H. If we build an AI that has an analogous relationship to H, then I’d say we’ve solved the alignment problem.

“Aligned” doesn’t mean “perfect:”

- They could misunderstand an instruction, or be wrong about what H wants at a particular moment in time.
- They may not know everything about the world, and so fail to recognize that an action has a particular bad side effect.
- They may not know everything about H’s preferences, and so fail to recognize that a particular side effect is bad.
- They may build an unaligned AI (while attempting to build an aligned AI).

I use alignment as a statement about the *motives* of the assistant, not about their knowledge or ability. Improving their knowledge or ability will make them a better assistant—for example, an assistant who knows everything there is to know about H is less likely to be mistaken about what H wants—but it won’t make them *more aligned.*

(For very low capabilities it becomes hard to talk about alignment. For example, if the assistant can’t recognize or communicate with H, it may not be meaningful to ask whether they are aligned with H.)

Clarifications

- The definition is intended *de dicto* rather than *de re*. An aligned A is trying to “do what H wants it to do.” Suppose A thinks that H likes apples, and so goes to the store to buy some apples, but H really prefers oranges. I’d call this behavior

aligned because A is trying to do what H wants, even though the thing it is trying to do ("buy apples") turns out not to be what H wants: the *de re* interpretation is false but the *de dicto* interpretation is true.

- An aligned AI can make errors, including moral or psychological errors, and fixing those errors isn't part of my definition of alignment except insofar as it's part of getting the AI to "try to do what H wants" *de dicto*. This is a critical difference between my definition and some other common definitions. I think that using a broader definition (or the *de re* reading) would also be defensible, but I like it less because it includes many subproblems that I think (a) are much less urgent, (b) are likely to involve totally different techniques than the urgent part of alignment.
- An aligned AI would also be trying to do what H wants **with respect to clarifying H's preferences**. For example, it should decide whether to ask if H prefers apples or oranges, based on its best guesses about how important the decision is to H, how confident it is in its current guess, how annoying it would be to ask, *etc.* Of course, it may also make a mistake at the meta level—for example, it may not understand when it is OK to interrupt H, and therefore avoid asking questions that it would have been better to ask.
- This definition of "alignment" is extremely imprecise. I expect it to correspond to some more precise concept that cleaves reality at the joints. But that might not become clear, one way or the other, until we've made significant progress.
- One reason the definition is imprecise is that it's unclear how to apply the concepts of "intention," "incentive," or "motive" to an AI system. One naive approach would be to equate the incentives of an ML system with the objective it was optimized for, but this seems to be a mistake. For example, humans are optimized for reproductive fitness, but it is wrong to say that a human is incentivized to maximize reproductive fitness.
- "What H wants" is even more problematic than "trying." Clarifying what this expression means, and how to operationalize it in a way that could be used to inform an AI's behavior, is part of the alignment problem. Without additional clarity on this concept, we will not be able to build an AI that tries to do what H wants it to do.

Postscript on terminological history

I originally described this problem as part of "the AI control problem," following Nick Bostrom's usage in *Superintelligence*, and used "the alignment problem" to mean "understanding how to build AI systems that share human preferences/values" (which would include efforts to clarify human preferences/values).

I adopted the new terminology after some people expressed concern with "the control problem." There is also a slight difference in meaning: the control problem is about coping with the possibility that an AI would have different preferences from its operator. Alignment is a particular approach to that problem, namely avoiding the preference divergence altogether (so excluding techniques like "put the AI in a really secure box so it can't cause any trouble"). There currently seems to be a tentative consensus in favor of this approach to the control problem.

I don't have a strong view about whether "alignment" should refer to this problem or to something different. I do think that *some* term needs to refer to this problem, to separate it from other problems like "understanding what humans want," "solving philosophy," *etc.*

This post was originally published here on 7th April 2018.

The next post in this sequence will post on Saturday, and will be "An Unaligned Benchmark" by Paul Christiano.

Tomorrow's AI Alignment Sequences post will be the first in a short new sequence of technical exercises from Scott Garrabrant.

An unaligned benchmark

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

My goal is to design AI systems that are aligned with human interests and competitive with unaligned AI.

I find it useful to have a particular AI algorithm in mind. Then I can think about how that algorithm could cause trouble, and try to find a safer variant.

I think of the possibly-unaligned AIs as a benchmark: it's what AI alignment researchers need to compete with. The further we fall short of the benchmark, the stronger the competitive pressures will be for everyone to give up on aligned AI and take their chances.

I have a few standard benchmarks I keep in mind. This post describes one of those benchmarks. It also tries to lay out clearly why I think that benchmark is unsafe, and explains how I think my current research could make a safe version.

I. Model-based RL with MCTS

We train three systems in parallel:

- A generative model to sample sequences of observations, conditioned on sequences of actions.
- A reward function that takes as input a sequence of actions and predicted observations and produces a reward.
- A policy and value function which take as input a sequence of observations and produce the next action and an estimate of the future return.

We train the policy and value function using (roughly) the AlphaZero algorithm: Use MCTS to improve the current policy. Update the policy at the root to predict the best move found by MCTS, update the value to predict its predicted value. Use the generative model to sample environment transitions and the reward function (with a small discount rate) to score them.

We train an autoregressive generative model, to maximize the log probability assigned to the actual sequence of actions and observations produced by the AI (with each observation conditioned on the past actions). This isn't actually a good way to train the generative model, but it's not really central to the discussion.

We train the reward function by showing humans sequences of actions and predicted observations, asking them to assign scores, then predicting those scores with supervised learning. We show humans the sequences of actions that look most promising to the system.

There are plenty of details you'd need in order to make this work well, but that's the basic idea. When applied with very powerful networks, it's plausible that this system would be able to decisively outcompete humans. It would be capable performing a

large intelligent search over long sequences of actions to find those that would be rated highly.

II. What goes wrong?

There are two classes of problems:

Problem 1: Bad objective

The goal of the system is to produce (action, observation) sequences that look good to humans. I claim that optimizing this objective faithfully will lead to bad outcomes.

As the system improves, the rationale of many individual actions will become incomprehensible to a human overseer. At this point the only option for a human is to evaluate sequence of observations based on whether the consequences look good.

The observations present a narrow view of the world, and I strongly suspect that the AI will find sequences of actions that make that narrow view look good without actually *being* good.

Control vs. intrinsic goodness. I think there are two strategies for defining a reward function:

1. Reward worlds in which *humans remain in control of the situation*, in which they are able to get accurate information and correct course as needed.
2. Reward worlds in which *intrinsically good things are happening*

Both of these strategies seem unworkable.

Strategy #1: maintaining control. This appears to be unworkable because determining if humans are actually in control is incredibly difficult—at best you can tell whether they appear to be and feel in control. It's very hard to understand if the humans are getting accurate information, if their understanding of the situation is roughly accurate, if their instructions are being faithfully executed, and so on. This is already an incredibly hard problem amongst humans and gets way harder as the AI gets smarter.

This problem is serious because, if there is even a small gap between the "real" notion of being in control (which an AI cares about in order to generate maximally appealing observation sequences further into the future) and a superficial appearance of control (which the reward function incentivizes), then the AI is going to exploit that gap by taking the real notion of control for itself and giving humans the appearance of control. Of course, if the long-term goals of the AI and human are well-aligned then there would be no conflict of interest, but trying to ensure that alignment brings us back to strategy #2 which I will argue is unworkable.

You could potentially notice this problem by looking ahead to the point where humans attempt to exercise control—if the humans don't get what they want, then they must not have really been in control. But evaluating whether they get what they want brings us back to strategy #2.

Strategy #2: evaluating goodness. This appears to be unworkable because it requires evaluating human preferences over very large objects (the whole world) under very complex conditions (in the far future, where we've had to make many decisions about what kind of world we want to have).

I am quite confident that "world which looks appealing on screen" is not the same as "world which is good;" the gap is already large today, and it will grow much larger over time.

Maintaining a fiction. Ultimately the goal of this AI is to make the sensors look like good things are happening in the world. This can be done either by making the world good or by carefully controlling what gets fed to the sensors. In general, there are many fewer constraints on "what gets fed to the sensors" than "what the world is actually like," and so it's going to be possible to produce much more desirable-looking outcomes, much more easily, by controlling the sensors.

If the humans understand what the AI is doing in detail then they can observe the sequence of actions to notice any actions that might corrupt the sensors. But as soon as the AI is engaging in complicated plans this appears to be totally intractable. Even if the AI was human-level, there are many extremely hard-to-detect plans: for example, the AI could collaborate with an external attacker, who will surreptitiously corrupt the sensor. Or the AI could create a copy of itself on an external server who then helped orchestrate a hard-to-detect corruption.

These actions would be very hard to detect even if you were monitoring a human-level intelligence, and the situation seems hopeless for much more powerful searches. Moreover, because you are searching over sequences of actions for those that receive high reward, you are directly applying optimization power to discovering covert attacks.

Other problems. I have a (fairly common) intuition that applying extremely powerful optimization at a target that isn't *quite* what you want will often lead to bad outcomes. The discussion above is not exhaustive, but I think it is illustrative.

Problem 2: distributional shift (and optimization daemons)

Our training procedure produces a policy and value function, most likely represented as (really big) neural networks. At test time, we combine these the policy and value with MCTS to decide on actions.

The value function and policy have been optimized to yield good performance on the data points we've seen so far, as judged by human evaluations. Unfortunately, there are likely to be a very large number of networks that encode the "wrong" goals but which also yield good performance. These networks will generalize poorly, and moreover when they fail to generalize they can result in an extremely powerful optimization process being pointed at the wrong objective.

A story about training. Originally the policy and value function don't encode anything at all. Over time, they begin to encode a complicated soup of heuristics which is correlated with good performance. If we are training sufficiently powerful models we hope they will eventually perform reasoning about outcomes. For example, the policy could learn to backwards chain from heuristics about what is valuable in

order to decide which moves are good. This is what we are trying to do—the policy is *supposed* to backwards chain, it's the only part of the system that can use heuristics in order to prioritize the search.

What humans actually want is somewhat complicated, so it seems quite likely that it's easier for models to pursue a complicated soup of heuristic goals than to understand exactly what we want. This is similar to the way in which humans acquired an extremely rich set of goals even though we were optimized according to evolutionary fitness. This is a complicated question, but I think it's the theoretical picture and I think historical experience with deep learning points tends to support it.

As the system improves, the reward function encourages it to exhibit an increasingly precise understanding of what we want. Unfortunately there are two ways to do this:

- The intended way: adjust the implicit goals baked into the model such that they converge towards "be helpful to humans." In the analogy to humans, this is like humans caring more and more about reproductive fitness (and less and less about things like beauty or fun except insofar as they are useful for reproductive fitness).
- The unintended way: correctly understand that earning human approval is necessary to survival and hence to achieving other goals, and act accordingly. In the analogy to humans, this is like humans continuing to care about beauty and fun, but believing that they need to have kids in order to realize those goals in the long run.

In practice, I expect both of these changes to occur to some extent, ending up with a model that has somewhat wrong goals together with an instrumental desire to appear helpful.

Catastrophic failure. This could lead to a catastrophic failure in a few different ways:

- An attacker deliberately produces inputs that drive our AI off of the training distribution, and it starts pursuing the wrong goals. That AI may then launch a similar attack against other AI systems it has access to, leading to cascading failures (as with a computer virus). Or an attacker may be able to simultaneously compromise a large number of systems.
- As AI systems acquire increasing influence in the world, they necessarily move off the training distribution. Eventually this sparks a failure in some systems. These failures could cause chaos in the world, pushing us further from the training distribution and leading to cascading failures; or they may all be triggered by the same events and so be correlated.

In either case, we could end up with a massive correlated failure of AI systems, where they start effectively maximizing the wrong goals. That looks effectively like a conflict between us and the AI systems we've built (just as a virus might effectively lead to a conflict between you and the computer you bought). If the AI systems either have significant responsibilities, or are much more intelligent than unaided humans, then there may not be any way to recover from this failure.

Problem 1.5: non-robust reward functions

There is another risk at the intersection between robustness and value specification.

We may learn a model of human approval which is accurate on the training distribution, but incorrectly assigns a very high value to some bad outcomes that didn't appear in training. Indeed, recent experience with adversarial examples suggests that our models often have very strange behavior on parts of the input space not visited in training and that this problem can be hard to correct. Presumably some of these inputs would be assigned unusually high values (just as some would be assigned unusually low values).

In order to reach the most pathological cases, the agent needs significant control over its own observations, which in in turn requires control over its environment. So even if the pathological inputs aren't any worse than random on average, the ones that the system is able to reach will systematically be those where it has acquired enough influence to control its own observations.

We can potentially correct this problem by eliciting human feedback for the problematic observations and then adding them to the training set. We're only in serious trouble if the reward function is really hard to fix. In that case, we may instead make it harder and harder for the system to modify its own observations. After a while, we might train a system which is only able to control its observations by physically overpowering us and causing a catastrophe.

III. How the problem might be fixed

I think that my research has a chance of producing an AI that (a) is nearly-as-good as the benchmark, but (b) doesn't do anything terrible.

The main changes are:

- Rather than learning the environment model directly from observation, produce it by amplification. The amplification process can use exactly the same data about the world (treating the data as an external resource), and it can ultimately be distilled into a neural network just like the one you would have trained. The important difference is that we now have a way to link the generative model back to (an astronomically large number of) human-comprehensible operations on human-comprehensible data and a sequence of decoders which can map from the latent space of the generative model to an implicit human-comprehensible representation.
- Rather than learning a reward function from human data, we also train it by amplification (acting on the same representations used by the generative model). Again, we can distill the reward function into a neural network that acts on sequences of observations, but now instead of learning to predict human judgments it's predicting a very large implicit deliberation.
- In addition to training the policy and value function using RL, we use some combination of these techniques to ensure there is *no* series of observations on which the policy behaves extremely badly. For example, we may use amplification in order to understand what the policy is doing, then use that understanding in order to identify possible (pseudo-)inputs on which the policy behaves badly, then add those (pseudo-)inputs to the training set.

Competitiveness depends on three properties:

- Training a predictor using amplification makes predictions at least as good as training a generative model directly.

- The worst-case training for the policy and value function don't interfere significantly with average case performance.
- Amplification doesn't add much overhead to the training process.

Safety is more subtle. It depends on three properties each defined in terms of some as-yet-undefined notion of "bad" behavior (incorrigible is the current leading candidate):

1. On average, the reward function incentivizes behavior which competes effectively and gives the user effective control.
2. There do not exist inputs on which the policy choose an action because it is bad, or the value function outputs a high reward because the prior behavior was bad. (Or else the policy on its own will generate bad consequences.)
3. The reward function never gives a behavior a higher reward because it is bad. (Or else the test-time optimization by MCTS can generate bad behavior.) For example, if the AI deludes the human operator so that the operator can't interfere with the AI's behavior, that behavior can't receive a higher reward even if it ultimately allows the AI to make more money.

My goals

In order to make all of that work, we'd need to solve a few research problems.

Reliability. Some combination of these techniques needs to successfully eliminate all bad behavior (and in particular to control optimization daemons).

Amplification. Amplification needs to be good enough for these three tasks:

- Learn models that make competitive predictions about the world, and which are "understandable" enough that we can…
- Implement a reward function, defined on those models, that correctly rewards competitive behavior and penalizes bad behavior.
- Understand the internals of competitive machine learning systems, well enough to do informed oversight and to help with reliability (if interpretability is an important ingredient for reliability). Probably the key requirement is reliably identifying bad behavior.

(Without introducing significant overhead.)

Understanding bad behavior. In order to do either of the above we need some suitable notion of "bad" behavior, such that:

- Avoiding bad behavior is sufficient to avoid catastrophic outcomes.
- Bad behavior is an act of commission that is "easy" to avoid, such that it can be achieved in the worst case.
- We can learn a reward function over that avoids creating instrumental incentives for bad behavior, e.g. by punishing any bad behavior which played an important role in receiving a high reward. (This is only plausible because our reward function operates on sequences of predicted states, and so if bad behavior is instrumentally useful it must be because the model "knows about" it.)

This post was originally published here, on 11th March 2018.

The next post in this sequence will be "Prosaic AI Alignment" by Paul Christiano, on Tuesday 20th November.

Tomorrow's AI Alignment Forum sequences post will be "Diagonalization Fixed Point Exercises" by Scott Garrabrant and Sam Eisenstat in the sequence "Fixed Points".

Prosaic AI alignment

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

(*Related: a possible stance for AI control.*)

It's conceivable that we will build "prosaic" AGI, which doesn't reveal any fundamentally new ideas about the nature of intelligence or turn up any "unknown unknowns." I think we wouldn't know how to align such an AGI; moreover, in the process of building it, we wouldn't necessarily learn anything that would make the alignment problem more approachable. So I think that understanding this case is a natural priority for research on AI alignment.

In particular, I don't think it is reasonable to say "we'll know how to cross that bridge when we come to it," or "it's impossible to do meaningful work without knowing more about what powerful AI will look like." If you think that prosaic AGI is plausible, then we may already know what the bridge will look like when we get to it: if we can't do meaningful work now, then we have a problem.

1. Prosaic AGI

It now seems possible that we could build "prosaic" AGI, which can replicate human behavior but doesn't involve qualitatively new ideas about "how intelligence works:"

- It's plausible that a large neural network can replicate "fast" human cognition, and that by coupling it to simple computational mechanisms—short and long-term memory, attention, etc.—we could obtain a human-level computational architecture.
- It's plausible that a variant of RL can train this architecture to actually implement human-level cognition. This would likely involve some combination of ingredients like model-based RL, imitation learning, or hierarchical RL. There are a whole bunch of ideas currently on the table and being explored; if you can't imagine any of these ideas working out, then I feel that's a failure of imagination (unless you see something I don't).

We will certainly learn something by developing prosaic AGI. The very fact that there were no qualitatively new ideas is itself surprising. And beyond that, we'll get a few more bits of information about which particular approach works, fill in a whole bunch of extra details about how to design and train powerful models, and actually get some experimental data.

But none of these developments seem to fundamentally change the alignment problem, and existing approaches to AI alignment are not bottlenecked on this kind of information. Actually having the AI in front of us may let us work several times more efficiently, but it's not going to move us from "we have no idea how to proceed" to "now we get it."

2. Our current state

2a. The concern

If we build prosaic superhuman AGI, it seems most likely that it will be trained by reinforcement learning (extending other frameworks to superhuman performance would require new ideas). It's easy to imagine a prosaic RL system learning to play games with superhuman levels of competence and flexibility. But we don't have any shovel-ready approach to training an RL system to autonomously pursue our values.

To illustrate how this can go wrong, imagine using RL to implement a decentralized autonomous organization (DAO) which maximizes its profit. If we had very powerful RL systems, such a DAO might be able to outcompete human organizations at a wide range of tasks—producing and selling cheaper widgets, but also influencing government policy, extorting/manipulating other actors, and so on.

The shareholders of such a DAO may be able to capture the value it creates as long as they are able to retain effective control over its computing hardware / reward signal. Similarly, as long as such DAOs are weak enough to be effectively governed by existing laws and institutions, they are likely to benefit humanity even if they reinvest all of their profits.

But as AI improves, these DAOs would become much more powerful than their human owners or law enforcement. And we have no ready way to use a prosaic AGI to actually represent the shareholder's interests, or to govern a world dominated by superhuman DAOs. In general, we have no way to use RL to actually interpret and implement human wishes, rather than to optimize some concrete and easily-calculated reward signal.

I feel pessimistic about human prospects in such a world.

2b. Behaving cautiously

We could respond by not letting powerful RL systems act autonomously, or handicapping them enough that we can maintain effective control.

This leads us to a potentially precarious situation: everyone agrees to deploy handicapped systems over which they can maintain meaningful control. But any actor can gain an economic advantage by skimping on such an agreement, and some people would prefer a world dominated by RL agents to one dominated by humans. So there are incentives for defection; if RL systems are very powerful, then these incentives may be large, and even a small number of defectors may be able to rapidly overtake the honest majority which uses handicapped AI systems.

This makes AI a "destructive technology" with similar characteristics to e.g. nuclear weapons, a situation I described in my last post. Over the long run I think we will need to reliably cope with this kind of situation, but I don't think we are there yet. I think we could *probably* handle this situation, but there would definitely be a significant risk of trouble.

The situation is especially risky if AI progress is surprisingly rapid, if the alignment problem proves to be surprisingly difficult, if the political situation is tense or dysfunctional, if other things are going wrong at the same time, if AI development is fragmented, if there is a large "hardware overhang," and so on.

I think that there are relatively few plausible ways that humanity could permanently and irreversibly disfigure its legacy. So I am extremely unhappy with "a significant risk of trouble."

2c. The current state of AI alignment

We know many *approaches* to alignment, it's just that none of these are at the stage of something you could actually implement ("shovel-ready")—instead they are at the stage of research projects with an unpredictable and potentially long timetable.

For concreteness, consider two intuitively appealing approaches to AI alignment:

- **IRL:** AI systems could infer human preferences from human behavior, and then try to satisfy those preferences.
- **Natural language:** AI systems could have an understanding of natural language, and then execute instructions described in natural language.

Neither of these approaches is shovel ready, in the sense that we have no idea how to actually write code that implements either of them—you would need to have some good ideas before you even knew what experiments to run.

We might hope that this situation will change automatically as we build more sophisticated AI systems. But I don't think that's necessarily the case. "Prosaic AGI" is at the point where we can actually write down some code and say "maybe this would do superhuman RL, if you ran it with enough computing power and you fiddled with the knobs a whole bunch." But these alignment proposals are nowhere near that point, and I don't see any "known unknowns" that would let us quickly close the gap. (By construction, prosaic AGI doesn't involve unknown unknowns.)

So if we found ourselves with prosaic AGI tomorrow, we'd be in the situation described in the last section, for as long as it took us to complete one of these research agendas (or to develop and then execute a new one). Like I said, I think this would *probably* be OK, but it opens up an unreasonably high chance of really bad outcomes.

3. Priorities

I think that prosaic AGI should probably be the largest focus of current research on alignment. In this section I'll argue for that claim.

3a. Easy to start now

Prosaic AI alignment is especially interesting because the problem is nearly as tractable today as it would be if prosaic AGI were actually available.

Existing alignment proposals have only weak dependencies on most of the details we would learn while building prosaic AGI (e.g. model architectures, optimization strategies, variance reduction tricks, auxiliary objectives…). As a result, ignorance about those details isn't a huge problem for alignment work. We may eventually reach the point where those details are critically important, but we aren't there yet.

For now, finding *any* plausible approach to alignment, that works for *any* setting of unknown details, would be a big accomplishment. With such an approach in hand we could start to ask how sensitive it is to the unknown details, but it seems premature to be pessimistic before even taking that first step.

Note that even in the extreme case where our approach to AI alignment would be completely different for different values of some unknown details, the speedup from knowing them in advance is at most 1/(probability of most likely possibility). The most plausibly critical details are large-scale architectural decisions, for which there is a much smaller space of possibilities.

3b. Importance

If we do develop prosaic AGI without learning a lot more about AI alignment, then I think it would be bad news (see section **2**). Addressing alignment earlier, or having a clear understanding of why it intractable, would make the situation a lot better.

I think the main way that an understanding of alignment could *fail* to be valuable is if it turns out that alignment is very easy. But in that case, we should also be able quickly to solve it now (or at least have some *candidate* solution), and then we can move on to other things. So I don't think "alignment is very easy" is a possibility that should keep us up at night.

Alignment for prosaic AGI in particular will be less important if we don't actually develop prosaic AGI, but I think that this is a very big problem:

First, I think there is a reasonable chance (>10%) that we will build prosaic AGI. At this point there don't seem to be convincing arguments against the possibility, and one of the lessons of the last 30 years is that learning algorithms and lots of computation/data can do surprisingly well compared to approaches that require understanding "how to think."

Indeed, I think that if you had forced someone in 1990 to write down a concrete way that an AGI might work, they could easily have put 10–20% of their mass on the same cluster of possibilities that I'm currently calling "prosaic AGI." And if you'd ask them to guess what prosaic AGI would look like, I think that they could have given more like 20–40%.

Second, even if we don't develop prosaic AGI, I think it is very likely that there will be important similarities between alignment for prosaic AGI and alignment for whatever kind of AGI we actually build. For example, whatever AGI we actually build is likely to exploit many of the same techniques that a prosaic AGI would, and to the extent that those techniques pose challenges for alignment we will probably have to deal with them one way or another.

I think that working with a concrete model that we have available now is one of the best ways to make progress on alignment, even in cases where we *are* sure that there will be at least one qualitative change in how we think about AI.

Third, I think that research on alignment is significantly more important in cases where powerful AI is developed relatively soon. And in these cases, the probability of prosaic AGI seems to be *much* higher. If prosaic AGI is possible, then I think there is a significant chance of building broadly human level AGI over the next 10–20 years. I'd

guess that hours of work on alignment are perhaps 10x more important if AI is developed in the next 15 years than if it is developed later, just based on simple heuristics based on diminishing marginal returns.

3c. Feasibility

Some researchers (especially at MIRI) believe that aligning prosaic AGI is probably infeasible—that the most likely approach to building an aligned AI is to understand intelligence in a much deeper way than we currently do, and that if we manage to build AGI before achieving such an understanding then we are in deep trouble.

I think that this shouldn't make us much less enthusiastic about prosaic AI alignment:

First, I don't think it's reasonable to have a confident position on this question. Claims of the form "problem X can't be solved" are *really hard to get right*, because you are fighting against the universal quantifier of all possible ways that someone could solve this problem. (This is very similar to the difficulty of saying "system X can't be compromised.") To the extent that there is any argument that aligning prosaic AGI is infeasible, that argument is nowhere near the level of rigor which would be compelling.

This implies on the one hand that it would be unwise to assign a high probability to the infeasibility of this problem. It implies on the other hand that even if the problem is infeasible, then we might expect to develop a substantially more complete understanding of why exactly it is so difficult.

Second, if this problem is actually infeasible, that is an extremely important fact with direct consequences for what we ought to do. It implies we will be unable to quickly play "catch up" on alignment after developing prosaic AGI, and so we would need to rely on coordination to prevent catastrophe. As a result:

- We should start preparing for such coordination immediately.
- It would be worthwhile for the AI community to substantially change its research direction in order to avoid catastrophe, even though this would involve large social costs.

I think we don't yet have very strong evidence for the intractability of this problem.

If we could *get* very strong evidence, I expect it would have a significant effect on changing researchers' priorities and on the research community's attitude towards AI development. Realistically, it's probably also a precondition for getting AI researchers to make a serious move towards an alternative approach to AI development, or to start talking seriously about the kind of coordination that would be needed to cope with hard-to-align AI.

Conclusion

I've claimed that prosaic AGI is conceivable, that it is a very appealing target for research on AI alignment, and that this gives us more reason to be enthusiastic for the overall tractability of alignment. For now, these arguments motivate me to focus on prosaic AGI.

This post was originally published here on 19th Nov 2016.

The next post in this sequence will be "Approval-directed agents: overview" by Paul Christiano, and will release on Thursday 22nd November.

Tomorrow's AI Alignment Forum sequences post will be "Iterated Fixed Point Exercises" by Scott Garrabrant and Sam Eisenstat, in the sequence "Fixed Points".

Approval-directed agents

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

*Note: This is the first post from **part two: basic intuitions** of the sequence on iterated amplification. The second part of the sequence outlines the basic intuitions that motivate iterated amplification. I think that these intuitions may be more important than the scheme itself, but they are considerably more informal.*

Research in AI is steadily progressing towards more flexible, powerful, and autonomous goal-directed behavior. This progress is likely to have significant economic and humanitarian benefits: it helps make automation faster, cheaper, and more effective, and it allows us to automate *deciding what to do*.

Many researchers expect goal-directed machines to predominate, and so have considered the long-term implications of this kind of automation. Some of these implications are worrying: if sophisticated artificial agents pursue their own objectives and are as smart as we are, then the future may be shaped as much by their goals as by ours.

Most thinking about "AI safety" has focused on the possibility of goal-directed machines, and asked how we might ensure that their goals are agreeable to humans. But there are other possibilities.

In this post I will flesh out one alternative to goal-directed behavior. I think this idea is particularly important from the perspective of AI safety.

Approval-directed agents

Consider a human Hugh, and an agent Arthur who uses the following procedure to choose each action:

Estimate the expected rating Hugh would give each action if he considered it at length. Take the action with the highest expected rating.

I'll call this "approval-directed" behavior throughout this post, in contrast with goal-directed behavior. In this context I'll call Hugh an "overseer."

Arthur's actions are rated more highly than those produced by any alternative procedure. That's comforting, but it doesn't mean that Arthur is optimal. An optimal agent may make decisions that have *consequences* Hugh would approve of, even if Hugh can't anticipate those consequences himself. For example, if Arthur is playing chess he should make moves that are actually good—not moves that Hugh thinks are good.

The quality of approval-directed decisions is limited by the *minimum* of Arthur's ability and Hugh's ability: Arthur makes a decision only if it looks good to both Arthur and Hugh. So why would Hugh be interested in this proposal, rather than doing things himself?

- Hugh doesn't actually rate actions, he just participates in a hypothetical rating process. So Hugh can oversee many agents like Arthur at once (and spend his actual time relaxing on the beach). In many cases, this is the whole point of automation.
- Hugh can (hypothetically) think for a very long time about each decision—longer than would be practical or cost-effective if he had to actually make the decision himself.
- Similarly, Hugh can think about Arthur's decisions at a very low level of detail. For example, Hugh might rate a chess-playing AI's choices about how to explore the game tree, rather than rating its final choice of moves. If Arthur is making billions of small decisions each second, then Hugh can think in depth about each of them, and the resulting system can be much smarter than Hugh.
- Hugh can (hypothetically) use additional resources in order to make his rating: powerful computers, the benefit of hindsight, many assistants, very long time periods.
- Hugh's capabilities can be gradually escalated as needed, and one approval-directed system can be used to bootstrap to a more effective successor. For example, Arthur could advise Hugh on how to define a better overseer; Arthur could offer advice in real-time to help Hugh be a better overseer; or Arthur could directly act as an overseer for his more powerful successor.

In most situations, I would expect approval-directed behavior to capture the benefits of goal-directed behavior, while being easier to define and more robust to errors.

Advantages

Facilitate indirect normativity

Approval-direction is closely related to what Nick Bostrom calls "indirect normativity"—describing what is good indirectly, by describing how to tell what is good. I think this idea encompasses the most credible proposals for defining a powerful agent's goals, but has some practical difficulties.

Asking an overseer to evaluate *outcomes* directly requires defining an **extremely** intelligent overseer, one who is equipped (at least in principle) to evaluate the entire future of the universe. This is probably impractical overkill for the kinds of agents we will be building in the near future, who *don't* have to think about the entire future of the universe.

Approval-directed behavior provides a more realistic alternative: start with simple approval-directed agents and simple overseers, and scale up the overseer and the agent in parallel. I expect the approval-directed dynamic to converge to the desired limit; this requires only that the simple overseers approve of scaling up to more powerful overseers, and that they are able to recognize appropriate improvements.

Avoid lock-in

Some approaches to AI require "locking in" design decisions. For example, if we build a goal-directed AI with the wrong goals then the AI might never correct the mistake on its own. For sufficiently sophisticated AI's, such mistakes may be very expensive to fix.

There are also more subtle forms of lock-in: an AI may also not be able to fix a bad choice of decision-theory, sufficiently bad priors, or a bad attitude towards infinity. It's hard to know what other properties we might inadvertently lock-in.

Approval-direction involves only extremely minimal commitments. If an approval-directed AI encounters an unforeseen situation, it will respond in the way that we most approve of. We don't need to make a decision until the situation actually arises.

Perhaps most importantly, an approval-directed agent can correct flaws in its own design, and will search for flaws if we want it to. It can change its own decision-making procedure, its own reasoning process, and its own overseer.

Fail gracefully

Approval-direction seems to "fail gracefully:" if we slightly mess up the specification, the approval-directed agent probably won't be actively malicious. For example, suppose that Hugh was feeling extremely apathetic and so evaluated proposed actions only superficially. The resulting agent would not aggressively pursue a flawed realization of Hugh's values; it would just behave lackadaisically. The mistake would be quickly noticed, unless Hugh deliberately approved of actions that concealed the mistake.

This looks like an improvement over misspecifying goals, which leads to systems that are actively opposed to their users. Such systems are motivated to conceal possible problems and to behave maliciously.

The same principle sometimes applies if you define the right overseer but the agent reasons incorrectly about it, if you misspecify the entire rating process, or if your system doesn't work quite like you expect. Any of these mistakes could be serious for a goal-directed agent, but are probably handled gracefully by an approval-directed agent.

Similarly, if Arthur is smarter than Hugh expects, the only problem is that Arthur won't be able to use all of his intelligence to devise excellent plans. This is a serious problem, but it can be fixed by trial and error—rather than leading to surprising failure modes.

Is it plausible?

I've already mentioned the practical demand for goal-directed behavior and why I think that approval-directed behavior satisfies that demand. There are other reasons to think that agents might be goal-directed. These are all variations on the same theme, so I apologize if my responses become repetitive.

Internal decision-making

We assumed that Arthur can predict what actions Hugh will rate highly. But in order to make these predictions, Arthur might use goal-directed behavior. For example, Arthur might perform a calculation because he believes it will help him predict what actions Hugh will rate highly. Our apparently approval-directed decision-maker may have goals after all, on the inside. Can we avoid this?

I think so: Arthur's internal decisions could also be approval-directed. Rather than performing a calculation because it will help make a good prediction, Arthur can perform that calculation because Hugh would rate this decision highly. If Hugh is coherent, then taking individual steps that Hugh rates highly leads to overall behavior that Hugh would approve of, just like taking individual steps that maximize X leads to behavior that maximizes X.

In fact the result may be more desirable, from Hugh's perspective, than maximizing Hugh's approval. For example, Hugh might incorrectly rate some actions highly, because he doesn't understand them. An agent maximizing Hugh's approval might find those actions and take them. But if the agent was internally approval-directed, then it wouldn't try to exploit errors in Hugh's ratings. Actions that lead to reported approval but not real approval, don't lead to approval for approved reasons

Turtles all the way down?

Approval-direction stops making sense for low-level decisions. A program moves data from register A into register B because that's what the next instruction says, not because that's what Hugh would approve of. After all, deciding whether Hugh would approve itself requires moving data from one register to another, and we would be left with an infinite regress.

The same thing is true for goal-directed behavior. Low-level actions are taken because the programmer chose them. The programmer may have chosen them because she thought they would help the system achieve its goal, but the actions themselves are performed because that's what's in the code, not because of an explicit belief that they will lead to the goal. Similarly, actions might be performed because a simple heuristic suggests they will contribute to the goal—the heuristic was chosen or learned because it was expected to be useful for the goal, but the action is motivated by the heuristic. Taking the action doesn't involve thinking about the heuristic, just following it.

Similarly, an approval-directed agent might perform an action because it's the next instruction in the program, or because it's recommended by a simple heuristic. The program or heuristic might have been chosen to result in approved actions, but the taking the action doesn't involve reasoning about approval. The aggregate effect of using and refining such heuristics is to effectively do what the user approves of.

In many cases, perhaps a majority, the heuristics for goal-directed and approval-directed behavior will coincide. To answer "what do I want this function to do next?" I very often ask "what do I want the end result to be?" In these cases the difference is in how we think about the behavior of the overall system, and what invariants we try to maintain as we design it.

Relative difficulty?

Approval-directed subsystems might be harder to build than goal-directed subsystems. For example, there is much more data of the form "X leads to Y" than of the form "the user approves of X." This is a typical AI problem, though, and can be approached using typical techniques.

Approval-directed subsystems might also be easier to build, and I think this is the case today. For example, I recently wrote a function to decide which of two methods to use for the next step of an optimization. Right now it uses a simple heuristic with mediocre performance. But I could also have labeled some examples as "use method A" or "use

method B," and trained a model to predict what I would say. This model could then be used to decide when to use A, when to use B, and when to ask me for more training data.

Reflective stability

Rational goal-directed behavior is reflectively stable: if you want X, you generally want to continue wanting X. Can approval-directed behavior have the same property?

Approval-directed systems inherit reflective stability (or instability) from their overseers. Hugh can determine whether Arthur "wants" to remain approval-directed, by approving or disapproving of actions that would change Arthur's decision-making process.

Goal-directed agents want to be wiser and know more, though their goals are stable. Approval-directed agents also want to be wiser and know more, but they also want their overseers to be wiser and know more. The overseer is not stable, but the overseer's values are. This is a feature, not a bug.

Similarly, an agent composed of approval-directed subsystems overseen by Hugh is *not* the same as an approval-directed agent overseen by Hugh. For example, the composite may make decisions too subtle for Hugh to understand. Again, this is a feature, not a bug.

Black box search

(Note: I no longer agree with the conclusions of this section. I now feel that approval-directed agents can probably be constructed out of powerful black-box search (or stochastic gradient descent); my main priority is now either handling this setting or else understanding exactly what the obstruction is. Ongoing work in this direction is collected at ai-control, and will hopefully be published in a clear format by the end of 2016.)

Some approaches to AI probably can't yield approval-directed agents. For example, we could perform a search which treats possible agents as a black boxes and measures their behavior for signs of intelligence. Such a search could (eventually) find a human-level intelligence, but would give us very crude control over how that intelligence was applied. We could get some kind of goal-directed behavior by selecting for it, but selecting for approval-directed behavior would be difficult:

1. The paucity of data on approval is a huge problem in this setting. (Note: semi-supervised reinforcement learning is an approach to this problem.)
2. You have no control over the internal behavior of the agent, which you would expect to be optimized for pursuing a particular goal: maximizing whatever measure of "approval" that you used to guide your search. (Note: I no longer endorse this argument as written; reward engineering is a response to the substance of this concern.)
3. Agents who maximized your reported approval in test cases need not do so in general, any more than humans are reliable reproductive-fitness-maximizers. (Note: red teaming is an approach to this problem.)

But [1] and especially [3] are also problems when designing a goal-directed agent with **agreeable** goals, or indeed any particular goals at all. Though approval-direction can't deal with these problems, they aren't new problems.

Such a black-box search—with little insight into the internal structure of the agents—seems worrying no matter how we approach AI safety. Fortunately, it also seems unlikely (though not out of the question).

A similar search is more likely to be used to produce internal components of a larger system (for example, you might train a neural network to identify objects, as a component of a system for navigating an unknown environment). This presents similar challenges, concerning robustness and unintended behaviors, whether we are designing a goal-directed or approval-directed agent.

"Implementation" details

So far I've talked about approval-direction imprecisely. Maybe I'm talking about something incoherent, which has desirable properties only in the same sense as a four-sided triangle—vacuously. I won't really be able to dispel this concern here, but I'll at least take some steps.

How do you define approval?

Eventually you would have to actually write code implementing approval-directed behavior. What might that code look like? I want to set aside the problem "what does a sophisticated AI look like?" since I obviously don't know. So let's suppose we had some black box that did all of the hard work. I'll consider a few cases for what the black box does, ranging from "easy to work with" to "very hard to work with."

(Note: I now believe that we can target AI systems trained (nearly) end-to-end with gradient descent, which is most similar to "learning from examples.")

Natural language

As an easy case, suppose we have a natural language question-answering system, which can assign a probability to any natural language sentence. In this case, we ask the question:

> "Suppose that Hugh understood the current situation, was asked `on a scale from 0 to 1, how good is the action a?' and was given a few hours to determine his answer. What would his answer be?"

We then loop over each action *a* and take the action with the highest expected answer.

In this framework, it is easy to replace Hugh by a more powerful overseer—all you have to do is specify the replacement in natural language.

"Math intuition module"

At an opposite extreme, suppose we have a "math intuition module," a system which can assign probabilities only to *perfectly* precise statements—perhaps of the form

“algorithm A returns output *y* on input *x*.”

I’ve written about defining “approval upon reflection” algorithmically (see here, here). These definition can be used to define approval-directed behavior completely precisely. I’m pretty hesitant about these definitions, but I do think it is promising that we can get traction even in such an extreme case.

In reality, I expect the situation to be somewhere in between the simple case of natural language and the hard case of mathematical rigor. Natural language is the case where we share all of our concepts with our machines, while mathematics is the case where we share only the most primitive concepts. In reality, I expect we will share some but not all of our concepts, with varying degrees of robustness. To the extent that approval-directed decisions are robust to imprecision, we can safely use some more complicated concepts, rather than trying to define what we care about in terms of logical primitives.

Learning from examples

In an even harder case, suppose we have a function learner which can take some labelled examples $f(x) = y$ and then predict a new value $f(x')$. In this case we have to define “Hugh’s approval” directly via examples. I feel less comfortable with this case, but I’ll take a shot anyway.

In this case, our approval-directed agent Arthur maintains a probabilistic model over sequences **observation**[*T*] and **approval**[*T*](*a*). At each step *T*, Arthur selects the action *a* maximizing **approval**[*T*](*a*). Then the timer *T* is incremented, and Arthur records **observation**[*T*+1] from his sensors. Optionally, Hugh might specify a value **approval**[*t](a')* for any time *t* and any action *a’*. Then Arthur updates his models, and the process continues.

Like AIXI, if Arthur is clever enough he eventually learns that **approval**[*T*](*a)*refers to whatever Hugh will retroactively input. But unlike AIXI, Arthur will make no effort to manipulate these judgments. Instead he takes the action maximizing his expectation of **approval**[*T*]—i.e., his prediction about what Hugh will say in the future, if Hugh says anything at all. (This depends on his self-predictions, since what Hugh does in the future depends on what Arthur does now.)

At any rate, this is quite a lot better than AIXI, and it might turn out fine if you exercise appropriate caution. I wouldn’t want to use it in a high-stakes situation, but I think that it is a promising idea and that there are many natural directions for improvement. For example, we could provide further facts about **approval** (beyond example values), interpolating continuously between learning from examples and using an explicit definition of the approval function. More ambitiously, we could implement “approval-directed learning,” preventing it from learning complicated undesired concepts.

How should Hugh rate?

So far I’ve been very vague about what Hugh should actually do when rating an action. But the approval-directed behavior depends on how Hugh decides to administer approval. How should Hugh decide?

If Hugh expects action *a* to yield better consequences than action *b*, then he should give action *a* a higher rating than action *b.* In simple environments he can simply pick

the best action, give it a rating of 1, and give the other options a rating of 0.

If Arthur is so much smarter than Hugh that he knows exactly what Hugh will say, then we might as well stop here. In this case, approval-direction amounts to Arthur doing exactly what Hugh instructs: “the minimum of Arthur’s capabilities and Hugh’s capabilities” is equal to “Hugh’s capabilities.”

But most of the time, Arthur won’t be able to tell exactly what Hugh will say. The numerical scale between 0 and 1 exists to accomodate Arthur’s uncertainty.

To illustrate the possible problems, suppose that Arthur is considering whether to drive across a bridge that may or may not collapse. Arthur thinks the bridge will collapse with 1% probability. But Arthur also think that Hugh knows for sure whether or not the bridge will collapse. If Hugh always assigned the optimal action a rating of 1 and every other action a rating of 0, then Arthur would take the action that was most likely to be optimal—driving across the bridge.

Hugh should have done one of two things:

- Give a bad rating for risky behavior. Hugh should give Arthur a high rating only if he drives across the bridge and *knows that it is safe*. In general, give a rating of 1 to the best action *ex ante*.
- Assign a very bad rating to incorrectly driving across the bridge, and only a small penalty for being too cautious. In general, give ratings that reflect the utilities of possible outcomes—to the extent you know them.

Probably Hugh should do both. This is easier if Hugh understands what Arthur is thinking and why, and what range of possibilities Arthur is considering.

Other details

I am leaving out many other important details in the interest of brevity. For example:

- In order to make these evaluations Hugh might want to understand what Arthur is thinking and why. This might be accomplished by giving Hugh enough time and resources to understand Arthur’s thoughts; or by letting different instances of Hugh “communicate” to keep track of what is going on as Arthur’s thoughts evolve; or by ensuring that Arthur’s thoughts remains comprehensible to Hugh (perhaps by using approval-directed behavior at a lower level, and only approving of internal changes that can be rendered comprehensible).
- It is best if Hugh optimizes his ratings to ensure the system remains robust. For example, in high stakes settings, Hugh should sometimes make Arthur consult the real Hugh to decide how to proceed—even if Arthur correctly knows what Hugh wants. This ensures that Arthur will seek guidance when he *incorrectly* believes that he knows what Hugh wants.

…and so on. The details I *have* included should be considered illustrative at best. (I don’t want anyone to come away with a false sense of precision.)

Problems

It would be sloppy to end the post without a sampling of possible pitfalls. For the most part these problems have more severe analogs for goal-directed agents, but it's still wise to keep them in mind when thinking about approval-directed agents in the context of AI safety.

My biggest concerns

I have three big concerns with approval-directed agents, which are my priorities for follow-up research:

- Is an approval-directed agent generally as useful as a goal-directed agent, or does this require the overseer to be (extremely) powerful? Based on the ideas in this post, I am cautiously optimistic.
- Can we actually define approval-directed agents by examples, or do they already need a shared vocabulary with their programmers? I am again cautiously optimistic.
- Is it realistic to build an intelligent approval-directed agent without introducing goal-directed behavior internally? I think this is probably the most important follow-up question. I would guess that the answer will be "it depends on how AI plays out," but we can at least get insight by addressing the question in a variety of concrete scenarios.

Motivational changes for the overseer

"What would I say if I thought for a *very* long time?" might have a surprising answer. The very process of thinking harder, or of finding myself in a thought experiment, might alter my priorities. I may care less about the real world, or may become convinced that I am living in a simulation.

This is a particularly severe problem for my proposed implementation of indirect normativity, which involves a truly outlandish process of reflection. It's still a possible problem for defining approval-direction, but I think it is much less severe.

"What I would say after a few hours," is close enough to real life that I wouldn't expect my thought process to diverge too far from reality, either in values or beliefs. Short time periods are much easier to predict, and give less time to explore completely unanticipated lines of thought. In practice, I suspect we can also define something like "what I would say after a few hours of sitting at my desk under completely normal conditions," which looks particularly innocuous.

Over time we will build more powerful AI's with more powerful (and perhaps more exotic) overseers, but making these changes gradually is much easier than making them all at once: small changes are more predictable, and each successive change can be made with the help of increasingly powerful assistants.

Treacherous turn

If Hugh inadvertently specifies the wrong overseer, then the resulting agent might be motivated to deceive him. *Any* rational overseer will be motivated to approve of actions that look reasonable to Hugh. If they don't, Hugh will notice the problem and fix the bug, and the original overseer will lose their influence over the world.

This doesn't seem like a big deal—a failed attempt to specify "Hugh" probably won't inadvertently specify a different Hugh-level intelligence, it will probably fail innocuously.

There are some possible exceptions, which mostly seem quite obscure but may be worth having in mind. The learning-from-examples protocol seems particularly likely to have problems. For example:

- Someone other than Hugh might be able to enter training data for **approval**[*T*](*a).* Depending on how Arthur is defined, these examples might influence Arthur's behavior as soon as Arthur expects them to appear. In the most pathological case, these changes in Arthur's behavior might have been the very reason that someone had the opportunity to enter fraudulent training data.
- Arthur could accept the motivated simulation argument, believing himself to be in a simulation at the whim of a simulator attempting to manipulate his behavior.
- The simplest explanation for Hugh's judgments may be a simple program motivated to "mimic" the series **approval**[*T*] and **observation**[*T*] in order to influence Arthur.

Ignorance

An approval-directed agent may not be able to figure out what I approve of.

I'm skeptical that this is a serious problem. It falls under the range of predictive problems I'd expect a sophisticated AI to be good at. So it's a standard objective for AI research, and AI's that can't make such predictions probably have significantly sub-human ability to act in the world. Moreover, even a fairly weak reasoner can learn generalizations like "actions that lead to Hugh getting candy, tend to be approved of" or "actions that take control away from Hugh, tend to be disapproved of."

If there is a problem, it doesn't seem like a serious one. Straightforward misunderstandings will lead to an agent that is inert rather than actively malicious (see the "Fail gracefully" section). And deep misunderstandings can be avoided, by Hugh approving of the decision "consult Hugh."

Conclusion

Making decisions by asking "what **action** would your owner most approve of?" may be more robust than asking "what **outcome** would your owner most approve of?" Choosing actions directly has limitations, but these might be overcome by a careful implementation.

More generally, the focus on achieving safe goal-directed behavior may have partially obscured the larger purpose of the AI safety community, which should be achieving safe and *useful* behavior. It may turn out that goal-directed behavior really is inevitable or irreplaceable, but the case has not yet been settled.

This essay was originally posted here on 1st December 2014.

Tomorrow's AI Alignment Forum sequences post will be 'Fixed Point Discussion' by Scott Garrabrant, in the sequence 'Fixed Points'.

The next posts in this sequence will be 'Approval directed bootstrapping' and 'Humans consulting HCH', two short posts which will come out on Sunday 25th November.

Approval-directed bootstrapping

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

Approval-directed behavior works best when the overseer is very smart. Where can we find a smart overseer?

One approach is *bootstrapping*. By thinking for a long time, a weak agent can oversee an agent (slightly) smarter than itself. Now we have a slightly smarter agent, who can oversee an agent which is (slightly) smarter still. This process can go on, until the intelligence of the resulting agent is limited by technology rather than by the capability of the overseer. At this point we have reached the limits of our technology.

This may sound exotic, but we can implement it in a surprisingly straightforward way.

Suppose that we evaluate Hugh's approval by predicting what Hugh would say if we asked him; the rating of action *a* is what Hugh would say if, instead of taking action *a,* we asked Hugh, "How do you rate action *a*?"

Now we get bootstrapping almost for free. In the process of evaluating a proposed action, Hugh can consult Arthur. This new instance of Arthur will, in turn, be overseen by Hugh—and in this new role Hugh can, in turn, be assisted by Arthur. In principle we have defined the entire infinite regress before Arthur takes his first action.

We can even learn this function by examples—no elaborate definitions necessary. Each time Arthur proposes an action, we actually ask Hugh to evaluate the action with some probability, and we use our observations to train a model for Hugh's judgments.

In practice, Arthur might not be such a useful assistant until he has acquired some training data. As Arthur acquires training data, the Hugh+Arthur system becomes more intelligent, and so Arthur acquires training data from a more intelligent overseer. The bootstrapping unfolds over time as Arthur adjusts to increasingly powerful overseers.

This was originally posted here on 21st December 2014.

Tomorrow's AI Alignment Forum sequences will take a break, and tomorrow's post will be Issue #34 of the Alignment Newsletter.

The next post in this sequence is 'Humans consulting HCH', also released today.

Humans Consulting HCH

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

(See also: strong HCH.)

Consider a human Hugh who has access to a question-answering machine. Suppose the machine answers question Q by perfectly imitating how Hugh would answer question Q, *if Hugh had access to the question-answering machine*.

That is, Hugh is able to consult a copy of Hugh, who is able to consult a copy of Hugh, who is able to consult a copy of Hugh…

Let's call this process HCH, for "Humans Consulting HCH."

I've talked about many variants of this process before, but I find it easier to think about with a nice handle. (Credit to Eliezer for proposing using a recursive acronym.)

HCH is easy to specify very precisely. For now, I think that HCH is our best way to precisely specify "a human's enlightened judgment." It's got plenty of problems, but for now I don't know anything better.

Elaborations

We can define realizable variants of this inaccessible ideal:

- For a particular prediction algorithm P, define HCH^P as:
  "P's prediction of what a human would say after consulting HCH^P"
- For a reinforcement learning algorithm A, define max-HCH^A as:
  "A's output when maximizing the evaluation of a human after consulting max-HCH^A"
- For a given market structure and participants, define HCH^{market} as:
  "the market's prediction of what a human will say after consulting HCH^{market}"

Note that e.g. HCH^P is totally different from "P's prediction of HCH." HCH^P will generally make worse predictions, but it is easier to implement.

Hope

The best case is that HCH^P, max-HCH^A, and HCH^{market} are:

- As capable as the underlying predictor, reinforcement learner, or market participants.
- Aligned with the enlightened judgment of the human, e.g. as evaluated by HCH.

(At least when the human is suitably prudent and wise.)

It is clear from the definitions that these systems can't be any *more* capable than the underlying predictor/learner/market. I honestly don't know whether we should expect

them to match the underlying capabilities. My intuition is that max-HCH^A probably can, but that HCH^P and HCH^{market} probably can't.

It is similarly unclear whether the system continues to reflect the human's judgment. In some sense this is in tension with the desire to be capable—the more guarded the human, the less capable the system but the more likely it is to reflect their interests. The question is whether a prudent human can achieve both goals.

This was originally posted here on 29th January 2016.

Tomorrow's AI Alignment Forum sequences will take a break, and tomorrow's post will be Issue #34 of the Alignment Newsletter.

The next post in this sequence is 'Corrigibility' by Paul Christiano, which will be published on Tuesday 27th November.

Corrigibility

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

(*Warning: rambling.*)

I would like to build AI systems which help me:

- Figure out whether I built the right AI and correct any mistakes I made
- Remain informed about the AI's behavior and avoid unpleasant surprises
- Make better decisions and clarify my preferences
- Acquire resources and remain in effective control of them
- Ensure that my AI systems continue to do all of these nice things
- …and so on

We say an agent is *corrigible* (article on Arbital) if it has these properties. I believe this concept was introduced in the context of AI by Eliezer and named by Robert Miles; it has often been discussed in the context of narrow behaviors like respecting an off-switch, but here I am using it in the broadest possible sense.

In this post I claim:

1. A benign act-based agent will be robustly corrigible if we want it to be.
2. A sufficiently corrigible agent will tend to become more corrigible and benign over time. Corrigibility marks out a broad basin of attraction towards acceptable outcomes.

As a consequence, we shouldn't think about alignment as a narrow target which we need to implement exactly and preserve precisely. We're aiming for a broad basin, and trying to avoid problems that could kick out of that basin.

This view is an important part of my overall optimism about alignment, and an important background assumption in some of my writing.

1. Benign act-based agents can be corrigible

A benign agent optimizes in accordance with our preferences. An act-basedagent considers our short-term preferences, including (amongst others) our preference for the agent to be corrigible.

If *on average* we are unhappy with the level of corrigibility of a benign act-based agent, then by construction it is mistaken about our short-term preferences.

This kind of corrigibility doesn't require any special machinery. An act-based agent turns off when the overseer presses the "off" button not because it has received new evidence, or because of delicately balanced incentives. It turns off because that's what the overseer prefers.

Contrast with the usual futurist perspective

Omohundro's The Basic AI Drives argues that "almost all systems [will] protect their utility functions from modification," and Soares, Fallenstein, Yudkowsky, and Armstrong cite as: "almost all [rational] agents are instrumentally motivated to preserve their preferences." This motivates them to consider modifications to an agent to remove this default incentive.

Act-based agents are generally an exception to these arguments, since the overseer has preferences about whether the agent protects its utility function from modification. Omohundro presents preferences-about-your-utility function case as a somewhat pathological exception, but I suspect that it will be the typical state of affairs for powerful AI (as for humans) and it does not appear to be unstable. It's also very easy to implement in 2017.

Is act-based corrigibility robust?

How is corrigibility affected if an agent is ignorant or mistaken about the overseer's preferences?

I think you don't need particularly accurate models of a human's preferences before you can predict that they want their robot to turn off when they press the off button or that they don't want to be lied to.

In the concrete case of an approval-directed agent, "human preferences" are represented by human responses to questions of the form "how happy would you be if I did *a*?" If the agent is considering the action *a* precisely because it is manipulative or would thwart the user's attempts to correct the system, then it doesn't seem hard to predict that the overseer will object to *a.*

Eliezer has suggested that this is a very anthropocentric judgment of "easiness." I don't think that's true—I think that given a description of a proposed course of action, the judgment "is agent X being misled?" is objectively a relatively easy prediction problem (compared to the complexity of generating a strategically deceptive course of action).

Fortunately this is the kind of thing that we will get a great deal of evidence about long in advance. Failing to predict the overseer becomes *less* likely as your agent becomes smarter, not more likely. So if in the near future we build systems that make good enough predictions to be corrigible, then we can expect their superintelligent successors to have the same ability.

(This discussion mostly applies on the training distribution and sets aside issues of robustness/reliability of the predictor itself, for which I think adversarial training is the most plausible solution. This issue will apply to any approach to corrigibility which involves machine learning, which I think includes any realistic approach.)

Is instrumental corrigibility robust?

If an agent shares the overseer's long-term values and is corrigible instrumentally, a slight divergence in values would turn the agent and the overseer into adversaries

and totally break corrigibility. This can also happen with a framework like CIRL—if the way the agent infers the overseer's values is slightly different from what the overseer would conclude upon reflection (which seems quite likely when the agent's model is misspecified, as it inevitably will be!) then we have a similar adversarial relationship.

2. Corrigible agents become more corrigible/aligned

In general, an agent will prefer to build other agents that share its preferences. So if an agent inherits a distorted version of the overseer's preferences, we might expect that distortion to persist (or to drift further if subsequent agents also fail to pass on their values correctly).

But a corrigible agent prefers to build other agents that share *the overseer's* preferences—even if the agent doesn't yet share the overseer's preferences perfectly. After all, even if you only approximately know the overseer's preferences, you know that the overseer would prefer the approximation get better rather than worse.

Thus an entire neighborhood of possible preferences lead the agent towards the same basin of attraction. We just have to get "close enough" that we are corrigible, we don't need to build an agent which exactly shares humanity's values, philosophical views, or so on.

In addition to making the initial target bigger, this gives us some reason to be optimistic about the dynamics of AI systems iteratively designing new AI systems. Corrigible systems want to design more corrigible and more capable successors. Rather than our systems traversing a balance beam off of which they could fall at any moment, we can view them as walking along the bottom of a ravine. As long as they don't jump to a completely different part of the landscape, they will continue traversing the correct path.

This is all a bit of a simplification (though I think it gives the right idea). In reality the space of possible errors and perturbations carves out a low degree manifold in the space of all possible minds. Undoubtedly there are "small" perturbations in the space of possible minds which would lead to the agent falling off the balance beam. The task is to parametrize our agents such that the manifold of likely-successors is restricted to the part of the space that looks more like a ravine. In the last section I argued that act-based agents accomplish this, and I'm sure there are alternative approaches.

Amplification

Corrigibility also protects us from gradual value drift during capability amplification. As we build more powerful compound agents, their values may effectively drift. But unless the drift is large enough to disrupt corrigibility, the compound agent will continue to attempt to correct and manage that drift.

This is an important part of my optimism about amplification. It's what makes it coherent to talk about preserving benignity as an inductive invariant, even when "benign" appears to be such a slippery concept. It's why it makes sense to talk about reliability and security as if being "benign" was a boolean property.

In all these cases I think that I should actually have been arguing for corrigibility rather than benignity. The robustness of corrigibility means that we can potentially get by with a *good enough* formalization, rather than needing to get it exactly right. The fact that corrigibility is a basin of attraction allows us to consider failures as discrete events rather than worrying about slight perturbations. And the fact that corrigibility eventually leads to aligned behavior means that *if* we could inductively establish corrigibility, then we'd be happy.

This is still not quite right and not at all formal, but hopefully it's getting closer to my real reasons for optimism.

Conclusion

I think that many futurists are way too pessimistic about alignment. Part of that pessimism seems to stem from a view like "any false move leads to disaster." While there are some kinds of mistakes that clearly do lead to disaster, I also think it is possible to build the kind of AI where *probable*perturbations or errors will be gracefully corrected. In this post I tried to informally flesh out my view. I don't expect this to be completely convincing, but I hope that it can help my more pessimistic readers understand where I am coming from.

Postscript: the hard problem of corrigibility and the diff of my and Eliezer's views

I share many of Eliezer's intuitions regarding the "hard problem of corrigibility" (I assume that Eliezer wrote this article). Eliezer's intuition that there is a "simple core" to corrigibility corresponds to my intuition that corrigible behavior is *easy to learn* in some non-anthropomorphic sense.

I *don't* expect that we will be able to specify corrigibility in a simple but algorithmically useful way, nor that we need to do so. Instead, I am optimistic that we can build agents which learn to reason by human supervision over reasoning steps, which pick up corrigibility along with the other useful characteristics of reasoning.

Eliezer argues that we shouldn't rely on a solution to corrigibility unless it is simple enough that we can formalize and sanity-check it ourselves, even if it appears that it can be learned from a small number of training examples, because an "AI that seemed corrigible in its infrahuman phase [might] suddenly [develop] extreme or unforeseen behaviors when the same allegedly simple central principle was reconsidered at a higher level of intelligence."

I don't buy this argument because I disagree with implicit assumptions about how such principles will be embedded in the reasoning of our agent. For example, I don't think that this principle would affect the agent's reasoning by being explicitly considered. Instead it would influence the way that the reasoning itself worked. It's possible that after translating between our differing assumptions, my enthusiasm about embedding corrigibility deeply in reasoning corresponds to Eliezer's enthusiasm about "lots of particular corrigibility principles."

I feel that my current approach is a reasonable angle of attack on the hard problem of corrigibility, and that we can currently write code which is reasonably likely to solve

the problem (though not knowably). I do not feel like we yet have credible alternatives.

I *do* grant that if we need to learn corrigible reasoning, then it is vulnerable to failures of robustness/reliability, and so learned corrigibility is not itself an adequate protection against failures of robustness/reliability. I could imagine other forms of corrigibility that do offer such protection, but it does not seem like the most promising approach to robustness/reliability.

I *do* think that it's reasonably likely (maybe 50–50) that there is some clean concept of "corrigibility" which (a) we can articulate in advance, and (b) plays an important role in our *analysis* of AI systems, if not in their construction.

This was originally posted here on 10th June 2017.

The next post in the sequence on 'Iterated Amplification' will be 'Iterated Distillation and Amplification' by Ajeya Cotra.

Tomorrow's AI Alignment Forum sequences posts will be 4 posts of agent foundations research, in the sequence 'Fixed Points'.

Iterated Distillation and Amplification

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

This is a guest post summarizing Paul Christiano's proposed scheme for training machine learning systems that can be robustly aligned to complex and fuzzy values, which I call Iterated Distillation and Amplification (IDA) here. IDA is notably similar to AlphaGoZero and expert iteration.

The hope is that if we use IDA to train each learned component of an AI then the overall AI will remain aligned with the user's interests while achieving state of the art performance at runtime—provided that any non-learned components such as search or logic are also built to preserve alignment and maintain runtime performance. This document gives a high-level outline of IDA.

Motivation: The alignment/capabilities tradeoff

Assume that we want to train a learner **A** to perform some complex fuzzy task, e.g. "Be a good personal assistant." Assume that **A** is *capable* of learning to perform the task at a superhuman level—that is, if we could perfectly specify a "personal assistant" objective function and trained **A** to maximize it, then **A** would become a far better personal assistant than any human.

There is a spectrum of possibilities for how we might train **A** to do this task. On one end, there are techniques which allow the learner to discover powerful, novel policies that improve upon human capabilities:

- Broad reinforcement learning: As **A** takes actions in the world, we give it a relatively sparse reward signal based on how satisfied or dissatisfied we are with the eventual consequences. We then allow **A** to optimize for the expected sum of its future rewards
- Broad inverse reinforcement learning: **A** attempts to infer our deep long-term values from our actions, perhaps using a sophisticated model of human psychology and irrationality to select which of many possible extrapolations is correct.

However, it is difficult to specify a broad objective that captures everything we care about, so in practice **A** will be optimizing for some proxy that is not completely aligned with our interests. Even if this proxy objective is "almost" right, its optimum could be disastrous according to our true values.

On the other end, there are techniques that try to narrowly emulate human judgments:

- Imitation learning: We could train **A** to exactly mimic how an expertwould do the task, e.g. by training it to fool a discriminative model trying to tell apart **A**'s actions from the human expert's actions.

- Narrow inverse reinforcement learning: We could train **A** to infer our near-term instrumental values from our actions, with the presumption that our actions are roughly optimal according to those values.
- Narrow reinforcement learning: As **A** takes actions in the world, we give it a dense reward signal based on how reasonable we judge its choices are (perhaps we directly reward state-action pairs themselves rather than outcomes in the world, as in TAMER). **A** optimizes for the expected sum of its future rewards.

Using these techniques, the risk of misalignment is reduced significantly (though not eliminated) by restricting agents to the range of known human behavior—but this introduces severe limitations on capability. This tradeoff between allowing for novel capabilities and reducing misalignment risk applies across different learning schemes (with imitation learning generally being narrowest and lowest risk) as well as within a single scheme.

The motivating problem that IDA attempts to solve: if we are only able to align agents that narrowly replicate human behavior, how can we build an AGI that is both aligned and ultimately much more capable than the best humans?

Core concept: Analogy to AlphaGoZero

The core idea of Paul's scheme is similar to AlphaGoZero (AGZ): We use a learned model many times as a subroutine in a more powerful decision-making process, and then re-train the model to imitate those better decisions.

AGZ's policy network **p** is the learned model. At each iteration, AGZ selects moves by an expensive Monte Carlo Tree Search (MCTS) which uses policy **p**as its prior; **p** is then trained to directly predict the distribution of moves that MCTS ultimately settles on. In the next iteration, MCTS is run using the new more accurate **p**, and **p** is trained to predict the eventual outcome of *that* process, and so on. After enough iterations, a fixed point is reached—**p** is unable to learn how running MCTS will change its current probabilities.

MCTS is an *amplification* of **p**—it uses **p** as a subroutine in a larger process that ultimately makes better moves than **p** alone could. In turn, **p** is a *distillation* of MCTS: it learns to directly guess the results of running MCTS, achieving comparable performance while short-cutting the expensive computation. The idea of IDA is to use the basic iterated distillation and amplification procedure in a much more general domain.

The IDA Scheme

IDA involves repeatedly improving a learned model through an amplification and distillation process over multiple iterations.

Amplification is interactive and human-directed in IDA

In AGZ, the amplification procedure is Monte Carlo Tree Search—it's a simple and well-understood algorithm, and there's a clear mechanism for how it improves on the policy network's original choices (it traverses the game tree more deeply). But in IDA, amplification is not necessarily a fixed algorithm that can be written down once and repeatedly applied; it's an interactive process directed by human decisions.

In most domains, humans are capable of improving their native capabilities by delegating to assistants (e.g. because CEOs can delegate tasks to a large team, they can produce orders of magnitude more output per day than they could on their own). This means if our learning procedure can create an adequate *helper* for the human, the human can use the AI to amplify their ability—this human/AI system may be capable of doing things that the human couldn't manage on their own.

Below I consider the example of using IDA to build a superhuman personal assistant. Let **A**[t] to refer to the state of the learned model after the end of iteration *t*; the initial agent **A**[0] is trained by a human overseer **H**.

Example: Building a superhuman personal assistant

H trains **A**[0] using a technique from the narrow end of the spectrum, such as imitation learning. *Here we are imagining a much more powerful version of "imitation learning" than current systems are actually capable of*—we assume that **A**[0] can acquire nearly human-level capabilities through this process. That is, the trained **A**[0] model executes all the tasks of a personal assistant as **H** would (including comprehending English instructions, writing emails, putting together a meeting schedule, etc).

Even though **A**[0] cannot discover any novel capabilities, it has two key advantages over **H**: it can run much faster, and many copies or versions of it can be run at once. We hope to leverage these advantages to construct a larger system—involving **H** and many copies of **A**[0]—that will substantially improve on **H**'s capabilities while preserving alignment with **H**'s values.

H can use calls to **A**[0] (along with other tools such as external memory) to become a better personal assistant. For example, **H** could assign one copy of **A**[0] to figuring out the best time to schedule the client's recurring team meetings, another copy to figure out what to order the client for lunch, another copy to balance the client's personal budget, etc. **H** now has the ability to get very quick solutions to sub-problems that are roughly as good as the ones **H** would have come up with on their own over a longer time period, and can combine these results to make much better decisions than an unaided human.

Let Amplify(**H**, **A**[0]) refer to the larger system of **H** + many copies of **A**[0] + aids. Compared to **A**[0] alone, the Amplify(**H**, **A**[0]) system has much higher time and resource costs but its eventual decisions are much better. Moreover, because in each of its individual decisions each copy of **A**[0] continues to act just as a human personal assistant would act, we can hope that Amplify(**H**, **A**[0]) preserves alignment.

In the next iteration of training, the Amplify(**H**, **A**[0]) system takes over the role of **H** as the overseer. **A**[1] is trained with narrow and safe techniques to quickly reproduce the results of Amplify(**H**, **A**[0]). Because we assumed Amplify(**H**, **A**[0]) was aligned, we

can hope that **A**[1] is also aligned if it is trained using sufficiently narrow techniques which introduce no new behaviors. **A**[1] is then used in Amplify(**H**, **A**[1]), which serves as an overseer to train **A**[2], and so on.

Pseudocode

```
          def IDA(H):
   A <- random initialization
   repeat:
      A <- Distill(Amplify(H, A))

def Distill(overseer):
   """
   Returns an AI trained using narrow, robust techniques to
   perform a task that the overseer already understands how to
   perform.
   """

def Amplify(human, AI):
   """
   Interactive process in which human uses many calls to AI to
   improve on human's native performance at relevant task(s).
   """
```

What properties must hold for IDA to work?

The IDA scheme is a template with “slots” for Amplify and Distill procedures that have not been fully specified yet—in fact, they rely on capabilities we don’t yet have. Because IDA itself is not fully specified, it’s not clear what minimal set of properties are necessary for it to succeed.

Achieving alignment and high capability

That said, here are some general properties which seem necessary—though likely not sufficient—for IDA agents to achieve robust alignment and high capability:

1. *The* Distill *procedure robustly preserves alignment:* Given an aligned agent **H** we can use narrow safe learning techniques to train a much faster agent **A** which behaves as **H** would have behaved, without introducing any misaligned optimization or losing important aspects of what **H** values.
2. *The* Amplify *procedure robustly preserves alignment:* Given an aligned agent **A**, it is possible to specify an amplification scheme which calls **A** multiple times as a subroutine in a way that reliably avoids introducing misaligned optimization.
3. *At least some human experts are able to iteratively apply amplification to achieve arbitrarily high capabilities at the relevant task:* a) there is some threshold of general capability such that if someone is above this threshold, they can eventually solve any problem that an arbitrarily intelligent system could

solve, provided they can delegate tasks to similarly-intelligent assistants and are given arbitrary amounts of memory and time; b) at least some human experts are above this threshold of generality—given enough time and resources, they can figure out how to use AI assistants and tools to improve their capabilities arbitrarily far.

The non-profit Ought is working on gathering more evidence about assumptions 2 and 3.

Achieving competitive performance and efficiency

Paul aims for IDA agents to be competitive with traditional RL agents in time and resource costs at *runtime*—this is a reasonable expectation because an IDA agent is ultimately just another learned model whose weights were tuned with an unusual training procedure.

Resource and time cost during *training* is a more open question; I haven't explored the assumptions that would have to hold for the IDA training process to be practically feasible or resource-competitive with other AI projects.

This was originally posted here.

Benign model-free RL

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

In my last post, I described three research areas in AI control that I see as central: reward learning, robustness, and deliberation.

In this post I argue that these three pieces may be *sufficient* to get a benign and competitive version of model-free reinforcement learning. I think this is an important intermediate goal of solving AI control.

This post doesn't discuss benign model-based RL at all, which I think is another key obstacle for prosaic AI control.

(*This post overlaps extensively with my post on ALBA, but I hope this one will be much clearer. Technically, ALBA is an implementation of the general strategy outlined in this post. I think the general strategy is much more important than that particular implementation.*)

Ingredients

Reward learning and robustness

Given a benign agent H, reward learning allows us to construct a reward function *r* that can be used to train a weaker benign agent A. If our training process is robust, the resulting agent A will remain benign off of the training distribution (though it may be *incompetent* off of the training distribution).

Schematically, we can think of reward learning + robustness as a widget which takes a slow, benign process H and produces a fast, benign process A

A’s capabilities should be roughly the “intersection” of H’s capabilities and our RL algorithms’ competence. That is, A should be able to perform a task whenever *both* H can perform that task and our RL algorithms can learn to perform that task.

In these pictures, the vertical axis corresponds intuitively to “capability,” with higher agents being more capable. But in reality I’m thinking of the possible capabilities as forming a complete lattice. That is, a generic pair of levels of capabilities is incomparable, with neither strictly dominating the other.

Amplification

If we iteratively apply reward learning and robustness, we will obtain a sequence of weaker and weaker agents. To get anywhere, we need some mechanism that lets us produce a *stronger* agent.

The capability amplification problem is to start with a weak agent A and a human expert H, and to produce a significantly more capable agent H^A. The more capable agent can take a lot longer to think, all we care about is that it *eventually* arrives at better decisions than A. The key challenge is ensuring that H^A remains benign, i.e. that the system doesn’t acquire new preferences as it becomes more capable.

An example approach is to provide A as an assistant to H. We can give H an hour to deliberate, and let it consult A thousands of times during that hour. H^A’s output is then whatever H outputs at the end of that process. Because H is consulting A a large number of times, we can hope that the resulting system will be much smarter than A. Of course, the resulting system will be thousands of times more computationally expensive than A, but that’s fine.

In general, meta-execution is my current preferred approach to capability amplification.

Schematically, we can think of amplification as a widget which takes a fast, benign process A and produces a slow, benign process H^A:

Putting it together

With these two widgets in hand, we can iteratively produce a sequence of increasingly competent agents:

That is, we start with our benign expert H. We then learn a reward function and train an agent A, which is less capable than H but can run much faster. By running many instances of A, we obtain a more powerful agent H^A, which is approximately as expensive as H.

We can then repeat the process, using H^A to train an agent A^+ which runs as fast as A but is more capable. By running A^+ for a long time we obtain a still more capable agent H^{A^+}, and the cycle repeats.

Collapsing the recursion

I've described an explicit sequence of increasingly capable agents. This is the most convenient framework for analysis, but actually implementing a sequence of distinct agents might introduce significant overhead. It also feels at odds with current practice, such that I would be intuitively surprised to actually see it work out.

Instead, we can collapse the entire sequence to a single agent:

In this version there is a single agent A which is simultaneously being trained and being used to define a reward function.

Alternatively, we can view this as a sequential scheme with a strong initialization: there is a separate agent at each time *t*, who oversees the agent at time *t*+1, but each agent is initialized using the previous one's state.

This version of the scheme is more likely to be efficient, and it feels much closer to a practical framework for RL. (I originally suggested a similar scheme here.)

However, in addition to complicating the analysis, it also introduces additional challenges and risks. For example, if H^A actually consults A, then there are unattractive equilibria in which A manipulates the reward function, and the manipulated reward function rewards manipulation. Averting this problem either requires H to sometimes avoid depending on A, or else requires us to sometimes run against an old version of A (a trick sometimes used to stabilize self-play). Both of these techniques implicitly reintroduce the iterative structure of the original scheme, though they may do so with lower computational overhead.

We will have an even more serious problem if our approach to reward learning relied on throttling the learning algorithm. When we work with an explicit sequence of agents, we can ensure that their capabilities improve gradually. It's not straightforward to do something analogous in the single agent case.

Overall I think this version of the scheme is more likely to be practical. But it introduces several additional complications, and I think it's reasonable to start by considering the explicit sequential form until we have a solid grasp of it.

Analysis

I'll make two critical claims about this construction. Neither claim has yet been formalized, and it's not clear whether it will be possible to formalize them completely.

Claim #1: All of these agents are benign.

This is plausible by induction:

- The original expert H is benign by definition.
- If we start with a benign overseer H, and have working solutions to reward learning + robustness, then the trained agent A is benign.
- If we start with a benign agent A, and have a woking solution to capability amplification, then the amplified agent H^A will be benign.

There are important subtleties in this argument; for example, an agent may be benign with high probability, and the error probability may increase exponentially as we proceed through the induction. Dealing with these subtleties will require careful definitions, and in some cases adjustments to the algorithm. For example, in the case of increasing failure probabilities, we need to strengthen the statement of amplification to avoid the problem.

Claim #2: The final agent has state-of-the-art performance.

This is plausible if our building blocks satisfy several desirable properties.

First, capability amplification should be able to cross every level non-maximal level of capability. That is, for every level of capability, it is possible to start with an agent A who is below that level, and end up with an agent H^A which is above that level:

For every possible place we could put the dotted line—every possible capability level—there must be some agent A for whom the orange arrow crosses that dotted line. Otherwise we would never be able to get to the other side of that dotted line, i.e. we would never be able to surpass that level of capability.

Second, capability amplification should be monotonic (if A is at least as capable as B then H^A should be at least as capable as H^B).

Third, reward learning should yield an agent whose capabilities are at least the infimum of our RL algorithm's capabilities and the overseer's capabilities, even if we train robustly.

Now given a sequence of increasingly powerful fast agents we can take the supremum of their capabilities. Those agents will all be weaker than our RL algorithms and so the supremum is not the maximal capability, so we can consider a starting point from which capability amplification would cross that supremum. By hypothesis the sequence must eventually cross this starting point, and at that point amplification will push it above the supremum (and reward learning will keep it above the supremum). Making this argument carefully shows that the supremum is the state of the art for RL algorithms and that we attain the supremum after some finite number of steps. (Though all of this is based on a leaky abstraction of "capabilities.")

Cost

I think this proposal will be most helpful if it imposes minimal additional overhead. My main goal is to develop algorithms with sublinear overhead, i.e. for which the fraction of overhead converges to 0 as the underlying algorithms become stronger.

The cost of this scheme depends on the quantitative properties of our basic building blocks:

Factor #1: How much do reward learning and robustness slow down training?

During RL, we need to evaluate the agent A many times. If we want to use a learned reward function we may need to evaluate A more times. And if we want to train a policy which remains benign off of the training distribution, we may need to evaluate A more times (e.g. since we may need to do adversarial training). Ideally that overhead will shrink as our algorithms become more powerful.

I think this is plausible but far from certain (for now it is uncertain whether reward learning and robustness are even plausible). Some reassuring factors:

- Reward learning / adversarial training can actually improve the performance of our system—the computational time spent on them might actually be well-spent even from a capabilities perspective
- The difficulty of the "additional learning problem" we are trying to solve in each case (e.g. the concept of "defer to human control") may not scale up linearly with the complexity of the underlying domain.

Factor #2: how many times do we have to invoke the overseer during training?

In addition to calling the agent A, we will need to call the overseer H in order to get information about the reward function. Because the overseer is much more expensive than the agent, we would like to minimize the number of times we call the overseer. This can be quantified by the ratio between the number of calls to H and the number of calls to A. For example, we may need to call H once for every hundred calls to A.

Factor #3: how expensive is capability amplification?

Capability amplification is possible only because we allow the agent H^A to think for much longer than A. But "much longer" could represent a range of values: is H^A a hundred times more expensive to evaluate than A? A thousand? A million?

Roughly speaking, factors #2 and #3 should be multiplied together to get the overhead from reward learning: factor #2 tells us how many times we have to call the overseer, while factor #3 tells us how expensive the overseer is.

The total overhead is thus (Factor #1) + (Factor #2) * (Factor #3). As an example, I'd be happy with values like 10% + 0.01% × 1000 = 20%.

Factor #4: do we need to train many separate agents?

If we need to use a sequence of N increasingly capable agents, then we would naively increase our training time by a factor of N. Naively, this would dominate the overhead, and in order for the scheme to be workable I think we would need to avoid it. I see a few plausible approaches:

- We could use the collapsed version with a single agent.
- We could use some other initialization or parameter-sharing scheme to effectively reuse the computational work done in training earlier agents.
- The earlier agents could require significantly less training time than the final agent, e.g. because they are less capable. For example, if each agent takes only 20% as long to train as the following one, then the total overhead is only 25%.

These mechanisms can work together; for example, each agent may require some amount of non-reusable computation, but that amount may be reduced by a clever initialization scheme.

Conclusion

I've outlined an approach to AI control for model-free RL. I think there is a very good chance, perhaps as high as 50%, that this basic strategy can eventually be used to train benign state-of-the-art model-free RL agents. Note that this strategy also applies to techniques like evolution that have historically been considered really bad news for control.

That said, the scheme in this post is still extremely incomplete. I have recently prioritized building a practical implementation of these ideas, rather than continuing to work out conceptual issues. That does not mean that I think the conceptual issues are worked out conclusively, but it does mean that I think we're at the point where we'd benefit from empirical information about what works in practice (which is a long way from how I felt about AI control 3 years ago!)

I think the largest technical uncertainty with this scheme is whether we can achieve enough robustness to avoid malign behavior in general.

This scheme does not apply to any components of our system which aren't learned end-to-end. The idea is to use this training strategy for any internal components of our system which use model-free RL. In parallel, we need to develop aligned variants of each other algorithmic technique that plays a role in our AI systems. In particular, I think that model-based RL with extensive planning is a likely sticking point for this program, and so is a natural topic for further conceptual research.

This was originally posted here on 19th March, 2017.

Factored Cognition

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

Note: This post (originally published here) is the transcript of a presentation about a project worked on at the non-profit Ought. It is included in the sequence because it contains a very clear explanation of some of the key ideas behind iterated amplification.

The presentation below motivates our Factored Cognition project from an AI alignment angle and describes the state of our work as of May 2018. Andreas gave versions of this presentation at CHAI (4/25), a Deepmind-FHI seminar (5/24) and FHI (5/25).

I'll talk about *Factored Cognition*, our current main project at Ought. This is joint work with Ozzie Gooen, Ben Rachbach, Andrew Schreiber, Ben Weinstein-Raun, and (as board members) Paul Christiano and Owain Evans.

Scalable learning & planning in complex environments

Before I get into the details of the project, I want to talk about the broader research program that it is part of. And to do that, I want to talk about research programs for AGI more generally.

Right now, the dominant paradigm for researchers who explicitly work towards AGI is what you could call "scalable learning and planning in complex environments". This paradigm substantially relies on training agents in simulated physical environments to solve tasks that are similar to the sorts of tasks animals and humans can solve, sometimes in isolation and sometimes in competitive multi-agent settings.

To be clear, not all tasks are physical tasks. There's also interest in more abstract environments as in the case of playing Go, proving theorems, or participating in goal-based dialog.

Scalable learning & planning in complex environments

- Optimize for success at task
- Internal decision-making learned implicitly

For our purposes, the key characteristic of this research paradigm is that agents are optimized for success at particular tasks. To the extent that they learn particular decision-making strategies, those are learned implicitly. We only provide external supervision, and it wouldn't be entirely wrong to call this sort of approach "recapitulating evolution", even if this isn't exactly what is going on most of the time.

As many people have pointed out, it could be difficult to become confident that a system produced through this sort of process is aligned - that is, that all its cognitive work is actually directed towards solving the tasks it is intended to help with. The reason for this is that alignment is a property of the decision-making process (what the system is "trying to do"), but that is unobserved and only implicitly controlled.

Aside: Could more advanced approaches to transparency and interpretability help here? They'd certainly be useful in diagnosing failures, but unless we can also leverage them for training, we might still be stuck with architectures that are difficult to align.

What's the alternative? It is what we could call *internal supervision* - supervising not just input-output behavior, but also cognitive processes. There is some prior work, with Neural Programmer-Interpreters perhaps being the most notable instance of that class. However, depending on how you look at it, there is currently much less interest in such approaches than in end-to-end training, which isn't surprising: A big part of the appeal of AI over traditional programming is that you don't need to specify how exactly problems are solved.

In this talk, I'll discuss an alternative research program for AGI based on internal supervision. This program is based on imitating human reasoning and meta-reasoning, and will be much less developed than the one based on external supervision and training in complex environments.

The goal for this alternative program is to codify reasoning that people consider "good" ("helpful", "corrigible", "conservative"). This could include some principles of good reasoning that we know how to formalize (such as probability theory and expected value calculations), but could also include heuristics and sanity checks that are only locally valid.

For a system built this way, it could be substantially easier to become confident that it is aligned. Any bad outcomes would need to be produced through a sequence of human-endorsed reasoning steps. This is far from a guarantee that the resulting behavior is good, but seems like a much better starting point. (See e.g. Dewey 2017.)

The hope would be to (wherever possible) punt on solving hard problems such as what decision theory agents should use, and how to approach epistemology and value learning, and instead to build AI systems that inherit our epistemic situation, i.e. that are uncertain about those topics to the extent that we are uncertain.

I've described external and internal supervision as different approaches, but in reality there is a spectrum, and it is likely that practical systems will combine both.

However, the right end of the spectrum - and especially approaches based on learning to reason from humans - are more neglected right now. Ought aims to specifically make progress on automating human-like or human-endorsed deliberation.

A key challenge for these approaches is scalability: Even if we could learn to imitate how humans solve particular cognitive tasks, that wouldn't be enough. In most cases where we figured out how to automate cognition, we didn't just match human ability, but exceeded it, sometimes by a large margin. Therefore, one of the features we'd want an approach to AI based on imitating human metareasoning to have is a story for how we could use that approach to eventually exceed human ability.

Aside: Usually, I fold "aligned" into the definition of "scalable" and describe Ought's mission as "finding scalable ways to leverage ML for deliberation".

Automating deliberation

Task?	Play a game of Go	?
Data?	Recorded human games	?
Objective?	Win the game	?

What does it mean to "automate deliberation"? Unlike in more concrete settings such as playing a game of Go, this is not immediately clear.

For Go, there's a clear task (choose moves based on a game state), there's relevant data (recorded human games), and there's an obvious objective (to win the game). For deliberation, none of these are obvious.

Automating deliberation

Task?	Play a game of Go	Question-answering
Data?	Recorded human games	?
Objective?	Win the game	?

As a task, we'll choose question-answering. This encompasses basically all other tasks, at least if questions are allowed to be big (i.e. can point to external data).

Automating deliberation

Task?	Play a game of Go	Question-answering
Data?	Recorded human games	Recorded human actions in “cognitive workspaces”
Objective?	Win the game	?

The data we'll train on will be recorded human actions in *cognitive workspaces*. I'll show an example in a couple of slides. The basic idea is to make thinking explicit by requiring people to break it down into small reasoning steps, to limit contextual information, and to record what information is available at each step.

An important point here is that our goal is not to capture human reasoning exactly as it is in day-to-day life, but to capture *a* way of reasoning that people would endorse. This is important, because the strategies we need to use to make thinking explicit will necessarily change how people think.

Automating deliberation

Task?	Play a game of Go	Question-answering
Data?	Recorded human games	Recorded human actions in “cognitive workspaces”
Objective?	Win the game	Take cognitive actions that humans would endorse after deliberation*

Finally, the objective will be to choose cognitive actions that people would endorse after deliberation.

Note the weird loop - since our task is automating deliberation, the objective is partially defined in terms of the behavior that we are aiming to improve throughout the training process. This suggests that we might be able to set up training dynamics where the supervision signal always stays a step ahead of the current best system, analogous to GANs and self-play.

I. Explicit deliberation

II. Automation

We can decompose the problem of automating deliberation into two parts:

1. How can we make deliberation sufficiently explicit that we could in principle replicate it using machine learning? In other words, how do we generate the appropriate kind of training data?
2. How do we actually automate it?

In case you're familiar with Iterated Distillation and Amplification: The two parts roughly correspond to amplification (first part) and distillation (second part).

Workspaces & actions

The core concept behind our approach is that of a *cognitive workspace*. A workspace is associated with a question and a human user is tasked with making progress on thinking through that question. To do so, they have multiple actions available:

- They can reply to the question.
- They can edit a scratchpad, writing down notes about intermediate results and ideas on how to make progress on this question.
- They can ask sub-questions that help them answer the overall question.

Sub-questions are answered in the same way, each by a different user. This gives rise to a tree of questions and answers. The size of this tree is controlled by a budget that is associated with each workspace and that the corresponding user can distribute over sub-questions.

The approach we'll take to automating cognition is based on recording and imitating actions in such workspaces. Apart from information passed in through the question and through answers to sub-questions, each workspace is isolated from the others. If we show each workspace to a different user and limit the total time for each workspace to be short, e.g. 15 minutes, we *factor* the problem-solving process in a way that guarantees that there is no unobserved latent state that is accumulated over time.

Workspaces & actions

Internal nodes, using pointers

A leaf node

More pointers

Algorithmic thinking

There are a few more technicalities that are important to making this work in practice.

The most important one is probably the use of *pointers*. If we can only ask plain-text questions and sub-questions, the bandwidth of the question-answering system is severely limited. For example, we can't ask "Why did the protagonist crash the car in book X" because the book X would be too large to pass in as a literal question. Similarly, we can't delegate "Write an inspiring essay about architecture", because the essay would be too large to pass back.

We can lift this restriction by allowing users to create and pass around pointers to datastructures. A simple approach for doing this is to replace plain text everywhere with *messages* that consist of text interspersed with references to other messages.

The combination of pointers and short per-workspace time limits leads to a system where many problems are best tackled in an algorithmic manner. For example, in many situations all a workspace may be doing is mapping a function (represented as a natural language message) over a list (a message with linked list structure), without the user knowing or caring about the content of the function and list.

The human policy

Human policy $\mathrm{H}\colon C \to A$

Ask sub-question:
"How should we estimate {the length of $1}?"

Now let's try to be a bit more precise about the parts of the system we've seen.

One component is the *human policy*, which we treat as a stateless map from contexts (immutable versions of workspaces) to actions (such as asking a particular sub-question).

Coming up with a single such actions should take the human at most a few minutes.

The transition function

Human policy $\mathrm{H}\colon C \to A$

Transition function $\mathrm{T}\colon C \times A \to \{C\}$

The other main component is the transition function, which consumes a context and an action and generates a set of new contexts.

For example, if the action is to ask a sub-question, there will be two new contexts:

1. The successor of the parent context that now has an additional reference to a sub-question.
2. The initial context for the newly generated sub-question workspace.

The entire evaluator

Human policy $\mathrm{H}\colon C \to A$

Transition function $\mathrm{T}\colon C \times A \to \{C\}$

Evaluation step (composition) $\mathrm{E}\colon C \to \{C\}$

Composed together, the human policy and the transition function define a kind of evaluator: A map from a context to a set of new contexts.

Amplification: Iterating the evaluator

Christiano, 2016

In what follows, nodes (depicted as circles) refer to workspaces. Note that both inputs and outputs of workspaces can be messages with pointers, i.e. can be very large objects.

I'll mostly collapse workspaces to just questions and answers, so that we can draw entire trees of workspaces more easily.

By iteratively applying the evaluator, we generate increasingly large trees of workspaces. Over the course of this process, the root question will become increasingly informed by answers to sub-computations, and should thus become increasingly correct and helpful. (What exactly happens depends on how the transition function is set up, and what instructions we give to the human users.)

This process is essentially identical to what Paul Christiano refers to as amplification: A single amplification step augments an agent (in our case, a human question-answerer) by giving it access to calls to itself. Multiple amplification steps generate trees of agents assisting each other.

Calculation

Q What is 1832 * 3290? (concrete)

→ Q What is 1000 * 3290?
A 3290000

Q What is 832 * 3290?

→ Q What is 800 * 3290?
A 2632000

→ Q What is 8 * 3290?
A 26320

→ Q What is 8 * 3000?
A 24000

Q What is 8 * 290?
A 2320

→ Q What is 8 * 200
A 1600

Q What is 8 * 90?
A 720

Q What is 1600 + 720?
A 2320

I'll now walk through a few examples of different types of thinking by recursive decomposition.

The longer-term goal behind these examples is to understand: How decomposable is cognitive work? That is, can amplification work - in general, or for specific problems, with or without strong bounds on the capability of the resulting system?

Perhaps the easiest non-trivial case is arithmetic: To multiply two numbers, we can use the rules of addition and multiplication to break down the multiplication into a few multiplications of smaller numbers and add up the results.

If we wanted to scale to very large numbers, we'd have to represent each number as a nested pointer structure instead of plain text as shown here.

Algorithms & data structures

Q What is the sum of the list $1 [1 $2 [2 $3 [3 $4 [4]]]]?
A The number 10

Q Is $1 empty?
A No

Q What is the first element of $1 ?
A The number 1

Q What is the remainder of $1 ?
A The list $2

Q What is the sum of the list $2 ?
A The number 9

Q Is $2 empty?
A No

Q What is the first element of $2 ?
A The number 2

Q What is the remainder of $2 ?
A The list $3

Q What is the sum of the list $3 ?
A The number 7

We can also implement other kinds of algorithms. Here, we're given a sequence of numbers as a linked list and we sum it up one by one. This ends up looking pretty much the same as how you'd sum up a list of numbers in a purely functional programming language such as Lisp or Scheme.

Universal computation

Q Evaluate expression $1 in environment $2

→ Q Is $1 a function application?
A Yes

Q For $18 [each sub-expression of $1], do $19 [evalute it in environment $2].
A Error: missing env

→ Q Is $18 empty?
A No

Q What is the first element of $18 ?
A The element $20 [sub-expression $14]

Q Apply $19 to $20
A The result is $21

→ Q Evaluate $14 in environment $2
A The result is $21 [the primitive function -]

Q What is the remainder of $18 ?

Indeed, we can implement *any* algorithm using this framework - it is computationally universal. One way to see this is to implement an evaluator for a programming language, e.g. following the example of the meta-circular evaluator in SICP.

As a consequence, if there's a problem we can't solve using this sort of framework, it's not because the framework can't run the program required to solve it. It's because the framework can't *come up with* the program by composing short-term tasks.

Fermi estimation

Q How many standard 2x4 lego blocks are needed to blanket the surface area of Lake Superior?

A Between $1 and $2

Q What is the top area of a 2x4 lego block?

A It is between $6 and $4

Q What is the length of $10 [the 2-side of a 2x4 lego block]?

A It is between $11 and $12

Q What is a lower bound on $10 ?

A It is at least $11 [3 mm]

Q What is an upper bound on $10 ?

A It is at most $12 [2cm]

Q What is the length of $9 [the 4-side of a 2x4 lego block]?

A It is between $7 and $8

Q What is a lower bound on $9 ?

A It is at least $7 [.5 cm]

Q What is an upper bound on $9 ?

A It is at most $8 [4cm]

Q What is $11 * $7 ?

A It is $6 [0.15 cm^2]

Q What is $12 * $8 ?

A Is is $4 [6 cm^2]

Let's start moving away from obviously algorithmic examples. This example shows how one could generate a Fermi estimate of a quantity by combining upper and lower bounds for the estimates of component quantities.

Probabilistic reasoning

Q Suppose we flip three fair coins and get at least two heads. What's the probability that the first coin got heads?

→ Q What outcomes are there if we run the process $1 [flip three fair coins with names 1-3] and how likely is each of them?

A The map from outcomes to probabilities $3

→ Q Is it possible to decompose process $1 into two independent processes?

A Yes

Q Decompose process $1 into two independent processes

A The pair of processes $14 [($15 [flip two coins with names 1-2], $16 [flip a coin with name 3])]

Q What outcomes are there if we run the process $15 and how likely is each of them?

A The map from outcomes to probabilities $6

→ Q Is it possible to decompose process $15 into two independent processes?

A Yes

Q Decompose process $15 into two independent processes

A The pair of processes $17 [($18 [flip a coin with name 1], $19 [flip a coin with name 2])]

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

This example hints at how one might implement conditioning for probability distributions. We could first generate a list of all possible outcomes together with their associated probabilities, then filter the list of outcomes to only include those that satisfy our condition, and renormalize the resulting (sub-)distribution such that the probabilities of all outcomes sum to one again.

The general principle here is that we're happy to run very expensive computations as long as they're semantically correct. What I've described for conditioning is more or less the textbook definition of exact inference, but in general that is computationally intractable for distributions with many variables. The reason we're happy with expensive computations is that eventually we won't instantiate them explicitly, but rather emulate them using cheap ML-based function approximators.

Language understanding

If we want to use this framework to implement agents that can eventually exceed human capability, we can't use most human object-level knowledge, but rather need to set up a process that can learn human-like abilities from data in a more scalable way.

Consider the example of understanding natural language: If we wanted to determine whether a pair of sentences is a contradiction, entailment, or neutral (as in the SNLI dataset), we could simply ask the human to judge - but this won't scale to languages that none of the human judges know.

Alternatively, we can break down natural language understanding into (very) many small component tasks and try to solve the task without leveraging the humans' native language understanding facilities much. For example, we might start by computing the meaning of a sentence as a function of the meanings of all possible pairs of sub-phrases.

As in the case of probabilistic inference, this will be computationally intractable, and getting the decomposition right in the first place is substantially harder than solving the object-level task.

Math proofs

Here's a class of problems that seems particularly challenging for factored cognition: Problems where people would usually learn concepts over an extended period of time.

Consider solving a problem that is posed halfway through a math textbook. Usually, the textbook reader would have solved many simpler problems up to this point and would have built up conceptual structures and heuristics that then allow them to solve this new problem. If we need to solve the problem by composing work done by a large collection of humans, none of which can spend more than 15 minutes on the task, we'll have to replace this intuitive, implicit process with an externalized, explicit alternative.

It's not entirely clear to me how to do that, but one way to start would be to build up knowledge about the propositions and entities that are part of the problem statement by effectively applying semantic parsing to the relevant parts of the textbook, so that we can later ask whether (e.g.) a proposition with meaning X implies a proposition with meaning Y, where both X and Y are large nested pointer structures that encode detailed meaning representations.

If this reminds you of Good Old-Fashioned AI, it is not by accident. We're essentially trying to succeed where GOFAI failed, and our primary advantage is that we're okay with exponentially expensive computations, because we're not planning to ever run them directly. More on that soon.

Low and high bandwidth

Low

High

Cf. Saunders on LessWrong: goo.gl/dLvFbs

So far, the workspaces we've looked at were quite sparse. All questions and answers were limited to a sentence or two. This "low-bandwidth" setting is not the only way to use the system - we could alternatively instruct the human users to provide more detail in their questions and to write longer answers.

For the purpose of automation, low bandwidth has advantages, both in the short term (where it makes automation easier) and in the long term (where it reduces a particular class of potential security vulnerabilities).

Empirical evidence from experiments with humans will need to inform this choice as well, and the correct answer is probably at least slightly more high-bandwidth than the examples shown so far.

Causal reasoning

Here's a kind of reasoning that I feel relatively optimistic that we can implement using factored cognition: Causal reasoning, both learning causal structures from data as well as computing the results of interventions and counterfactuals.

The particular tree of workspaces shown here doesn't really illustrate this, but I can imagine implementing Pearl-style algorithms for causal inference in a way where each step locally makes sense and slightly simplifies the overall problem.

Meta-reasoning

Q Should Ought do other activities besides the current planned research?

→ Q Should $1 [Ought] do $2 [Execute other activities besides the current planned research]?

→ Q How should I approach the question $3 [Should $1 do $2]?

→ Q How should I approach the question $4 [How should I approach the question $3 ?]?

→ Q How should I approach the question $5 [How should I approach the question $4 ?]?

S $7 [$8 [- I could think about similar questions and list approaches to answering them]
$13 [$9 [- I could list features that a good answer to this question would have and generate approaches that result in answers with those features]
$12 [$10 [- I could look at the content of the question and see if it prompts ideas about how to answer it]
$11 [- I could think about possible approaches to questions abstractly (without looking at this one) and then specialize]]]]

A I should do $6 [Follow each approach in $7 and aggregate the results]

Q Execute approach $6 to answer question $4

→ Q Follow each approach in $7 in order to answer question $4

→ Q For each $7 do $14 [Follow the approach in order to answer question $4]

The final example, meta-reasoning, is in some ways the most important one: If we want factored cognition to eventually produce *very* good solutions to problems - perhaps being competitive with any other systematic approach - then it's not enough to rely on the users directly choosing a good object-level decomposition for the problem at hand. Instead, they'll need to go meta and use the system to reason about what decompositions would work well, and how to find them.

One kind of general pattern for this is that users can ask something like "What approach should we take to problem #1?" as a first sub-problem, get back an answer #2, and then ask "What is the result of executing approach #2 to question #1?" as a second sub-question. As we increase the budget for the meta-question, the object-level approach can change radically.

And, of course, we could also go meta twice, ask about approaches to solving the first meta-level problem, and the same consideration applies: Our meta-level approach to finding good object-level approaches could improve substantially as we invest more budget in meta-meta.

Design space for the transition function

- How does $\mathrm{T}: C \times A \to \{C\}$ work?
  - Structure of C ?
    {Workspace with sub-questions, document, tree, …}
  - Actions available to H ?
    {Reply, Ask, Edit, Expand, Reflect, Message, …}

More: ought.org/projects/factored-cognition/taxonomy

So far, I've shown one particular instantiation of factored cognition: a way to structure workspaces, a certain set of actions, and a corresponding implementation of the transition function that generates new workspace versions.

By varying each of these components, we can generate other ways to build systems in this space. For example, we might include actions for asking clarifying questions. I've written about these degrees of freedom on our taxonomy page.

hch.ought.org

Here's one example of an alternate system. This is a straightforward Javascript port of parts of Paul Christiano's ALBA implementation.

Workspaces are structured as sequences of observations and actions. All actions are commands that the user types, including `ask`, `reply`, `view` (for expanding a pointer), and `reflect` (for getting a pointer to the current context).

The command-line version is available on Github.

Open source

oughtinc / **hch**

oughtinc / **patchwork** NEW

github.com/oughtinc

A few days ago, we open-sourced Patchwork, a new command-line app for recursive question-answering where we paid particular attention to build it in a way that is a good basis for multiple users and automation. To see a brief screencast, take a look at the README.

I. Explicit deliberation

II. Automation

Suppose decomposition worked and we could solve difficult problems using factored cognition - how could we transition from only using human labor to partial automation and eventually full automation? I'll discuss a few approaches, starting from very basic ideas that we can implement now and progressing to ones that will not be tractable using present-day ML.

Basic automation

Let's again consider a tree of workspaces, and in each workspace, one or more humans taking one or more actions.

For simplicity, I'll pretend that there is just a single action per workspace. This allows me to equivocate nodes and actions below. Nothing substantial changes if there are multiple actions.

I'll also pretend that all humans are essentially identical, which is obviously false, but allows me to consider the simpler problem of learning a single human policy from data.

Basic automation

- Memoize H

As a first step towards automation, we'll *memoize* the human H. That is, whenever we would show a context to H, we first check whether we've shown this context to some other H before, and if so, we directly reuse the action that was taken previously.

This is a big win if many contexts are simple. For example, it may be very common to want to map a function over a list, and this operation will always involve the same kinds of sub-questions (check if the list is empty, if not get the first element, apply the function to the first element, etc). Ideally, we only do this sort of work once and then reuse it in the future. Memoization gets us part of the way there.

Basic automation

- Memoize H
- Learn to imitate H

Crucial property:
Robust generalization in
AI-complete domains

A significant step up in difficulty, we can try to imitate the behavior of H even in cases where the new context doesn't match any element of our "training set" exactly.

Of course, for general question-answering, we won't be able to fully automate the human policy any time soon. This means that any imitation algorithms we do apply will need to make choices about whether a context under consideration is the sort of situation where they can make good predictions about what a human would do, or whether to abstain.

If algorithms can make multiple choices in sequence, we need algorithms that are well-calibrated about when their actions are appropriate, and that in particular have very few false positives. Otherwise, even a relatively low probability of false positives could cascade into sequences of inappropriate actions.

We've tried to isolate and study this particular problem - making well-calibrated predictions in AI-complete domains - in a separate project called Predicting Slow Judgments. So far, we've found it challenging to make non-trivial predictions about human responses for the dataset we've collected there.

Basic automation

- Memoize H
- Learn to imitate H

Crucial property:
Robust generalization in
AI-complete domains

How useful would it be to be able to automate some fraction of human actions? If the total number of actions needed to solve a task is exponentially large (e.g. because we're enumerating all potential sub-phrases of a paragraph of text), even being able to automate 90% of all actions wouldn't be enough to make this approach computationally tractable. To get to tractability in that regime, we need to automate entire subtrees. (And we need to do so using an amount of training data that is not itself exponentially large - an important aspect that this talk won't address at all.)

Implicit exponential-time computation via ML

Christiano 2017, via Cotra 2018: “Iterated Distillation and Amplification”

Let's reconsider amplification. Recall that in this context, each node represents the question-answer behavior implemented by a workspace operated on by some agent (to start with, a human). This agent can pose sub-questions to other agents who may or may not themselves get to ask such sub-questions, as indicated by whether they have nodes below them or not.

Each step grows the tree of agents by one level, so after *n* steps, we have a tree of size $O(2^n)$. This process will become intractable before long.

(The next few slides describe Paul Christiano's Iterated Distillation and Amplification approach to training ML systems.)

Implicit exponential-time computation via ML

Christiano 2017, via Cotra 2018: “Iterated Distillation and Amplification”

Instead of iterating amplification, let's pause after one step. We started out with a single agent (a human) and then built a composite system using multiple agents (also all humans). This composite system is slower than the one we started out with. This slowdown perhaps isn't too bad for a single step, but it will add up over the course of multiple steps. To iterate amplification many times, we need to avoid this slowdown. What can we do?

Implicit exponential-time computation via ML

Christiano 2017, via Cotra 2018: “Iterated Distillation and Amplification”

The basic idea is to train an ML-based agent to imitate the behavior of the composite system. A simple (but insufficient!) approach would be to generate training data - questions and answers - based on the behavior of the composite system, and to train a supervised learner using this dataset.

Christiano 2017, via Cotra 2018: “Iterated Distillation and Amplification”

In practice, this sort of training ("distillation") would probably need to involve not just imitation, but more advanced techniques, including adversarial training and approaches to interpretability that allow the composite system (the "overseer") to reason about the internals of its fast ML-based successor.

If we wanted to implement this training step in rich domains, we'd need ML techniques that are substantially better than the state of the art as of May 2018, and even then, some domains would almost certainly resist efficient distillation.

Christiano 2017, via Cotra 2018: “Iterated Distillation and Amplification”

But, hypothetically, if we could implement faithful distillation, we would have a much better starting point for the next amplification step: We could compose together multiple instances of the fast ML-based learner, and the result would be a tree of agents that is only as large as the one we built in the first step (3 nodes, say), but exhibits the question-answer behavior of an agent that has multiple advisors, *each of which* as capable as the entire tree at the first step.

Christiano 2017, via Cotra 2018: “Iterated Distillation and Amplification”

We can repeat whatever training process we used in the first step to get a yet better distilled system that "imitates" the behavior of the overseer composed of the systems trained in the previous step.

Christiano 2017, via Cotra 2018: “Iterated Distillation and Amplification”

Through repeated amplification and distillation, we could hope to eventually satiate the representational and computational abilities of whatever ML system we're using in the distillation step, while guiding it towards implementing question-answer behavior that corresponds to what H would do if they had a large number of well-resourced assistants.

Implicit exponential-time computation via ML

Christiano 2017, via Cotra 2018: “Iterated Distillation and Amplification”

In practice, we might not want to implement this process as a series of distinct systems, and instead run self-play where a single system serves both as the overseer and the system-to-be-trained.

Implicit exponential-time computation via ML

If Iterated Amplification and Distillation can work, we might be able to approximate the results of running some computations that would naively take exponential time: after *n* steps of amplification and distillation, we'd use a fast ML-based approximator to run computations that would take $O(2^n)$ time if we instantiated them explicitly.

As a particularly interesting special case, this might include the kinds of human-guided computations that arise from people take actions in cognitive workspaces.

Open questions

- Feasibility
  - Decompose all tasks?
  - Reach arbitrary capability?

- Challenge problems

There are many open questions for the scheme described above, both on whether we can make reasoning explicit, and on whether we can automate it efficiently even if it is made explicit. While I've talked a bit about automation, anything beyond basic automation is out of scope for Ought right now, so I'll focus on open questions related to decomposition.

For decomposition, the two main questions we ask ourselves are:

1. Can factored cognition recover the ability of a single human working over time for essentially all important tasks?
2. If so, can we exceed the capability of other systematic approaches to problem-solving if we just use sufficiently large budgets, i.e. compose sufficiently many small workspaces in sufficiently large trees? Equivalently, can we reach essentially arbitrarily high capability if we execute sufficiently many amplifications steps?

Our plan is to study both of these questions using a set of challenge problems.

Challenge problems

- Creative insight
  → math & programming puzzles

- Learning over time
  → textbook questions

- Incremental reasoning about evidence
  → fact-checking

- Interaction with stateful world
  → dialog for cost-benefit analysis

- Meta-reasoning
  → prioritizing tasks

The idea behind these challenge problems is to pick problems that are particularly likely to stretch the capabilities of problem solving by decomposition:

1. When people tackle tricky *math or programming puzzles*, they sometimes give up, go to bed, and the next day in the shower they suddenly know how to solve it. Can we solve such puzzles even if no single individual spends more than 15 minutes on the problem?
2. We've already seen a math textbook example earlier. We want to know more generally whether we can replicate the effects of learning over time, and are planning to study this using different kinds of *textbook problems*.
3. Similarly, when people reason about evidence, e.g. about whether a statement that a politician made is true, they seem to make incremental updates to opaque internal models and may use heuristics that they find difficult to verbalize. If we instead require all evidence to be aggregated explicitly, can we still match or exceed their *fact-checking* capabilities?
4. All examples of problems we've seen are one-off problems. However, ultimately we want to use automated systems to interact with a stateful world, e.g. through *dialog*. Abstractly, we know how to approach this situation, but we'd like to try it in practice e.g. on personal questions such as "Where should I go on vacation?".
5. For systems to scale to high capability, we've noted earlier that they will need to reason about cognitive strategies, not just object-level facts. *Prioritizing tasks* for a user might be a domain particularly suitable for testing this, since the same kind of reasoning (what to work on next) could be used on both object- and meta-level.

Open questions

- Feasibility
  - Decompose all tasks?
  - Reach arbitrary capability?

- The human policy
  - What knowledge?
  - What size?

$$\mathrm{H}\colon C \to A$$

- Dynamics of deliberation
  - Remain corrigible?
  - Basin of attraction, or chaotic?

If we make progress on the feasibility of factored cognition and come to believe that it might be able to match and eventually exceed "normal" thinking, we'd like to move towards learning more about how this process would play out.

What would the human policy - the map from contexts to actions - look like that would have these properties? What concepts would be part of this policy? For scaling to high capability, it probably can't leverage most of the object-level knowledge people have. But what else? Abstract knowledge about how to reason? About causality, evidence, agents, logic? And how big would this policy be - could we effectively treat it as a lookup table, or are there many questions and answers in distinct domains that we could only really learn to imitate using sophisticated ML?

What would happen if we scaled up by iterating this learned human policy many times? What instructions would the humans that generate our training data need to follow for the resulting system to remain corrigible, even if run with extremely large amounts of computation (as might be the case if distillation works)? Would the behavior of the resulting system be chaotic, strongly dependent on its initial conditions, or could we be confident that there is a basin of attraction that all careful ways of setting up such a system converge to?

Supervising strong learners by amplifying weak experts

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.
This is a linkpost for https://arxiv.org/pdf/1810.08575.pdf

Abstract

Many real world learning tasks involve complex or hard-to-specify objectives, and using an easier-to-specify proxy can lead to poor performance or misaligned behavior. One solution is to have humans provide a training signal by demonstrating or judging performance, but this approach fails if the task is too complicated for a human to directly evaluate. We propose Iterated Amplification, an alternative training strategy which progressively builds up a training signal for difficult problems by combining solutions to easier subproblems. Iterated Amplification is closely related to Expert Iteration (Anthony et al., 2017; Silver et al., 2017b), except that it uses no external reward function. We present results in algorithmic environments, showing that Iterated Amplification can efficiently learn complex behaviors.

Tomorrow's AI Alignment Forum sequences post will be 'AI safety without goal-directed behavior' by Rohin Shah, in the sequence on Value Learning.

The next post in this sequence on Iterated Amplification will be 'AlphaGo Zero and capability amplification', by Paul Christiano, on Tuesday 8th January.

AlphaGo Zero and capability amplification

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

AlphaGo Zero is an impressive demonstration of AI capabilities. It also happens to be a nice proof-of-concept of a promising alignment strategy.

How AlphaGo Zero works

AlphaGo Zero learns two functions (which take as input the current board):

- A prior over moves **p** is trained to predict what AlphaGo will eventually decide to do.
- A value function **v** is trained to predict which player will win (if AlphaGo plays both sides)

Both are trained with supervised learning. Once we have these two functions, AlphaGo actually picks it moves by using 1600 steps of Monte Carlo tree search (MCTS), using **p** and **v** to guide the search. It trains **p** to bypass this expensive search process and directly pick good moves. As **p** improves, the expensive search becomes more powerful, and **p** chases this moving target.

Iterated capability amplification

In the simplest form of iterated capability amplification , we train one function:

- A “weak” policy **A** , which is trained to predict what the agent will eventually decide to do in a given situation.

Just like AlphaGo doesn’t use the prior **p** directly to pick moves, we don’t use the weak policy **A** directly to pick actions. Instead, we use a capability amplification scheme: we call **A** many times in order to produce more intelligent judgments. We train **A** to bypass this expensive amplification process and directly make intelligent decisions. As **A** improves, the amplified policy becomes more powerful, and **A** chases this moving target.

In the case of AlphaGo Zero, **A** is the prior over moves, and the amplification scheme is MCTS. (More precisely: **A** is the pair (**p**, **v**), and the amplification scheme is MCTS + using a rollout to see who wins.)

Outside of Go, **A** might be a question-answering system, which can be applied several times in order to first break a question down into pieces and then separately answer each component. Or it might be a policy that updates a cognitive workspace, which can be applied many times in order to “think longer” about an issue.

The significance

Reinforcement learners take a reward function and optimize it; unfortunately, it's not clear where to get a reward function that faithfully tracks what we care about. That's a key source of safety concerns.

By contrast, AlphaGo Zero takes a policy-improvement-operator (like MCTS) and converges towards a fixed point of that operator. If we can find a way to improve a policy *while preserving its alignment*, then we can apply the same algorithm in order to get very powerful but aligned strategies.

Using MCTS to achieve a simple goal in the real world wouldn't preserve alignment, so it doesn't fit the bill. But "think longer" might. As long as we start with a policy that is close enough to being aligned—a policy that "wants" to be aligned, in some sense—allowing it to think longer may make it both smarter *and* more aligned.

I think designing alignment-preserving policy amplification is a tractable problem today, which can be studied either in the context of existing ML or human coordination. So I think it's an exciting direction in AI alignment. A candidate solution could be incorporated directly into the AlphaGo Zero architecture, so we can already get empirical feedback on what works. If by good fortune powerful AI systems look like AlphaGo Zero, then that might get us much of the way to an aligned AI.

This was originally posted here on 19th October 2017.

Tomorrow's AI Alignment Forum sequences will continue with a pair of posts, 'What is narrow value learning' by Rohin Shah and 'Ambitious vs. narrow value learning' by Paul Christiano, from the sequence on Value Learning.

The next post in this sequence will be 'Directions for AI Alignment' by Paul Christiano on Thursday.

Directions and desiderata for AI alignment

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

*Note: This is the first post from **part four: what needs doing** of the sequence on iterated amplification. The fourth part of the sequence describes some of the black boxes in iterated amplification and discusses what we would need to do to fill in those boxes. I think these are some of the most important open questions in AI alignment.*

In the first half of this post, I'll discuss three research directions that I think are especially promising and relevant to AI alignment:

1. **Reliability and robustness.** Building ML systems which behave acceptably in the worst case rather than only on the training distribution.
2. **Oversight / reward learning.** Constructing objectives and training strategies which lead our policies to do what we intend.
3. **Deliberation and amplification.** Surpassing human performance without simultaneously abandoning human preferences.

I think that we have several angles of attack on each of these problems, and that solutions would significantly improve our ability to align AI. My current feeling is that these areas cover much of the key work that needs to be done.

In the second half of the post, I'll discuss three desiderata that I think should guide research on alignment:

1. **Secure**. Our solutions should work acceptably even when the environment itself is under the influence of an adversary.
2. **Competitive**. Our solutions should impose minimal overhead, performance penalties, or restrictions compared to malign AI.
3. **Scalable.** Our solutions should continue to work well even when the underlying learning systems improve significantly.

I think that taking these requirements seriously leads us to substantially narrow our focus.

It may turn out that these desiderata are impossible to meet, but if so I think that the first order of business should be understanding clearly *why* they are impossible. This would let us better target our work on alignment and better prepare for a future where we won't have a completely satisfying solution to alignment.

(The ideas in this post are not novel. My claimed contribution is merely collecting these things together. I will link to my own writing on each topic in large part because that's what I know.)

I. Research directions

1. Reliability and robustness

Traditional ML algorithms optimize a model or policy to perform well on the training distribution. These models can behave arbitrarily badly when we move away from the training distribution. Similarly, they can behave arbitrarily badly on a small part of the training distribution.

I think this is bad news:

- Deploying ML systems will critically change their environment, in a way that is hard or impossible to simulate at training time. (The "treacherous turn" is a special case of this phenomenon.)
- Deployed ML systems are interconnected and exposed to the same world. So if conditions change in a way that causes one of them to fail, *many*systems may fail simultaneously.
- If ML systems are extremely powerful, or if they play a critical role in society, then a widespread failure may have catastrophic consequences.

I'm aware of three basic approaches to reliability that seem to me like they could plausibly scale and be competitive:

(*ETA: this list is superseded by the list in Techniques for Optimizing Worst-Case Performance. I removed consensus and added interpretability and verification. I don't discuss "learning the right model," which I still consider a long shot.*)

- **Adversarial training**. At training time, attempt to construct inputs that induce problematic behavior and train on those. Eventually, we hope there will be no catastrophe-inducing inputs left. We don't yet know what is possible to achieve. (Szegedy 2014, Goodfellow 2015)
- **Ensembling and consensus**. We often have confidence that there exists *some* models which will generalize appropriately. If we can verify that many models agree about an answer, we can be confident that the consensus is correct. If we use this technique, we will often need to abstain on unfamiliar inputs, and in order to remain competitive we will probably need to represent the ensemble implicitly. (Khani 2016)
- **Learning the right model**. If we understood enough about the structure of our model (for example if it reflected the structure of the underlying data-generating process), we might be confident that it will generalize correctly. Very few researchers are aiming for a secure / competitive / scalable solution along these lines, and finding one seems almost (but not completely) hopeless to me. This is MIRI's approach.

Usual caveats apply: these approaches may need to be used in combination; we are likely to uncover completely different approaches in the future; and I'm probably overlooking important existing approaches.

I think this problem is pretty well-understood and well-recognized, but it looks really hard. ML researchers mostly focus on improving performance rather than robustness, and so I think that this area remains neglected despite the problem being well-recognized.

(Previous posts on this blog: *red teams*, *learning with catastrophes*, *thoughts on training highly reliable models*)

2. Oversight / reward learning

ML systems are typically trained by optimizing some objective over the training distribution. For this to yield “good” behavior, the objective needs to sufficiently close to what we really want.

I think this is also bad news:

- Some tasks are very “easy” to frame as optimization problems. For example, we can already write an objective to train an RL agent to operate a profit-maximizing autonomous corporation (though for now we can only train very weak agents).
- Many tasks that humans care about, such as maintaining law and order or helping us better understand our values, are extremely hard to convert into precise objectives: they are inherently poorly-defined or involve very long timescales, and simple proxies can be “gamed” by a sophisticated agent.
- As a result, many tasks that humans care about may not get done well; we may find ourselves in an increasingly sophisticated and complex world driven by completely alien values.

So far, the most promising angle of attack is to optimize extremely complex objectives, presumably by learning them.

I’m aware of two basic approaches to reward learning that seem like they could plausibly scale:

- **Inverse reinforcement learning**. We can observe human behavior in a domain and try to infer what the human is “trying to do,” converting it into an objective that can be used to train our systems. (Russell 1998, Ng 2000, Hadfield-Menell 2016)
- **Learning from human feedback**. We can pose queries to humans to figure out which behaviors or outcomes they prefer, and then optimize our systems accordingly. (Isbell 2001, Thomaz 2006, Pilarski 2011, Knox 2012)

These solutions seem much closer to working than those listed in the previous section on reliability and robustness. But they still face many challenges, and are not yet competitive, scalable, *or* secure:

- IRL requires a prior over preferences and a model of how human behavior relates to human preferences. Current implementations either only work in severely restricted environments, or use simple models of human rationality which cause the learner to attempt to very precisely imitate the human’s behavior (which might be challenging or impossible).
- For similar reasons, existing IRL implementations are not able to learn from other data like human utterances or off-policy behavior, even though these constitute the largest and richest source of data about human preferences.
- Human feedback requires accurately eliciting human preferences, which introduces many complications. (I discuss a few easy problems here.)
- Human feedback is expensive and so we will need to be able to learn from a relatively small amount of labeled data. Demonstrations are also expensive and so may end up being a bottleneck for approaches based on IRL though it’s not as clear.
- Both imitation learning and human feedback may fail when evaluating a behavior requires understanding where the behavior came from. For example, if you ask a human to evaluate a painting they may not be able to easily check

whether it is derivative, even if over the long run they would prefer their AI to paint novel paintings.

(I've described these approaches in the context of "human" behavior, but the expert providing feedback/demonstrations might themselves be a human augmented with AI assistance, and eventually may simply be an AI system that is aligned with human interests.)

This problem has not received much attention in the past, but it seems to be rapidly growing in popularity, which is great. I'm currently working on a project in this area.

(*Previous posts on this blog: the reward engineering problem, ambitious vs. narrow value learning, against mimicry, thoughts on reward engineering.*)

3. Deliberation and amplification

Machine learning is usually applied to tasks where feedback is readily available. The research problem in the previous section aims to obtain quick feedback in general by using human judgments as the "gold standard." But this approach breaks down if we want to exceed human performance.

For example, it is easy to see how we could use machine learning to train ML systems to make human-level judgments about urban planning, by training them to produce plans that sound good to humans. But if we want to train an ML system to make superhuman judgments about how to lay out a city, it's completely unclear how we could do it—without spending billions of dollars trying out the system's ideas and telling it which ones work.

This is a problem for the same reasons discussed in the preceding section. If our society is driven by systems superhumanly optimizing short-term proxies for what we care about—such as how much they impress humans, or how much money they make—then we are liable to head off in a direction which does not reflect our values or leave us in meaningful control of the situation.

If we lowered our ambitions and decide that superhuman performance is inherently unsafe, we would be leaving huge amounts of value on the table. Moreover, this would be an unstable situation: it could last only as long as everyone with access to AI coordinated to pull their punches and handicap their AI systems.

I'm aware of two approaches to this problem that seem like they could scale:

- **IRL [hard mode]**. In principle we can use IRL to recover a representation of human preferences, and then apply superhuman intelligence to satisfy those preferences much better than a human could. However, this is a much more ambitious and challenging form of IRL than is usually discussed, which remains quite challenging even when you set aside all of the usual algorithmic and statistical difficulties. (Jacob Steinhardt and Owain Evans discuss this issue in a recent post.)
- **Iterated amplification**. A group of interacting humans can potentially be smarter than a single human, and a group of AI systems could be smarter than the original AI system. By using these groups as "experts" in place of individual humans, we could potentially train much smarter systems. The key questions are how to perform this composition in a way that causes the group to

implement the same preferences as its members, and whether the cognitive benefits for groups are large enough to overcome the overhead of coordination. (I discuss this approach here and in follow-up work.)
- **IRL for cognition**. Rather than applying IRL to a humans' actions, we could apply it to the cognitive actions taken by a human while they deliberate about a subject. We can then use those values to execute a longer deliberation process, asking "what would the human do if they had more time to think / more powerful cognitive tools?" I think this approach ends up being similar to a blend of the previous two.

It's completely unclear how hard this problem is or how far we are from a solution. It is a much less common research topic than either of the preceding points.

In the short term, I think it might be easier to study analogs of this problem in the context of human behavior than to attempt to directly study it in the context of AI systems.

Ought is a non-profit aimed at addressing (roughly) this problem; I think it is reasonably likely to make significant progress.

(*Previous posts on this blog: capability amplification, reliability amplification, security amplification, meta-execution, the easy goal inference problem is still hard*)

II. Desiderata

I'm most interested in algorithms that are secure, competitive, and scalable, and I think that most research programs are very unlikely to deliver these desiderata (this is why the lists above are so short).

Since these desiderata are doing a lot of work in narrowing down the space of possible research directions, it seems worthwhile to be thoughtful and clear about them. It would be easy to gloss over any of them as obviously unobjectionable, but I would be more interested in people pushing back on the strong forms than implicitly accepting a milder form.

1. Secure

Many pieces of software work "well enough" most of the time; we often learn this not by a deep analysis but by just trying it and seeing what happens. "Works well enough" often breaks down when an adversary enters the prediction.

Whether or not that's a good way to build AI, I think it's a bad way to do alignment research right now.

Instead, we should try to come up with alignment solutions that work in the least convenient world, when nature itself is behaving adversarially. Accomplishing this requires argument and analysis, and cannot be exclusively or based on empirical observation.

AI systems obviously won't work well in the worst case (there is no such thing as a free lunch) but it's reasonable to hope that our AI systems will never respond to a bad

input by actively *trying* to hurt us —at least as long as we remain in physical control of the computing hardware, and the training process, *etc.*

Why does security seem important?

- It's really hard to anticipate what is going to happen in the future. I think it's easy to peer into the mists and say "well, hard to know what's going to happen, but this solution might work out OK," and then to turn out to be too optimistic. It's harder to make this error when we hold ourselves to a higher standard, of actually giving an argument for why things work. I think that this is a general principle for doing useful research in advance of when it is needed—we should hold ourselves to standards that are unambiguous and clear even when the future is murky. This is a theme that will recur in the coming sections.
- We are used to technological progress proceeding slowly compared to timescales of human judgment and planning. It seems quite likely that powerful AI will be developed during or after a period of acceleration, challenging those assumptions and undermining a traditional iterative approach to development.
- The world really does contain adversaries. It's one thing to build insecure software when machines have power over modest amounts of money with significant human oversight, it's another thing altogether when they have primary responsibility for enforcing the law. I'm not even particularly worried about human attackers, I'm mostly worried about a future where all it takes to launch attacks is money (which can itself be earned by executing attacks). Moreover, if the underlying ML is insecure and ML plays a role in almost all software, we are going to have a hard time writing any secure software at all.

(*Previous posts: security and AI alignment*)

2. Competitive

It's easy to avoid building an unsafe AI system (for example: build a spreadsheet instead). The only question is how much you have to sacrifice to do it.

Ideally we'll be able to build benign AI systems that are just as efficient and capable as the best AI that we could build by any means. That means: we don't have to additional domain-specific engineering work to align our systems, benign AI doesn't require too much more data or computation, and our alignment techniques don't force us to use particular techniques or restrict our choices in other ways.

(More precisely, I would consider an alignment strategy a success if the additional costs are sublinear: if the fraction of resources that need to be spent on alignment research and run-time overhead *decreases* as the AI systems become more powerful, converging towards 0.)

Why is competitiveness important?

A. It's easy to tell when a solution is plausibly competitive, but very hard to tell exactly how uncompetitive an uncompetitive solution will be. For example, if a purported alignment strategy requires an AI not to use technique or development strategy X, it's easy to tell that this proposal isn't competitive in general, but very hard to know exactly how uncompetitive it is.

As in the security case, it seems very easy to look into the fog of the future and say "well this seems like it will probably be OK" and then to turn out to be too optimistic. If we hold ourselves to the higher standard of competitiveness, it is much easier to stay honest.

Relatedly, we want alignment solutions that work across an extremely large range of techniques not just because we are uncertain about which techniques will be important, but because generalizing across all of the situations we can foresee is a good predictor of working for situations we can't foresee.

B. You can't unilaterally use uncompetitive alignment techniques; we would need global coordination to avoid trouble. If we *don't* know how to build competitive benign AI, then users/designers of AI systems have to compromise efficiency in order to maintain reliable control over those systems. The most efficient systems will by default be built by whoever is willing to accept the largest risk of catastrophe (or perhaps by actors who consider unaligned AI a desirable outcome).

It may be possible to avert this kind of race to the bottom by effective coordination by e.g. enforcing regulations which mandate adequate investments in alignment or restrict what kinds of AI are deployed. Enforcing such controls domestically is already a huge headache. But internationally things are even worse: a country that handicapped its AI industry in order to proceed cautiously would face the risk of being overtaken by a less prudent competitor, and avoiding *that* race would require effective international coordination.

Ultimately society will be able and willing to pay *some* efficiency cost to reliably align AI with human interests. But the higher that cost, the harder the coordination problem that we will need to solve. I think the research community should be trying to make that coordination problem as easy as possible.

(*Previous posts: prosaic AI alignment, a possible stance for AI control, efficient and safely scalable*)

3. Scalable

Over time, we are acquiring more data, more powerful computers, richer model classes, better optimization algorithms, better exploration strategies, and so on. If we extrapolate these trends, we end up with very powerful models and policies.

Many approaches to alignment break down at some point in this extrapolation. For example, if we train an RL agent with a reward function which imperfectly approximates what we want, it is likely to fail once the agent becomes sufficiently sophisticated—unless the reward function itself becomes more sophisticated in parallel.

In contrast, let's say that a technique is "scalable" if it continues to work just as well even when the underlying learning becomes much more powerful. (See also: Eliezer's more colorful "omnipotence test.")

This is another extremely demanding requirement. It rules out many possible approaches to alignment. For example, it probably rules out any approach that involves hand-engineering reward functions. More subtly, I expect it will rule out any

approach that requires hand-engineering an informative prior over human values (though some day we will hopefully find a scalable approach to IRL).

Why is scalability important?

- As in the previous sections, it's easy to be too optimistic about exactly when a non-scalable alignment scheme will break down. It's much easier to keep ourselves honest if we actually hold ourselves to producing scalable systems.
- If AI progress rapidly, and especially if AI research is substantially automated, then we may literally confront the situation where the capabilities of our AI systems are changing rapidly. It would be desirable to have alignment schemes that continued to work in this case.
- If we don't have scalable solutions then we require a continuing investment of research on alignment in order to "keep up" with improvements in the underlying learning. This risks compromising competitiveness, forcing AI developers to make a hard tradeoff between alignment and capabilities. This would be acceptable if the ongoing investments in alignment are modest compared to the investments in capabilities. But as with the last point, that's a very murky question about which it seems easy to be overly optimistic in advance. If we think the problem will be easy in the future when we have more computing, then we ought to be able to do it now. Or at the very least we ought to be able to explain how more computing will make it easy. If we make such an explanation sufficiently precise then it will itself become a scalable alignment proposal (though perhaps one that involves ongoing human effort).

(*Previous posts: scalable AI control*)

Aside: feasibility

One might reject these desiderata because they seem too demanding: it would be great if we had a secure, competitive and scalable approach to alignment, but that might not be possible.

I am interested in trying to satisfy these desiderata despite the fact that they are quite demanding, for two reasons:

- I think that it is very hard to say in advance what is possible or impossible. I don't yet see any fundamental obstructions to achieving these goals, and until I see hard obstructions I think there is a significant probability that the problem will prove to be feasible (or "almost possible," in the sense that we may need to weaken these goals only slightly).
- If there is some fundamental obstruction to achieving these goals, then it would be good to understand that obstruction in detail. Understanding it would help us understand the nature of the problem we face and would allow us to do better research on alignment (by focusing on the key aspects of the problem). And knowing that these problems are impossible, and understanding exactly how impossible they are, helps us prepare for the future, to build institutions and mechanisms that will be needed to cope with unavoidable limitations of our AI alignment strategies.

III. Conclusion

I think there is a lot of research to be done on AI alignment; we are limited by a lack of time and labor rather than by a lack of ideas about how to make progress.

Research relevant to alignment is already underway; researchers and funders interested in alignment can get a lot of mileage by supporting and fleshing out existing research programs in relevant directions. I don't think it is correct to assume that if anyone is working on a problem then it is going to get solved—even amongst things that aren't literally at the "no one else is doing it" level, there are varying degrees of neglect.

At the same time, the goals of alignment are sufficiently unusual that we shouldn't be surprised or concerned to find ourselves doing unusual research. I think that area #3 on deliberation and amplification is almost completely empty, and will probably remain pretty empty until we have clearer statements of the problem or convincing demonstrations of work in that area.

I think the distinguishing feature of research motivated by AI alignment should be an emphasis on secure, competitive, and scalable solutions. I think these are very demanding requirements that significantly narrow down the space of possible approaches and which are rarely explicitly considered in the current AI community.

It may turn out that these requirements are infeasible; if so, one key output of alignment research will be a better understanding of the key obstacles. This understanding can help guide less ambitious alignment research, and can help us prepare for a future in which we won't have a completely satisfying solution to AI alignment.

This post has mostly focused on research that would translate directly into concrete systems. I think there is also a need for theoretical research building better abstractions for reasoning about optimization, security, selection, consequentialism, and so on. It is plausible to me that we will produce acceptable systems with our current conceptual machinery, but if we want to convincingly *analyze* those systems then I think we will need significant conceptual progress (and better concepts may lead us to different approaches). I think that practical and theoretical research will be attractive to different researchers, and I don't have strong views about their relative value.

This was originally posted here on 6th February 2017.

Tomorrow's AI Alignment Forum sequences post will be 'Human-AI Interaction' by Rohin Shah in the sequence on Value Learning.

The next post in this sequence will be 'The reward engineering problem' by Paul Christiano, on Tuesday 15th Jan.

The reward engineering problem

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

Today we usually train reinforcement learning agents to perform narrow tasks with simple goals. We may eventually want to train RL agents to behave "well" in open-ended environments where there is no simple goal.

Suppose that we are trying to train an RL agent **A**. In each episode, **A** interacts with an environment, producing a transcript τ. We then *evaluate* that transcript, producing a reward $r \in [0, 1]$. **A** is trained is to maximize its reward.

We would like to set up the rewards so that **A** will learn to behave well—that is, such that *if* **A** learns to receive a high reward, *then* we will be happy with **A**'s behavior.

To make the problem feasible, we assume that we have access to another agent **H** which

1. is "smarter" than **A**, and
2. makes "good" decisions.

In order to evaluate transcript τ, we allow ourselves to make any number of calls to **H**, and to use any other tools that are available. The question is: how do we carry out the evaluation, so that the optimal strategy for **A** is to also make "good" decisions?

Following Daniel Dewey, I'll call this the *reward engineering problem*.

Note that our evaluation process may be quite expensive, and actually implementing it may be infeasible. To build a working system, we would need to combine this evaluation with semi-supervised RL and learning with catastrophes.

Possible approaches and remaining problems

I know of 3 basic approaches to reward engineering:

1. **Direct supervision**. Use **H** to evaluate **A**'s behavior, and train **A** to maximize **H**'s evaluations. In some contexts we could compare two behaviors instead of evaluating one in isolation.
2. **Imitation learning**. Use **H** to generate a bunch of transcripts, and train **A**to produce similar-looking transcripts. For example, we could train a model to distinguish **A**'s behavior from **H**'s behavior, and reward **A** when it fools the distinguisher.
3. **Inverse reinforcement learning**. Use **H** to generate a bunch of transcripts, and then infer a reward function which is being approximately optimized by **H**. Use this reward function to evaluate **A**'s behavior.

All of these approaches are promising but face significant challenges. I'll describe some of these problems in the next 3 sections.

1. Direct supervision

In direct supervision, **H** looks at a transcript of **A**'s behavior, and estimates how good that transcript is.

To see the problem with this scheme, suppose that **A** has been asked to draw a picture, and **A** does it by copying an existing picture with some modifications. If originality is especially important, then this may be a very "bad" policy. But even if **H** is much smarter than **A**, it may be hard to tell that the picture is not original—creating a derivative work only requires looking at a single existing picture, while checking if a work is derivative requires considering *every* picture.

More formally: in order for direct supervision to be effective, **H** needs to be better-informed than **A** about what is "good." If this condition is satisfied, then from **A**'s perspective, estimating **H**'s estimate of goodness is equivalent to estimating actual goodness. This condition is superficially plausible—after all, we did assume that **H** is smarter than **A**.

The problem is that when **A** picks an action, **A** is especially well-informed about that action—the computation which produced the action provides evidence about it, and **H** may not have access to that evidence.

Transparency

One response is to let **H** see *how* **A** computed its action. If **H** can understand that process, then **H** may be able to effectively evaluate the action.

Sometimes this is straightforward: for example, if **A** uses an attention mechanism to look at a particular painting and copy it, we can simply tell **H**what **A** looked at.

In other contexts it might be much more challenging. For example, **A** may copy an image by "memorizing" the image in its weights, rather than by "looking at" the image at test time.

One approach to these challenging cases is to train **A** to produce actions *and*to produce explanations that are maximally helpful for evaluating those actions. For example, we may train **A** to produce images *and* to point out similarities between its output and training images.

This is a challenging problem for several reasons. One issue is that producing these explanations involves a huge "action space" and a very complex objective. A more subtle problem is that there are two conflicting objectives: **A**wants to produce actions that **H** evaluates as "good," but providing useful information will sometimes lead **H** to produce a lower evaluation. Training **A**to do both tasks requires a new approach.

Other problems

We can imagine other failure modes of direct supervision. For example, **A** may find an action that exploits one of **H**'s biases or blind spots in order to receive a high rating.

We hope that these "attacks" can only succeed if **H** is ignorant about the process that produced a given action, and so can be resolved by whatever form of transparency allows **H** to accurately evaluate **A**'s actions in general.

That is, if **A** carefully explains to **H** how an action was chosen to exploit **H**'s biases, then **H** can hopefully avoid being exploited. This seems especially plausible given that **H** is smarter than **A**.

2. Imitation learning

Imitation learning has two conceptual problems:

- If **H** is more competent than **A**, then **A** will generally be unable to imitate **H**'s behavior.
- We don't have a totally satisfactory framework for reducing imitation learning to an optimization problem.

What if A can't imitate H?

Suppose that **A** has been asked to build a block tower. **H** can quickly stack the blocks, and 99% of the time the tower stays standing; 1% of the time **H**messes up and the tower falls down. **A** is not as capable as **H**, and so if it tries to stack the blocks quickly the tower falls down 100% of the time.

The "best" behavior for **A** may be to stack the blocks more slowly, so that the tower can stay standing. But this behavior is hard to induce with imitation learning, because **H** *never* stacks the blocks slowly. Instead, an imitation learner is more likely to try to stack the blocks quickly and fail (since at least **H** does this 1% of the time).

One response to this problem is to have **H** "dumb down" its behavior so that it can be copied by **A**.

However, this process may be challenging for **H**. Finding a way to do a task which is within **A**'s abilities may be much harder than simply doing the task—for example, it may require a deep understanding of **A**'s limitations and capabilities.

I've proposed a procedure, "meeting halfway," for addressing this problem. The idea is that we train a discriminator to distinguish **H**'s behavior from **A**'s behavior, and use the discriminator's output to help **H** behave in an "**A**-like" way. This proposal faces many challenges, and it's not at all clear if it can work.

How do you train an imitator?

The plagiarism example from the last section is also a challenge for imitation learning. Suppose that **A** has been asked to draw a picture. **H** would draw a completely original picture. How can we train **A** to draw an original picture?

The most plausible existing approach is probably generative adversarial networks. In this approach, a discriminator is trained to distinguish **A**'s behavior from **H**'s behavior, and **A** is trained to fool the discriminator.

But suppose that **A** draws a picture by copying an existing image. It may be hard for the discriminator to learn to distinguish "original image" from "derivative of existing image," for exactly the same reasons discussed before. And so **A** may receive just as high a reward by copying an existing image as by drawing a novel picture.

Unfortunately, solving this problem seems even more difficult for reinforcement learning. We can't give the discriminator any access to **A**'s internal state, since the discriminator isn't supposed to know whether it is looking at data that came from **A** or from **H**.

Instead, it might be easier to use an alternative to the generative adversarial networks framework. There are some plausible contenders, but nothing is currently known that could plausibly scale to general behavior in complex environments. (Though the obstacles are not always obvious.)

3. Inverse reinforcement learning

In IRL, we try to infer a reward function that **H** is approximately maximizing. We can then use that reward function to train **A**.

This approach is closely connected to imitation learning, and faces exactly analogous difficulties:

- **H**'s behavior does not give much information about the reward function in regions far from **H**'s trajectory.
- If we first learn a reward function and then use it to train **A,** then the reward function is essentially a direct supervisor and faces exactly the same difficulties.

The second problem seems to be the most serious: unless we find a resolution to that problem, then direct supervision seems more promising than IRL. (Though IRL may still be a useful technique for the resulting RL problem—understanding the supervisor's values is a critical subtask of solving an RL problem defined by direct supervision.)

H's behavior is not sufficiently informative

Consider the block tower example from the last section. If **H** always quickly builds a perfect block tower, then **H**'s behavior does not give any evidence about tradeoffs between different imperfections: how much should **A** be willing to compromise on quality to get the job done faster? If the tower can only be tall *or* stable, which is preferred?

To get around this difficulty, we would like to elicit information from **H** other than trajectories. For example, we might ask **H** questions, and use those questions as evidence about **H**'s reward function.

Incorporating this information is much less straightforward than incorporating information from **H**'s behavior. For example, updating on **H**'s statements require an explicit model of how **H** believes its statements relate to its goals, even though we can't directly observe that relationship. This is much more complex than existing approaches like MaxEnt IRL, which fit a simple model directly to **H**'s behavior.

These issues are central in "cooperative IRL." For now there are many open problems.

The major difficulties of direct supervision still apply

The bigger problem for IRL is how to represent the reward function:

- If the reward function is represented by a concrete, learned function from trajectories to rewards, then we are back in the situation of direct supervision.
- Instead the reward function may act on an abstract space of "possible worlds." This approach potentially avoids the difficulties of direct supervision, but it seems to require a particular form of model-based RL. It's not clear if this constraint will be compatible with the most effective approaches to reinforcement learning.

Ideally we would find a better representation that incorporates the best of both worlds—avoiding the difficulties of direct supervision, without seriously restricting the form of **A**.

Alternatively, we could hope that powerful RL agents have an appropriate model-based architecture. Or we could do research on appropriate forms of model-based RL to increase the probability that they are competitive.

Research directions

Each of the problems discussed in this post is a possible direction for research. I think that three problems are especially promising:

- Training ML systems to produce the kind of auxiliary information that could make direct supervision reliable. There are open theoretical questions about how this training should be done—and huge practical obstacles to actually making it work.
- Developing alternative objectives for imitation learning or generative modeling. There has been a lot of recent progress in this area, and it is probably worth doing more conceptual work to see if we can find new frameworks.
- Experimenting with "meeting halfway," or with other practical approaches for producing imitable demonstrations.

Conclusion

If we cannot solve the reward engineering problem in practice, it seems unlikely that we will be able to train robustly beneficial RL agents.

Conversely, if we can solve the reward engineering problem, then I believe that solution could be leveraged into an attack on the whole value alignment problem (along these lines—I will discuss this in more detail over my next few posts).

Reward engineering is not only an important question for AI control, but also appears to be tractable today; there are both theoretical and experimental lines of attack. I'm optimistic that we will understand this problem much better over the coming years, and I think that will be very good news for AI control.

(This research was supported as part of the Future of Life Institute FLI-RFP-AI1 program, grant #2015–143898.)

This was originally posted here on 30th May 2016.

The next post in this sequence will be Capability Amplification by Paul Christiano.

Tomorrow's post will be in the sequence on Value Learning by Rohin Shah.

Capability amplification

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

(*Note: In the past I have referred to this process as 'bootstrapping' or 'policy amplification,' but those terms are too broad—there are other dimensions along which policies can be amplified, and 'bootstrapping' is used all over the place.*)

Defining the "intended behavior" of a powerful AI system is a challenge.

We don't want such systems to simply imitate human behavior—we want them to improve upon human abilities. And we don't want them to only take actions that look good to humans—we want them to improve upon human judgment.

We also don't want them to pursue simple goals like "minimize the probability that the bridge falls down" or "pick the winning move." A precise statement of our real goals would be incredibly complicated, and articulating them precisely is itself a massive project. Moreover, we often care about consequences over years or decades. Such long-term consequences would have little use as a *practical* problem definition in machine learning, even if they could serve as a *philosophical* problem definition.

So: what else can we do?

Instead of defining what it means for a policy to be "good," we could define a transformation which turns one policy into a "better" policy.

I call such a transformation *capability amplification*—it "amplifies" a weak policy into a strong policy, typically by using more computational resources and applying the weak policy many times.

Motivation

I am interested in capability amplification because I think it is the most plausible route to defining the goals of powerful AI systems, which I see as a key bottleneck for building aligned AI. The most plausible alternative approach is probably inverse RL, but I think that there are still hard philosophical problems to solve, and that in practice IRL would probably need to be combined with something like capability amplification.

More directly, I think that capability amplification might be a workable approach to training powerful RL systems when combined with semi-supervised RL, adversarial training, and informed oversight (or another approach to reward engineering).

Example of capability amplification: answering questions

Suppose that we would like like to amplify one question-answering system **A** into a "better" question-answering system $\mathbf{A}^+$.

We will be given a question *Q* and an implementation of **A**; we can use **A**, or any other tools at our disposal, to try to answer the question *Q*. We have some time limit; in reality it might be eight hours, but for the purpose of a simple example suppose it is twenty seconds. The amplification $\mathbf{A}^+(Q)$ is defined to be whatever answer we come up with by the end of the time limit. The goal is for this answer to be "better" than the answer that **A** would have given on its own, or to be able to answer harder questions than **A** could have answered directly.

For example, suppose that *Q* = "Which is more water-soluble, table salt or table sugar?" Suppose further that **A** can't answer this question on its own: **A**("Which is more water-soluble…") = "I don't know."

I could start by computing **A**("How do you quantify water-solubility?"); say this gives the answer "By measuring how much of the substance can dissolve in a fixed quantity of water." Then I ask **A**("How much table salt will dissolve in a liter of water?") and get back the answer "360 grams." Then I ask **A**("How much sugar will dissolve in a liter of water?") and get back the answer "2 kilograms." Then I reply "Sugar is about six times more soluble than salt."

Thus $\mathbf{A}^+$("Which is more water-soluble, table salt or table sugar?") = "Sugar is about six times more soluble than salt." This is better than the answer that **A**gave—in some sense, we've successfully amplified **A** into something smarter.

The general problem

The capability amplification problem is to use one policy **A** to implement a new policy $\mathbf{A}^+$ which is strictly "better" than **A**. (Recall that a policy is a mapping from inputs to outputs.)

We'll discuss the definition of "better" in the next section, but for now you can use an intuitive definition. Note that "better" *does not* mean that we can implement $\mathbf{A}^+$ using fewer computational resources than **A**—in fact we will implement $\mathbf{A}^+$ by using a huge amount of computation and time.

What does it mean to "implement" the amplified policy $\mathbf{A}^+$? It means that we have some process that takes as input an observation *o*[1] and produce an action *a*[1]. It then takes as input the next observation *o*[2] and produces the next action *a*[2], and so on.

The process that implements $\mathbf{A}^+$ may instantiate any number of agents who use the policy **A** and interact with them. The process might make copies of any of these agents. And the process can involve us personally thinking about the problem, or using any other tools that we have available—having access to **A**may be a useful resource, but we can also do things from scratch if that's easier.

The capability amplification problem comes with a time limit—we need to provide an implementation that runs within that time limit. (When we submit one action *a*[k], we immediately see the next observation *o*[k+1].) Once the time limit runs out, we automatically output a nil action in response to each additional observation. One way to be "better" is to be able to handle longer sequences of observations.

The time limit could be arbitrary, but I'll pick one day for concreteness. I'll assume that we have an implementation of **A** that runs in one second per episode.

Note that we can start from the trivial policy $\varnothing$ which always outputs nil. In this case, $\varnothing^+$ is a policy that we can implement "from scratch."

Reachability

To measure how well we can solve capability amplification, we'll introduce the concept of *reachability*.

Reachability is defined with respect to a class of policies $\mathscr{A}$ and a preference ordering $\geq$. Intuitively:

- $\mathscr{A}$ is a class of policies that we are capable of implementing efficiently. For example, $\mathscr{A}$ might be the set of policies that can be implemented by a ten layer neural network.
- We say that $\mathbf{A} \geq \mathbf{B}$ if we are *at least as happy* with policy **A** as with policy **B** (in any situation that we think might arise in practice).

We say that **C** is *reachable* from **A** if:

- $\mathbf{A}^+ \geq \mathbf{C}$, where $\mathbf{A}^+$ is the amplification as described in the last section; or
- There is an intermediate $\mathbf{B} \in \mathcal{A}$ which is reachable from **A** and which can reach **C** .

Equivalently:

- **C** is reachable from **A** if there is a chain of policies in $\mathscr{A}$ which starts at **A**and ends at **C**, and where each policy in the chain is no better than the amplification of the previous policy.

The better we are at capability amplification, the more policies will be reachable from any given starting point. Our goal is to have as many policies as possible be reachable from the trivial policy $\varnothing$—ideally, *every* policy in $\mathcal{A}$ would be reachable from $\varnothing$.

Obstructions

An *obstruction* to capability amplification is a partition of the policy class $\mathcal{A}$ into two parts $\mathcal{L}$ and $\mathcal{H}$, such that we cannot amplify *any* policy in $\mathcal{L}$ to be at least as good as *any* policy in $\mathcal{H}$.

Obstructions are dual to reachability in a natural sense. If there are any non-reachable policies, then there is some corresponding obstruction. The desired output of research on capability amplification are a *matching* amplification strategy and obstruction—a way to reach many policies, and an obstruction that implies that we can't reach any more.

Analogously, we say that a function $L : \mathcal{A} \to \mathbb{R}$ is an obstruction if our amplification procedure cannot always increase L. That is, L is an obstruction if there exists a threshold ℓ such that the two sets $\{ \mathbf{A} \in \mathcal{A} : L(\mathbf{A}) \leq \ell \}$ and $\{ \mathbf{A} \in \mathcal{A} : L(\mathbf{A}) > \ell \}$ are an obstruction, or such that $\{ \mathbf{A} \in \mathcal{A} : L(\mathbf{A}) < \ell \}$ and $\{ \mathbf{A} \in \mathcal{A} : L(\mathbf{A}) \geq \ell \}$ are an obstruction.

If we could find a convincing argument that some partition was an obstruction, then that would help further our understanding of value alignment. The next step would be to ask: can we sensibly define "good behavior" for policies in the inaccessible part $\mathcal{H}$? I suspect this will help focus our attention on the most philosophically fraught aspects of value alignment.

In the appendices I give an example of an obstruction in a particular simple model.

Relationship to value alignment

Why capability amplification seems feasible

Capability amplification is a special case of the general problem of "building an AI that does the right thing." It is easier in two respects:

1. In the general problem we need to construct a "good" policy from scratch. In capability amplification we need to construct a good policy $\mathbf{A}^+$ starting from a slightly weaker policy **A**.
2. In the general problem we must *efficiently implement* a good policy. In capability amplification our implementation of $\mathbf{A}^+$ is allowed to take up to a day, even though the goal is to improve upon a policy **A** that runs in one second.

Intuitively, these seem like large advantages.

Nevertheless, it may be that capability amplification contains the hardest aspects of value alignment. If true, I think this would change our conception of the value alignment problem and what the core difficulties are. For example, if capability amplification is the "hard part," then the value alignment problem is essentially orthogonal to the algorithmic challenge of building an intelligence.

Why capability amplification seems useful

Capability amplification can be combined with reward engineering in a natural way:

- Define **A0** = ∅
- Apply capability amplification to obtain $\mathbf{A0}^+$
- Apply reward engineering to define a reward function, and use this to train an agent **A1** which is better than **A0**
- Apply capability amplification to obtain $\mathbf{A1}^+$
- Repeat to obtain a sequence of increasingly powerful agents

This is very informal, and actually carrying out such a process requires resolving many technical difficulties. But it suggests that capability amplification and reward engineering might provide a foundation for training an aligned AI.

What to do?

Theory

The best approach seems to be to work from both sides, simultaneously searching for challenging obstructions and searching for amplification procedures that address those obstructions.

There are at least two very different angles on capability amplification:

- Collaboration: figure out how a bunch of agents using **A** can break a problem down into smaller pieces and attack those pieces separately, allowing them to solve harder problems than they could solve independently.
- Philosophy: try to better understand what "good" reasoning is, so that we can better understand how good reasoning is composed of simpler steps. For example, mathematical proof is a technique which relates hard problems to long sequences of simple steps. There may be more general ideas along similar lines.

In the appendices, I describe some possible amplification schemes and obstructions, along with some early ideas about capability amplification in general.

Experiment

Today, it is probably most worthwhile to study capability amplification when **A** is a *human's* policy.

In this setting, we are given some weak human policy **A**—say, a human thinking for an hour. We would like to amplify this to a strong collaborative policy $\mathbf{A}^+$, by invoking a bunch of copies of **A** and having them interact with each other appropriately.

In some sense this is the fully general problem of organizing human collaborations. But we can focus our attention on the most plausible obstructions for capability amplification, and try to design collaboration frameworks that let us overcome those obstructions.

In this context, I think the most interesting obstruction is working with concepts that are (slightly) too complicated for any individual copy of **A** to understand on its own. This looks like a hard problem that is mostly unaddressed by usual approaches to collaboration.

This post lays out a closely related problem—quickly evaluating arguments by experts—which gets at most of the same difficulties but may be easier to study. Superficially, evaluating arguments may seem easier than solving problems from scratch. But because it is so much easier to collaboratively *create* arguments once you have a way to evaluate them, I think the gap is probably only superficial.

Conclusion

The capability amplification problem may effectively isolate the central *philosophical* difficulties of value alignment. It's not easy to guess how hard it is—we may already have "good enough" solutions, or it may effectively be a restatement of the original problem.

Capability amplification asks us to implement a powerful policy that "behaves well," but it is easier than value alignment in two important respects: we are given access to a slightly weaker policy, and our implementation can be extremely inefficient. It may

be that these advantages are not significant advantages, but if so that would require us to significantly change our understanding of what the value alignment problem is about.

Capability amplification appears to be less tractable than the other research problems I've outlined. I think it's unlikely to be a good research direction for machine learning researchers interested in value alignment. But it may be a good topic for researchers with a philosophical focus who are especially interested in attacking problems that might otherwise be neglected.

(This research was supported as part of the Future of Life Institute FLI-RFP-AI1 program, grant #2015–143898.)

Appendix: iterating amplification

Let **H** be the input-output behavior of a human + all of the non-**A** tools at their disposal. Then an amplification procedure defines $\mathbf{A}^+$ as a simple computation that uses **H** and **A** as subroutines.

In particular, $\varnothing^+$ is a computation that uses **H** as a subroutine. If we amplify again, we obtain $\varnothing^{++}$, which is a computation that uses **H** and $\varnothing^+$ as subroutines. But since $\varnothing^+$ is a simple computation that uses **H** as a subroutine, we can rewrite $\varnothing^{++}$ as a simple computation that uses only **H** as a subroutine.

We can go on in this way, reaching $\varnothing^{+++}$, $\varnothing^{++++}$ and so on. By induction, all of these policies are defined by simple computations that use as **H** as a subroutine. (Of course these "simple computations" are exponentially expensive, even though they are easy to specify. But they have a simple form and can be easily written down in terms of the amplification procedure.)

Under some simple ergodicity assumptions, this sequence converges to a fixed point **Ω** (very similar to HCH). So a capability amplification procedure essentially uniquely defines an "optimal" policy **Ω**; this policy is uncomputable, but has a concise representation in terms of **H**.

If there is anything that **Ω** can't do, then we have found an unreachable policy. This perspective seems useful for identifying the hard part of the capability amplification problem.

Specifying an amplification strategy also specifies a way to set up an interaction between a bunch of copies of **H** such that they implement **Ω**. Indeed, designing such an interaction is *easier* than designing an amplification procedure that converges to **Ω**. So if we can't design a procedure for a bunch of copies of **H** to collaboratively accomplish some task T, then we also can't design an amplification procedure that can reach policies that can accomplish T.

But in some cases this seems extremely challenging. For example, suppose that we would like to write a very good textbook about general relativity. Recall that **H** is the policy implemented by a human over one day. To make the game hard, suppose that **H** has never encountered calculus.

The most natural approaches require at least one copy of **H** to understand general relativity. But **H** only lasts a single day, and there many not be *any* sequence of observations that makes **H** understand general relativity within a day.

So in order for **Ω** to succeed at this task, it needs to be able to write a textbook without *any* single copy of **H** understanding general relativity. The understanding must instead be distributed across a large number of copies of **H**. Yet they must somehow collaborate to write individual sentences that correctly communicate the theory.

This sounds like a hard problem.

Appendix: examples of capability amplification

ETA: my current best proposal is meta-execution, which is essentially a formalization of the "universal" scheme below.

Warmup I: Preprocessing

We instantiate a "leader" agent who uses the policy **A**. For each observation $o[k]$:

- We instantiate a new agent using **A**. We give them observation: "think about the observation $\{o[k]\}$ and output a useful hint for a copy of yourself who has to respond to it." Let $h[k]$ be the agent's action. Then we destroy the agent.
- We give the leader the observation "The observation is $\{o[k]\}$, and a copy of you prepared the hint $\{h[k]\}$." Let $a[k]$ be its action.

We might hope that this composite is smarter than **A**. After all, it has twice as long to think—even if the copy can't come up with a very helpful hint, it doesn't seem so hard to come up with a hint that is better than nothing.

Warmup II: Asking questions

We instantiate a "leader" agent who uses the policy **A**. We give each observation $o[k]$ to the leader, and ask them to reply with an action $a[k]$.

But we also allow the leader to instantiate subagents and communicate with those agents.

That is, if the leader takes an action of the special form "Instantiate a new agent" then we create a new agent using **A**, assign it a unique ID, and give the leader the observation: "A new agent has been created with ID [X]." Similarly, the leader can take the special action "Copy the agent with ID [X]."

If the leader takes an action of the special form "Ask [Q] to the agent with ID [X]" then we give Q as an observation to the agent with ID X. That agent will reply with some action A, and we give the leader the next observation: "The agent replied with [A]."

Eventually, the leader takes an action of the special form "Take action [A]". Then we set $a[k]$ = A, and move on to the next observation.

The scheme from the last section is essentially the special case where the leader instantiates a single subagent and asks the single question "What is a useful hint for

responding to observation *o*[*k*]?”

Warmup III: Recursion

We improve the scheme from the last section by allowing *every* agent to instantiate new subagents, rather than restricting this capability to the leader.

The messages between agents can contain not only words, but pointers to other agents. This allows the agents to pass arbitrarily large messages. For example, they can build a data structure out of agents, and just pass a “pointer” to the root of that data structure.

We can also slightly improve the scheme by allowing the leader to copy *itself*. This may be helpful if the leader has built up some useful context for the current situation.

Universal amplification

We improve the scheme further by giving the agents a large external memory. Rather than directly giving the leader the observation *o*[*k*], we write the observation to the external memory and tell the leader where the observation is written.

In addition to allowing agents to instantiate new agents and communicate with them, any agent can take the action “execute [P]”, where P is a program to be run against the external memory; the agent sees the observation “the program returned [*x*]” where *x* is P’s return value (or “nothing”).

Rather than having the leader directly return an action, the leader can take the special action “output the action returned by program [P].”

Messages between agents can now contain pointers to this external memory. For example, the leader could instantiate a subagent and ask it the question “Can you distinguish [*x*] from an array of random bytes?” where *x* is a pointer to an array in external memory.

We can easily generalize this setup to a parallel model of computation. We can also replace the shared memory by a more natural model for interprocess communication.

Appendix: knowledge about humans

Human values are complex. If you are only able to interact with a human for a day, it may be completely impossible to figure out what they value, no matter how smart you are. Understanding what someone values may require giving them a large amount of time to reflect on their values, doing neuroscience, or carrying out other processes that take longer than a day.

This may imply an obstruction to capability amplification—we can’t reach policies that have more knowledge about humans than can be acquired by interacting with **H.**

However, even if this is a real obstruction, it does not seem to be an important one, for the following reason.

Suppose that we are able to train a very good policy, which does not reflect any complex facts about human values-upon-reflection. This optimal policy still can reflect many basic facts about human preferences:

1. We don't want anything terrible to happen.
2. We want to "stay in control" of the agents we build.
3. We don't want our agent to get left behind by its competitors; it should fight as hard as it can to retain influence over the world, subject to #1 and #2.

Moreover, all of these concepts are relatively easy to understand even if you have minimal understanding of human values.

So an excellent agent with a minimal understanding of human values seems OK. Such an agent could avoid getting left behind by its competitors, and remain under human control. Eventually, once it got enough information to understand human values (say, by interacting with humans), it could help us implement our values.

In the worst case the agent would lack a nuanced understanding of what we consider terrible, and so would have to either be especially conservative or else risk doing terrible things in the short term. In the scheme of things, this is not a catastrophic problem.

Appendix: an example obstruction

Suppose that my brain encodes a random function f: $\{0, 1\}^* \to \{0, 1\}$ in the following sense: you can give me a sequence of bits, one per second, and then I can tell you the value of f on that sequence. There is no way to evaluate $f$other than to ask me.

Let N be the length of our capability amplification procedure, in seconds.

Let $\mathcal{L} \subseteq \mathcal{A}$ be the set of policies that can be implemented using an oracle for f, restricted to inputs of length N.

Then it's easy to see that $\mathcal{L}$ forms an obstruction:

- We can simulate access to any policy in $\mathcal{L}$ using an oracle for f restricted to inputs of length N. And we can simulate my role in the amplification procedure using an oracle for f restricted to inputs of length N. So policies in $\mathcal{L}$ can only be amplified to other policies in $\mathcal{L}$.
- We cannot evaluate f on even a single input of length N+1 using an oracle for f on inputs of length N. Most interesting classes $\mathcal{A}$ will contain some policies not in $\mathcal{L}$.

Whether this is a real obstruction depends on what the information is about:

- If it's just random bits, then we don't care at all—any other random bits would be "just as good."
- If the random function encodes important information about my values, then we are in the situation described in the previous section, which doesn't seem so bad.
- The worst case is when the function f encodes important information about how to behave effectively. For example, it encodes information about how to make accurate predictions. In this case we may actually be in trouble, since a policy that doesn't know f may be outcompeted by one which does.

This was originally posted here on 2nd October 2016.

The next post in this sequence will be 'Learning with catastrophes' by Paul Christiano.

Tomorrow's post will be 'Following Human Norms' in the sequence Value Learning by Rohin Shah.

Learning with catastrophes

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

A *catastrophe* is an event so bad that we are not willing to let it happen even a single time. For example, we would be unhappy if our self-driving car *ever* accelerates to 65 mph in a residential area and hits a pedestrian.

Catastrophes present a theoretical challenge for traditional machine learning—typically there is no way to reliably avoid catastrophic behavior without strong statistical assumptions.

In this post, I'll lay out a very general model for catastrophes in which they are avoidable under much weaker statistical assumptions. I think this framework applies to the most important kinds of catastrophe, and will be especially relevant to AI alignment.

Designing practical algorithms that work in this model is an open problem. In a subsequent post I describe what I currently see as the most promising angles of attack.

Modeling catastrophes

We consider an agent A interacting with the environment over a sequence of episodes. Each episode produces a transcript τ, consisting of the agent's observations and actions, along with a reward $r \in [0, 1]$. Our primary goal is to quickly learn an agent which receives high reward. (Supervised learning is the special case where each transcripts consist of a single input and a label for that input.)

While training, we assume that we have an oracle which can determine whether a transcript τ is "catastrophic." For example, we might show a transcript to a QA analyst and ask them if it looks catastrophic. This oracle can be applied to arbitrary sequences of observations and actions, including those that don't arise from an actual episode. So training can begin before the very first interaction with nature, using only calls to the oracle.

Intuitively, a transcript should only be marked catastrophic if it satisfies two conditions:

1. The agent made a catastrophically bad decision.
2. The agent's observations are plausible: we have a right to expect the agent to be able to handle those observations.

While actually interacting with the environment, the agent cannot query the oracle—there is no time to wait for a QA engineer to review a proposed action to check if it would be catastrophic.

Moreover, if interaction with nature ever produces a catastrophic transcript, we immediately fail. The performance of an algorithm is characterized by two parameters: the probability of catastrophic failure, and the total reward assuming no catastrophic failure.

We assume that there are some policies such that no matter what nature does, the resulting transcript is *never* catastrophic.

Traditionally in RL the goal is to get as much reward as the best policy from some class C. We' slightly weaken that goal, and instead aim to do as well as the best policy from C that never makes a catastrophic decision.

Batch learning

I've described an online version of learning with catastrophes. We can also consider the batch version, where the learner is first given a large number of "training" episodes.

In the batch version, there is no penalty for catastrophes at training time, and we don't care about training error. The two performance parameters are test-time performance and test-time catastrophe probability.

The oracle

This definition depends on an oracle who determines which transcripts are catastrophic.

For weak AI systems, the oracle may be a human. But a powerful AI system might take actions which are catastrophic but which look inoffensive to a human judge, so this approach doesn't cut it.

In general, the judge should be a human+AI team which is *more* competent than the system being trained, armed with an adequate solution to the informed oversight problem.

Approach

Learning with catastrophes is straightforward *given an unlimited number of queries to the catastrophe oracle*. Given any online learning algorithm A, we can "harden" it by running the following process before the beginning of each episode (a similar process can be applied to a batch learner):

1. Search over all possible environments, running A on each one to obtain a transcript.
2. If we find a catastrophic transcript τ, then add τ to A's training data with a reward of -1, and go back to step 1.

It's easy to prove that this process converges, if A is competent: the number of times we invoke step 2 is at most the time required to learn an optimal catastrophe-free policy (plus the number of episodes).

The big problem with this algorithm is the exponential search in step 1.

Adversarial training

The most obvious approach to *tractable* learning with catastrophes is to replace step 1 with a tractable algorithm that approximates the search. This process is an example of adversarial training.

We can formalize this as a simple game between the agent A and an adversary B. The two agents alternate taking moves, with A playing the role of the agent and B playing the role of the environment. B wins if the resulting transcript is catastrophic, A wins if it is not catastrophic.

Before each episode, we play this game N times, where $N > 0$ is a hyperparameter. If B is able to win *any* of the games, then we add the resulting transcript to A's training data and start again. Once B is not able to win any games, then we move on to a "real" episode. We hope that B is good enough at the game that if it can't produce a catastrophic transcript, then the real environment can't produce a catastrophic transcript either.

More precisely, before each episode we perform the following process:

1. Set $i = 0$.
2. A and B alternate taking moves, producing a transcript τ.
3. If τ is catastrophic, we add τ to A's training data with a reward of -1, and add τ to B's training data with a reward of $+1$. Then we go back to step 1.
4. If τ is not catastrophic, we add τ to B's training data with a reward of -1.
5. If $i < N$, we increment i and go back to step 2.

I discuss this idea in more detail in my post on red teams. There are serious problems with this approach and I don't think it can work on its own, but fortunately it seems combinable with other techniques.

Conclusion

Learning with catastrophes is a very general model of catastrophic failures which avoids being obviously impossible. I think that designing competent algorithms for learning with catastrophes may be an important ingredient in a successful approach to AI alignment.

This was originally posted here on 28th May, 2016.

Tomorrow's AI Alignment sequences post will be in the sequence on Value Learning by Rohin Shah.

The next post in this sequence will be 'Thoughts on Reward Engineering' by Paul Christiano, on Thursday.

Thoughts on reward engineering

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

*Note: This is the first post from **part five: possible approaches** of the sequence on iterated amplification. The fifth section of the sequence breaks down some of these problems further and describes some possible approaches.*

Suppose that I would like to train an RL agent to help me get what I want.

If my preferences could be represented by an easily-evaluated utility function, then I could just use my utility function as the agent's reward function. But in the real world that's not what human preferences look like.

So if we actually want to turn our preferences into a reward function suitable for training an RL agent, we have to do some work.

This post is about the straightforward parts of reward engineering. I'm going to deliberately ignore what seem to me to be the hardest parts of the problem. Getting the straightforward parts out of the way seems useful for talking more clearly about the hard parts (and you never know what questions may turn out to be surprisingly subtle).

The setting

To simplify things even further, for now I'll focus on the special case where our agent is taking a single action *a*. All of the difficulties that arise in the single-shot case also arise in the sequential case, but the sequential case also has its own set of additional complications that deserve their own post.

Throughout the post I will imagine myself in the position of an "overseer" who is trying to specify a reward function R(*a*) for an agent. You can imagine the overseer as the user themselves, or (more realistically) as a team of engineer and/or researchers who are implementing a reward function intended to expresses the user's preferences.

I'll often talk about the overseer computing R(*a*) themselves. This is at odds with the usual situation in RL, where the overseer implements a very fast function for computing R(*a*) in general ("1 for a win, 0 for a draw, -1 for a loss"). Computing R(*a*) for a particular action *a* is strictly easier than producing a fast general implementation, so in some sense this is just another simplification. I talk about why it might not be a crazy simplification in section 6.

Contents

1. **Long time horizons**. How do we train RL agents when we care about the long-term effects of their actions?
2. **Inconsistency and unreliability**. How do we handle the fact that we have only imperfect access to our preferences, and different querying strategies are not

guaranteed to yield consistent or unbiased answers?
3. **Normative uncertainty**. How do we train an agent to behave well in light of its uncertainty about our preferences?
4. **Widely varying reward**. How do we handle rewards that may vary over many orders of magnitude?
5. **Sparse reward**. What do we do when our preferences are very hard to satisfy, such that they don't provide any training signal?
6. **Complex reward**. What do we do when evaluating our preferences is substantially more expensive than running the agent?

- **Conclusion**.
- **Appendix: harder problems.**

1. Long time horizons

A single decision may have very long-term effects. For example, even if I only care about maximizing human happiness, I may instrumentally want my agent to help advance basic science that will one day improve cancer treatment.

In principle this could fall out of an RL task with "human happiness" as the reward, so we might think that neglecting long-term effects is just a shortcoming of the single-shot problem. But even in theory there is no way that an RL agent can learn to handle arbitrarily long-term dependencies (imagine training an RL agent to handle 40 year time horizons), and so focusing on the sequential RL problem doesn't address this issue.

I think that the only real approach is to choose a reward function that reflects the overseer's expectations about long-term consequences—i.e., the overseer's task involves both making predictions about what will happen, and value judgments about how good it will be. This makes the reward function more complex and in some sense limits the competence of the learner by the competence of the reward function, but it's not clear what other options we have.

Before computing the reward function R(a), we are free to execute the action a and observe its short-term consequences. Any data that could be used in our training process can just as well be provided as an input to the overseer, who can use the auxiliary input to help predict the long-term consequences of an action.

2. Inconsistency and unreliability

A human judge has no hope of making globally consistent judgments about which of two outcomes are preferred—the best we can hope for is for their judgments to be right in sufficiently obvious cases, and to be some kind of noisy proxy when things get complicated. Actually outputting a numerical reward—implementing some utility function for our preferences—is even more hopelessly difficult.

Another way of seeing the difficult is to suppose that the overseer's judgment is a noisy and potential biased evaluation of the quality of the underlying action. If both R(a) and R(a') are both big numbers with a lot of noise, but the two actions are actually quite similar, then the difference will be dominated by noise. Imagine an overseer trying to estimate the impact of drinking a cup of coffee on Alice's life by

estimating her happiness in a year conditioned on drinking the coffee, estimating happiness conditioned on not drinking the coffee, and then subtracting the estimates.

We can partially address this difficulty by allowing the overseer to make *comparisons* instead of assessing absolute value. That is, rather than directly implementing a reward function, we can allow the overseer to implement an antisymmetric comparison function C(*a, a′*): which of two actions *a* and *a′* is a better in context? This function can take real values specifying *how much*one action is better than another, and should by antisymmetric.

In the noisy-judgments model, we are hoping that the noise or bias of a comparison C(*a*, *a′*) depends on the actual magnitude of the difference between the actions, rather than on the absolute quality of each action. This hopefully means that the total error/bias does not drown out the actual signal.

We can then define the decision problem as a zero-sum game: two agents propose different actions *a* and *a′*, and receive rewards C(*a*, *a′*) and C(*a′*, *a*). At the equilibrium of this game, we can at least rest assured that the agent doesn't do anything that is *unambiguously worse* than another option it could think of. In general, this seems to give us sensible guarantees when the overseer's preferences are not completely consistent.

One subtlety is that in order to evaluate the comparison C(*a*, *a′*), we may want to observe the short-term consequences of taking action *a* or action *a′*. But in many environments it will only be possible to take one action. So after looking at both actions we will need to choose at most one to actually execute (e.g. we need to estimate how good drinking coffee was, after observing the short-term consequences of drinking coffee but without observing the short-term consequences of not drinking coffee). This will generally increase the variance of C, since we will need to use our best guess about the action which we didn't actually execute. But of course this is a source of variance that RL algorithms already need to contend with.

3. Normative uncertainty

The agent is uncertain not only about its environment, but also about the overseer (and hence the reward function). We need to somehow specify how the agent should behave in light of this uncertainty. Structurally, this is identical to the philosophical problem of managing normative uncertainty.

One approach is to pick a fixed yardstick to measure with. For example, our yardstick could be "adding a dollar to the user's bank account." We can then measure C(*a*, *a′*) as a multiple of this yardstick: "how many dollars would we have to add to the user's bank account to make them indifferent between taking action *a* and action *a′*?" If the user has diminishing returns to money, it would be a bit more precise to ask: "what chance of replacing *a* with *a′* is worth adding a dollar to the user's bank account?" The comparison C(*a*, *a′*) is then the inverse of this probability.

This is exactly analogous to the usual construction of a utility function. In the case of utility functions, our choice of yardstick is totally unimportant—different possible utility functions differ by a scalar, and so give rise to the same preferences. In the case of normative uncertainty that is no longer the case, because we are specifying how to *aggregate* the preferences of different possible versions of the overseer.

I think it's important to be aware that different choices of yardstick result in different behavior. But hopefully this isn't an *important* difference, and we can get sensible behavior for a wide range of possible choices of yardstick—if we find a situation where different yardsticks give very different behaviors, then we need to think carefully about how we are applying RL.

For many yardsticks it is possible to run into pathological situations. For example, suppose that the overseer might decide that dollars are worthless. They would then radically increase the value of all of the agent's decisions, measured in dollars. So an agent deciding what to do would effectively care much more about worlds where the overseer decided that dollars are worthless.

So it seems best to choose a yardstick whose value is relatively stable across possible worlds. To this effect we could use a broader basket of goods, like 1 minute of the user's time + 0.1% of the day's income + *etc.* It may be best for the overseer to use common sense about how important a decision is relative to some kind of idealized influence in the world, rather than sticking to any precisely defined basket.

It is also desirable to use a yardstick which is simple, and preferably which minimizes the overseer's uncertainty. Ideally by standardizing on a single yardstick throughout an entire project, we could end up with definitions that are very broad and robust, while being very well-understood by the overseer.

Note that if the same agent is being trained to work for many users, then this yardstick is also specifying how the agent will weigh the interests of different users—for example, whose accents will it prefer to spend modeling capacity on understanding? This is something to be mindful of in cases where it matters, and it can provide intuitions about how to handle the normative uncertainty case as well. I feel that economic reasoning is useful for arriving at sensible conclusions in these situations, but there are other reasonable perspectives.

4. Widely varying reward

Some tasks may have widely varying rewards—sometimes the user would only pay 1¢ to move the decision one way or the other, and sometimes they would pay $10,000.

If small-stakes and large-stakes decisions occur comparably frequently, then we can essentially ignore the small-stakes decisions. That will happen automatically with a traditional optimization algorithm—after we normalize the rewards so that the "big" rewards don't totally destroy our model, the "small" rewards will be so small that they have no effect.

Things get more tricky when small-stakes decisions are much more common than the large-stakes decisions. For example, if the importance of decisions is power-law distributed with an exponent of 1, then decisions of all scales are in some sense equally important, and a good algorithm needs to do well on all of them. This may sound like a very special case, but I think it is actually quite natural for there to be several scales that are all comparably important *in total*.

In these cases, I think we should do importance sampling—we oversample the high-stakes decisions during training, and scale the rewards down by the same amount, so that the contribution to the total reward is correct. This ensures that the scale of

rewards is basically the same across all episodes, and lets us apply a traditional optimization algorithm.

Further problems arise when there are some *very* high-stakes situations that occur very rarely. In some sense this just means the learning problem is actually very hard—we are going to have to learn from few samples. Treating different scales as the same problem (using importance sampling) may help if there is substantial transfer between different scales, but it can't address the whole problem.

For very rare+high-stakes decisions it is especially likely that we will want to use simulations to avoid making any obvious mistakes or missing any obvious opportunities. Learning with catastrophes is an instantiation of this setting, where the high-stakes settings have only downside and no upside. I don't think we really know how to cope with rare high-stakes decisions; there are likely to be some fundamental limits on how well we can do, but I expect we'll be able to improve a lot over the current state of the art.

5. Sparse reward

In many problems, "almost all" possible actions are equally terrible. For example, if I want my agent to write an email, almost all possible strings are just going to be nonsense.

One approach to this problem is to adjust the reward function to make it easier to satisfy—to provide a "trail of breadcrumbs" leading to high reward behaviors. I think this basic idea is important, but that changing the reward function isn't the right way to implement it (at least conceptually).

Instead we could treat the problem statement as given, but view auxiliary reward functions as a kind of "hint" that we might provide to help the algorithm figure out what to do. Early in the optimization we might mostly optimize this hint, but as optimization proceeds we should anneal towards the actual reward function.

Typical examples of proxy reward functions include "partial credit" for behaviors that look promising; artificially high discount rates and careful reward shaping; and adjusting rewards so that small victories have an effect on learning even though they don't actually matter. All of these play a central role in practical RL.

A proxy reward function is just one of many possible hints. Providing demonstrations of successful behavior is another important kind of hint. Again, I don't think that this should be taken as a change to the reward function, but rather as side information to help achieve high reward. In the long run, we will hopefully design learning algorithms that automatically learn how to use general auxiliary information.

6. Complex reward

A reward function that intends to capture all of our preferences may need to be very complicated. If a reward function is implicitly estimating the expected consequences of an action, then it needs to be even more complicated. And for powerful learners, I expect that reward functions will need to be learned rather than implemented directly.

It is tempting to substitute a simple proxy for a complicated real reward function. This may be important for getting the optimization to work, but it is problematic to change the definition of the problem.

Instead, I hope that it will be possible to provide these simple proxies as hints to the learner, and then to use semi-supervised RL to optimize the real hard-to-compute reward function. This may allow us to perform optimization even when the reward function is many times more expensive to evaluate than the agent itself; for example, it might allow a human overseer to compute the rewards for a fast RL agent on a case by case basis, rather than being forced to design a fast-to-compute proxy.

Even if we are willing to spend much longer computing the reward function than the agent itself, we still won't be able to find a reward function that perfectly captures our preferences. But it may be just as good to choose a reward function that captures our preferences "for all that the agent can tell," i.e. such that the conditioned on two outcomes receiving the same expected reward the agent cannot predict which of them we would prefer. This seems much more realistic, once we are willing to have a reward function with much higher computational complexity than the agent.

Conclusion

In reinforcement learning we often take the reward function as given. In real life, we are only given our preferences—in an implicit, hard-to-access form—and need to engineer a reward function that will lead to good behavior. This presents a bunch of problems. In this post I discussed six problems which I think are relatively straightforward. (Straightforward from the reward-engineering perspective—the associated RL tasks may be very hard!)

Understanding these straightforward problems is important if we want to think clearly about very powerful RL agents. But I expect that most of our time will go into thinking about harder problems, for which we don't yet have any workable approach. These harder problems may expose more fundamental limits of RL, that will require substantially new techniques to address.

Appendix: harder problems

Informed oversight

The process that produces a decision may encode important information about the quality of that decision, and if we throw out that information then a very bad decision may nevertheless receive a high reward. For example, if we want an agent to produce original art, we may not be able to evaluate a painting without understanding the process that produced it (and, in particular, what other artwork it was inspired by).

In some cases, recovering information about the process seems feasible. For example, if the agent is using attention or an external memory, the overseer can check what art the agent looked at. But in other cases it seems much more challenging: if the agent's weights have memorized properties of particular artwork, we would need new techniques in order to make that information available to the overseer.

It is not clear whether there is any general way to give the overseer enough insight into the agent's thought process to avoid problems like this; there may be some inherent limitations on delegation. Either way, it would be nice to understand the situation much better.

Capability amplification

By folding our preferences and long-term predictions into the reward function, we are in some sense limiting the agent to be no stronger than the overseer. If we start with a human, then RL will only ever produce human-level reasoning about long-term consequences or about "what is good."

So we need some other mechanism for producing better long-term reasoning. I think that the most promising approach is to treat "reasoning well" as a short-term problem, for which humans can provide adequate oversight, and then to run that reasoning process for longer in order to produce more powerful reasoning.

Reward hacking / security

If our reward function has any problems, a powerful RL agent may tend to find actions *a* that exploit those problems in order to receive a high reward. In order to get good consequences, our reward function needs to be sufficiently secure that the agent can't attack it in this way.

The first line of defense against an attack is to avoid immediately executing the agent's action. Instead, the overseer can inspect that action and try to ensure that it isn't intended as an attack. This leaves attacks which the overseer cannot recognize as attacks, or which do damage even when the overseer looks at them.

If the techniques from the previous sections actually allow the overseer to evaluate the agent's actions, then they can also probably allow the overseer can detect attacks. Security during evaluation itself is an additional question though.

The main cause for hope is if the overseer can (1) be smarter than the agent which is trying to attack it, and (2) have access to some information about the agent's thought process. Hopefully (2) allows the overseer to overcome the disadvantage of the "position of the interior"—if the agent picks a particular attack vector, the overseer can "watch them thinking" and then devote its energies to trying to detect or defend against that particular attack.

Techniques for optimizing worst-case performance

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

If powerful ML systems fail catastrophically, they may be able to quickly cause irreversible damage. To be safe, it's not enough to have an average-case performance guarantee on the training distribution—we need to ensure that even if our systems fail on new distributions or with small probability, they will never fail *too* badly.

The difficulty of optimizing worst-case performance is one of the most likely reasons that I think prosaic AI alignment might turn out to be impossible (if combined with an unlucky empirical situation).

In this post I want to explain my view of the problem and enumerate some possible angles of attack. My goal is to communicate why I have hope that worst-case guarantees are achievable.

None of these are novel proposals. The intention of this post is to explain my view, not to make a new contribution. I don't currently work in any of these areas, and so this post should be understood as an outsider looking in, rather than coming from the trenches.

Malign vs. benign failures and corrigibility

I want to distinguish two kinds of failures:

- "Benign" failures, where our system encounters a novel situation, doesn't know how to handle it, and so performs poorly. The resulting behavior may simply be erratic, or may serve an external attacker. Their effect is similar to physical or cybersecurity vulnerabilities—they create an opportunity for destructive conflict but don't systematically disfavor human values. They may pose an existential risk when combined with high-stakes situations, in the same way that human incompetence may pose an existential risk. Although these failures are important, I don't think it is necessary or possible to eliminate them in the worst case.
- "Malign" failures, where our system continues to behave competently but applies its intelligence in the service of an unintended goal. These failures systematically favor whatever goals AI systems tend to pursue in failure scenarios, at the expense of human values. They constitute an existential risk independent of any other destructive technology or dangerous situation. Fortunately, they seem both less likely and potentially possible to avoid even in the worst case.

I'm most interested in malign failures, and the narrower focus is important to my optimism.

The distinction between malign and benign failures is not always crisp. For example, suppose we try to predict a human's preferences, then search over all strategies to

find the one that best satisfies the predicted preferences. Guessing the preferences even a little bit wrong would create an adversarial optimizer incentivized to apply its intelligence to a purpose at odds with our real preferences. If we take this approach, incompetence does systematically disfavor human values.

By aiming for corrigible rather than optimal behavior (see here or here) I'm optimistic that it is possible to create a sharper distinction between benign and malign failures, which can be leveraged by the techniques below. But for now, this hope is highly speculative.

Amplification

I believe that these techniques are much more likely to work if we have access to an overseer who is significantly smarter than the model that we are trying to train. I hope that amplification makes this possible.

It seems realistic for a strong overseer to recognize an (input, output) pair as a malign failure mode (though it may require a solution to informed oversight). So now we have a concrete goal: find a model that never gives an output the overseer would diagnose as catastrophically bad.

Historically researchers in the AI safety community have been extremely pessimistic about reliability. I think part of that pessimism is because they have been imagining working with models much smarter than the overseer.

Techniques

I'll describe three categories of techniques:

- Adversarial training
- Verification
- Transparency

Previous versions of this list included implicit ensembles, e.g. Bayesian neural nets. I still think there might be useful techniques in that space, but I've become pessimistic enough that it no longer makes the list.

Adversarial training

(*Examples: Explaining and Harnessing Adversarial Examples, Adversarial Machine Learning at Scale.)*

If we want to know whether our model ever fails catastrophically, it's natural to train an adversary to search for catastrophe-inducing inputs. If the adversary can't find one, then we hope that none exist. In order to produce a catastrophe-free model, we can integrate the results of testing into the training process.

The basic problem with this strategy is that no matter how hard you test, it's always possible (realistically, it's likely) that there are possible failure cases your adversary didn't notice. For example, suppose that it takes 2^N seconds to find the Nth failure.

Because nature will be many times more complex than our training procedure, it's inevitable that there are many types of failures that nature can find but we can't.

We can attack this problem by trying to make the adversary's job easier. Some possibilities:

- Use interpretability techniques (see below) to help the adversary identify possible failures.
- Use interpretability to expand what counts as catastrophic—if the adversary can get the model to think in a way that looks like it *could* lead to catastrophic behavior under different situations, that could count as a success for the adversary.
- Combine with verification (see below) to help guide the adversary's search and expand the space of allowed attacks.
- Allow the adversary to directly control some of the internal computation of our model rather than merely controlling inputs. This requires a way to constrain the actions of the adversary—we want them to be able to produce any failure that could arise naturally, but if we give them too much power then they can trivially cause a catastrophic failure.
- Change the system's inputs so they are easier for the adversary to fake: see the "distorting observations" and "abstracting observations" sections of this post.
- Make the adversary more powerful, without changing their task, by applying amplification.

Adversarial training faces other difficulties as well. For example, it might take a long time for adversarial training to converge, or training might never converge due to non-convexity. I think these are significantly less likely to be deal-breakers, but they are certainly important research problems if we actually want to apply adversarial training.

Verification

(Examples: Certified Defenses against Adversarial Examples, Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks)

In verification the goal is to find a certificate that there is no input on which a system behaves badly.

Verification has (at least) two problems:

- **Complexity**. It's really, really hard to verify properties of a computationally interesting model.
- **Specification**. It's unclear what we should be trying to verify.

Handling complexity is definitely challenging. But if we are free to train the model in order to facilitate verification, and if we are only interested in certifying some "easy" property that the model satisfies with slack, then it's not clearly doomed.

Prima facie, specification looks more like an unfixable deal-breaker. In the rest of this section I'll give three possible approaches for finding specifications. I think none of these is satisfactory on their own, but they leave me optimistic that verification can be useful despite specification difficulty.

My current view is that specification and complexity are both potentially surmountable difficulties, but that it's likely their resolutions will result in partial solutions that need to be combined with other approaches.

Use a slow model of overseer as a specification. If our overseer is a physical process, to actually get a specification we'd need to learn a model of the overseer. So we've just shifted the problem to the model of the overseer.

Fortunately, this move does potentially made the problem one step better. Because we don't need to use the overseer model at runtime, we can afford to use a much slower model. So verification could distill a very slow reliable model into a fast reliable model.

Amplification. If our overseer is produced by amplifying some process H, and we have a trusted model of H, then we can obtain a trusted model of the overseer. So if we were able to form a trusted model of the first step of amplification, then we could iteratively use verification to construct trusted models at each subsequent step.

We are still left with the problem at the first step of amplification. But at this step we might be able to restrict attention to a small range of questions for which worst-case guarantees are more feasible; we may be able to train weaker models for which worst-case guarantees are more feasible; or we might be able to eliminate the human altogether and begin amplification from some explicit simple core of reasoning. Any of these seems plausible.

"Verification" without specification. I find it helpful to think of verification from the primal/dual perspective, which also makes the connection to adversarial training clearer.

In adversarial training, the adversary wants to find an input on which the system behaves poorly. The dual to the adversary is an explicit list of all inputs, showing that each one of them performs well. The dual is exponentially large, and (relatedly) the adversary's problem is exponentially hard.

We can view verification in a given proof system as making the adversary's job easier, by expanding the set of allowable "attacks." If we do this carefully, we can make the dual certificate much smaller and make the adversary's task much more tractable.

For example, if our proofs (dual solutions) are in first order logic, the adversary's task is finding any model of the axioms in which there exists an attack. (This is only different in the infinite case.) If our proofs are sum of squares proofs, the adversary's task is finding a pseudodistribution over attacks. For certain linear programming proofs, the adversary's goal is finding an attack on a particular extrapolation of the model. And so on.

On this perspective, we don't necessarily need an actual specification. Instead, we can take some external catastrophe-checker, relax it to a catastrophe checker over the adversary's relaxed action space, and then try to implement an adversary against that catastrophe checker. This would change the meaning of the dual, which would now have a subtle statistical interpretation rather than being a traditional certificate.

Transparency

(*Examples: Feature Visualization, Deep Inside Convolutional Networks*)

While training a model, we are free to look inside and “see what it’s thinking.” If we could actually understand those thoughts, we might be able to more easily predict under what conditions it would fail malignantly.

A (weird) example: suppose that human evolution was part of an alien project to train a mathematician, and some humans are extracted from our simulation and trained to solve alien math problems. If aliens are only able to test the humans’ behavior they might conclude that they always faithfully solve whatever problem they are posed—especially after a few generations of selection removed all the humans who didn’t do so. But if the aliens were able to look inside the humans’ heads they might discover that this behavior is extremely contingent—perhaps humans are only solving problems because they believe they will die otherwise, and the humans would behave very differently as soon as they saw an opportunity to escape.

That example is exotic and somewhat implausible, but I think that the same basic dynamic is likely to apply in much messier practical situations. If a model would fail malignantly under some conditions, I think there is some sense in which the model must be testing those conditions during training.

Malign vs. benign failures: I think the prospects for using transparency to detect malign failures in models trained by gradient descent are *much* better than the prospects for detecting benign failures, or for detecting failures in arbitrary models.

The key point is that a malign failure requires leveraging the intelligence of the model to do something actively bad. If our model is trained by gradient descent, its behavior can only be intelligent when it is exercised on the training distribution—if part of the model never (or very rarely) does anything on the training distribution, then that part of the model can’t be intelligent. So in some sense a malign failure mode needs to use a code path that gets run on the training distribution, just under different conditions that cause it to behave badly.

That means that we don’t necessarily need to analyze our model in the abstract, we can look at its behavior on actual training data. This looks much more likely to be tractable. (Though this entire discussion is extremely subtle, and without being more precise it’s very easy for us to go wrong.)

Amplification: Transparency will be hard if we ultimately want to train models much smarter than we are, but deeply understanding what a model is thinking probably requires being much smarter than it. I think that problem can potentially be overcome by using amplification to stay ahead of the models we are training. This means that current work on transparency, which considers models doing tasks that are within the abilities of the humans who are trying to understand them, could potentially remain relevant even as AI improves significantly.

What do you do with transparency? Merely understanding that a model might behave catastrophically could be useful, but it would be much nicer to actually fix the problem. Adversarial training gives a natural mechanism: once we understand a failure we can synthesize appropriate data and then train on that data.

This approach puts significantly more stress on our transparency techniques. Even if were initially able to use transparency to see how our model might fail, after we perform many generations of selection we might weed out exactly the comprehensible failures and leave the incomprehensible ones. You would only want to apply this technique if you had a great deal of faith in your methods; if you were feeling at all shaky about your ability to achieve worst-case guarantees, and

transparency techniques let you see one potential catastrophic failure, it would be better to consider that a near-miss and seriously rework your project rather than plowing on.

Conclusion

Making ML systems work in the worst case is hard, even if we are only concerned with malign failures and have access to an overseer who can identify them. If we can't solve this problem, I think it seriously calls into question the feasibility of aligned ML.

Fortunately there are at least a few plausible angles of attack on this problem. All of these approaches feel very difficult, but I don't think we've run into convincing deal-breakers. I also think these approaches are complementary, which makes it feel even more plausible that they (or their descendants) will eventually be successful. I think that exploring these angles of attack, and identifying new approaches, should be a priority for researchers interested in alignment.

This was originally posted here on 1st February, 2018.

The next post in this sequence is "Reliability Amplification", and will come out on Tuesday.

Reliability amplification

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

In a recent post I talked about capability amplification, a putative procedure that turns a large number of fast weak agents into a slower, stronger agent.

If we do this in a naive way, it will decrease reliability. For example, if…

- Our weak policy fails with probability 1%.
- In order to implement a strong policy we combine 10 decisions made by weak agents.
- If any of these 10 decisions is bad, then so is the combination.

…then the combination will be bad with 10% probability.

Although the combination can be more powerful than any individual decision, in this case it is much *less* reliable. If we repeat policy amplification several times, our failure probability could quickly approach 1, even if it started out being exponentially small.

Complementary to capability amplification is *reliability amplification*: given a policy H that usually works, can we implement a policy H that works significantly more reliably?

To be slightly less imprecise (but still quite crude):

- Given a distribution **A** over policies that ε-close to a benign policy for some $\varepsilon \ll 1$, can we implement a distribution $\mathbf{A}^+$ over policies which is δ-close to a benign policy of similar capability, for some $\delta \ll \varepsilon$?

When I say that **A** is ε-close to a benign policy, I mean that there is a benign distribution over policies which has total variation distance at most ε from **A**. We can view this is as a simple mixture between a single aligned policy that is chosen with probability $(1-\varepsilon)$ and an adversarial policy that is chosen with probability ε.

I will move freely between talking about "policies" and "distributions over policies;" we can turn a distribution into a single policy by averaging.

The notion of "implement" here is exactly the same as for capability amplification: we can use any number of implementations of **A** and whatever other resources are at our disposal**,** and must compute $\mathbf{A}^+$'s responses to a sequence of observations, within some time limit such as a day.

Reliability amplification via deliberation

I think there is one very natural way to do reliability amplification—if some variant of this idea doesn't work, I'm not sure what else to try.

As a warm-up, let's consider the setting where the agents want to make a single decision rather than a sequence of decisions—we'll return to the sequential case soon.

We sample three agents **A**, **B**, and **C** from the given distribution. We ask them to talk for a while and then each propose an action. Then we ask them to do approval voting amongst the three options, and randomly pick from the maximally-approved-of proposals.

If all three agents are aligned, then this proposal is clearly going to produce an aligned action. If two the agents behave adversarially then we are clearly out of luck, but that happens only with probability $O(\varepsilon^2)$, so we can basically ignore it.

The interesting case is when there is one adversarial agent and two aligned agents. If we threw up our hands and picked randomly in this case, then we would have a failure probability of ε. So the question is: can we do better than that?

The obvious hope is that aligned agents can do better-than-chance at recognizing each other / at recognizing good actions. If they can get any non-negligible advantage, then we can get a non-negligible reduction in error probability (and by iterating the procedure we can potentially get large reductions).

The sequential case

So far we have talked about making a single decision. The problem is more challenging when we need to make a sequence of decisions.

We can try to simply generalize the voting approach, running a new vote for each action. To see the difficulty, suppose that the optimal policy looks as follows:

1. Generate some hard-to-transfer internal state X. Think of X as something like an intuitive vision for how to approach the situation.
2. Use X to generate a first action.
3. After making an observation, use X to generate a second action. It is important that the same state X be used in steps [2] and [3].

Now suppose that in the first step the three agents all follow this protocol correctly. They arrive at three different proposed actions, which are backed up by three different visions. Now we have to pick one of these actions, effectively committing ourselves to a single agent. This destroys our ability to ensemble effectively in future steps: if the chosen agent fails then we are out of luck, since the other two agents don't have the vision needed to follow-through on that action.

To get around this problem we need to count on a stronger property: whatever kind of private state X an agent could generate, the agents need to be able to jointly develop a "consensus" private state which serves the same function. Then if one agent fails, the other agents can also use the consensus private state in order to continue acting intelligently.

(It would also be OK if only pairs of agents could establish such a consensus state, as long as the third agent was able to arbitrate between the two in the event of a future failure.)

Implementation

As of commit 1cbaae, this approval-voting scheme is part of my ALBA implementation on GitHub.

It is implemented in amplify.reliability.amplify_reliability, and is applied as part of amplify.amplify directly prior to applying HCH.

As a result I've moved one issue from the "FIXME: will fail" column to the "TODO: needs improvement" column.

Three differences with redundant computation

There is an easy recipe for fault-tolerant computation: compute the result 3 times, and take the majority vote. This increases the computational cost by a factor of 3, but decreases the error rate from ε to $O(\varepsilon^2)$.

Our reliability amplification scheme is considerably more subtle, and if we were to formalize the reliability amplification problem as stated we would find it impossible. This section touches on the most important additional subtleties.

Building consensus

Redundant computation normally relies on the existence of a unique correct answer on which all uncorrupted computations will agree. In our setting, there may be a range of "correct" behaviors. This introduces the additional challenge of being able to agree on a particular correct answer (and to build a consensus version of the hidden state underlying that answer).

Without some assumption of this form, reliability amplification is impossible. For example, suppose that there are 99 possible good answers and 1 bad answer. Our distribution **A** picks at random amongst the 100 possible answers. If the copies of **A** with good answers are completely unable to agree with each other, then there may simply be no way to tell the good answers from the bad answer. In this case, we can never reduce the error probability below 1%.

Balanced abilities

In order for our scheme to work we need to assume that the abilities of all of the agents are roughly balanced. I think this is probably necessary for any scheme to work. If you have an ε chance of a malicious superintelligence and a $(1-\varepsilon)$ chance of a friendly human, you are probably out of luck—in a deliberative process involving two humans and a superintelligence, the superintelligence is likely to have disproportionately much influence over the outcome even if it is wrong.

Our particular scheme fails outright in the context of powerful adversaries. In general there is a 3ε chance that *at least one* of the agents is unaligned. We may be able to prevent the unaligned agent from causing the group to adopt a really bad answer. But

even in the best case, the unaligned agent will probably be able to tilt the balance between the two options proposed by the aligned agents, allowing them to increase the probability of failure. If the unaligned agent is no smarter than the aligned agents, then the aligned agents can counteract this effect by preferentially picking the better action. But if the unaligned agent is much smarter than the aligned agents, then it can increase the probability of failures which are too subtle for the unaligned agents to anticipate, and the aligned agents have no way to decrease the probability of these failures.

Less time to think

Traditionally redundant computation increases costs by a constant factor. But in the context of reliability amplification, we also make life harder for each agent: in addition to picking an action, they need to spend time arguing about actions and critiquing each others' choices. This gives each agent less time to think about their action. It is not clear whether we can offset this effect

Relation to capability amplification and ALBA

Necessity

Reliability amplification seems completely necessary if we want to do something like ALBA using imitation or imitation+RL, since otherwise reliability will fall with each iteration of capability amplification. Of course we could have a "robust" capability amplification procedure which does not decrease reliability. But designing such a procedure is strictly harder than reliability amplification. So I think it makes a lot of sense to split up the problem into two pieces.

If working with approval-direction and pure RL, there is actually a form of reliability amplification already baked in: if the overseer fails with probability 1%, then this only changes the reward function by 0.01, and an RL agent should still avoid highly undesirable actions. That said, capability amplification may still be necessary in a pure RL setup if we can't solve the RL problem to arbitrary precision. In that case we may always have some non-negligible probability of making a bad decision, and after capability amplification this probability could become too large.

Balancing capability/reliability

Reliability amplification decreases our agent's capability but increases its reliability. Capability amplification increases capability and decreases reliability.

The hope is that we can somehow put these pieces together in a way that ends up increasing both reliability and capability.

If our reliability amplification step achieves a superlinear reduction in error probability from ε to $o(\varepsilon)$, and our capability amplification causes a linear increase from ε to $\Theta(\varepsilon)$, then this seems almost guaranteed to work.

To see this, consider the capability decrease from reliability amplification. We know that for large enough N, N iterations of capability amplification will more than offset this capability decrease. This N is a constant which is independent of the initial error rate ε, and hence the total effect of N iterations is to increase the error rate to $\Theta(\varepsilon)$. For sufficiently small ε, this is more than offset by the $\varepsilon \to o(\varepsilon)$ reliability improvement from reliability amplification. So for sufficiently small ε we can increase both reliability and capability.

A reduction from ε to $O(\varepsilon^2)$ is basically the "best case" for reliability amplification, corresponding to the situation where two aligned agents can always reach correct consensus. In general, aligned agents will have some imperfect ability to reach consensus and to correctly detect bad proposals from a malicious agent. In this setting, we are more likely to have an $\varepsilon \to O(\varepsilon)$ reduction. Hopefully the constant can be very good.

There are also lower bounds on the achievable reliability ε derived from the reliability of the human and of our learning procedures.

So in fact reliability amplification will increase reliability by some factor R and decrease capability by some increment Δ, while capability amplification decreases reliability by some factor R' and increases capability by some increment Δ′. Our hope is that there exists some capability amplification procedure with $\Delta'/\log(R') > \Delta/\log(R)$, and which is efficient enough to be used as a reward function for semi-supervised RL.

I think that this condition is quite plausible but definitely not a sure thing; I'll say more about this question in future posts.

Conclusion

A large computation is almost guaranteed to experience some errors. This poses no challenge for the theory of computing because those errors can be corrected: by computing redundantly we can achieve arbitrarily low error rates, and so we can assume that even arbitrarily large computations are essentially perfectly reliable.

A long deliberative process is similarly guaranteed to experience periodic errors. Hopefully, it is possible to use a similar kind of redundancy in order to correct these errors. This question is substantially more subtle in this case: we can still use a majority vote, but here the space of options is very large and so we need the additional step of having the correct computations negotiate a consensus.

If this kind of reliability amplification can work, then I think that capability amplification is a plausible strategy for aligned learning. If reliability amplification doesn't work well, then cascading failures could well be a fatal problem for attempts to define a powerful aligned agent as a composition of weak aligned agents.

This was originally posted here on 20th October, 2016.

The next post in this sequence will be 'Security Amplification' by Paul Christiano, on Saturday 2nd Feb.

Security amplification

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

An apparently aligned AI system may nevertheless behave badly with small probability or on rare "bad" inputs. The reliability amplification problem is to reduce the failure probability of an aligned AI. The analogous **security amplification problem** is to reduce the prevalence of bad inputs on which the failure probability is unacceptably high.

We could measure the prevalence of bad inputs by looking at the probability that a random input is bad, but I think it is more meaningful to look at the *difficulty of finding a bad input*. If it is exponentially difficult to find a bad input, then in practice we won't encounter any.

If we could transform a policy in a way that multiplicatively increase the difficulty of finding a bad input, then by interleaving that process with a distillation step like imitation or RL we could potentially train policies which are as secure as the learning algorithms themselves—eliminating any vulnerabilities introduced by the starting policy.

For sophisticated AI systems, I currently believe that meta-execution is a plausible approach to security amplification. (ETA: I still think that this basic approach to security amplification is plausible, but it's now clear that meta-execution on its own can't work.)

Motivation

There are many inputs on which any particular implementation of "human judgment" will behave surprisingly badly, whether because of trickery, threats, bugs in the UI used to elicit the judgment, snow-crash-style weirdness, or whatever else. (The experience of computer security suggests that complicated systems typically have *many* vulnerabilities, both on the human side and the machine side.) If we aggressively optimize something to earn high approval from a human, it seems likely that we will zoom in on the unreasonable part of the space and get an unintended result.

What's worse, this flaw seems to be inherited by any agent trained to imitate human behavior or optimize human approval. For example, inputs which cause humans to behave badly would also cause a competent human-imitator to behave badly.

The point of security amplification is to remove these human-generated vulnerabilities. We can start with a human, use them to train a learning system (that inherits the human vulnerabilities), use security amplification to reduce these vulnerabilities, use the result to train a new learning system (that inherits the reduced set of vulnerabilities), apply security amplification to reduce those vulnerabilities further, and so on. The agents do not necessarily get more powerful over the course of this process—we are just winnowing away the idiosyncratic human vulnerabilities.

This is important, if possible, because it (1) lets us train more secure systems, which is good in itself, and (2) allows us to use weak aligned agents as reward functions for a extensive search. I think that for now this is one of the most plausible paths to capturing the benefits of extensive search without compromising alignment.

Security amplification would not be directly usable as a substitute for informed oversight, or to protect an overseer from the agent it is training, because informed oversight is needed for the distillation step which allows us to iterate security amplification without exponentially increasing costs.

Note that security amplification + distillation will only remove the vulnerabilities that came from the human. We will still be left with vulnerabilities introduced by our learning process, and with any inherent limits on our model's ability to represent/learn a secure policy. So we'll have to deal with those problems separately.

Towards a definition

The security amplification problem is to take as given an implementation of a policy **A**, and to use it (along with whatever other tools are available) to implement a significantly more secure policy $\mathbf{A}^+$.

Some clarifications:

- "implement:" This has the same meaning as in capability amplification or reliability amplification. We are given an implementation of **A** that runs in a second, and we have to implement $\mathbf{A}^+$ over the course of a day.
- "secure": We can measure the security of a policy **A** as the difficulty of finding an input on which **A** behaves badly. "Behaves badly" is slippery and in reality we may want to use a domain-specific definition, but intuitively it means something like "fails to do even roughly what we want."
- "more secure:" Given that difficulty (and hence security) is not a scalar, "more secure" is ambiguous in the same way that "more capable" is ambiguous. In the case of capability amplification, we need to show that we could amplify capability in *every* direction. Here we just need to show that there is *some* notion of difficulty which is significantly increased by capability amplification.
- "significantly more secure": We would like to reach very high degrees of security after a realistic number of steps. This requires an exponential increase in difficulty, i.e. for each step to multiplicatively increase the difficulty of an attack. This is a bit subtle given that difficulty isn't a scalar, but intuitively it should take "twice as long" to attack an amplified system, rather than taking a constant additional amount of work.
- Security amplification is probably only possible when the initial system is sufficiently secure—if random inputs cause the system to fail with significant probability, then we are likely to be out of luck. This is analogous to reliability amplification, which is only possible when initial system is sufficiently reliable. Under the intended interpretation of "security," humans are relatively secure; we can implement a policy **H**which is relatively hard to exploit (e.g. which humans aren't capable of reliably exploiting). So humans suffice to get the ball rolling.

Capability amplification vs. security amplification

If we interpret “capability” broadly, then capability amplification subsumes security amplification. Moreover, I expect the two problems to be solved by the same mechanism (unlike reliability amplification, which probably requires something completely different). So in some sense it is most natural to think of capability and security amplification as a single problem.

But I think that security amplification has different implications, may require a different style of analysis, and may be possible or impossible independently of other parts of capability amplification. And in general I think it is good practice to try to split up a strong claim into several weaker claims, even if after establishing each of the weaker claims you will just have to prove a completely new theorem that generalizes all of them. The weaker claims give us a nice testing ground in which to find and resolve some of the difficulties before having to confront the whole problem.

Security amplification by meta-execution

I am hopeful that security can be amplified by something like meta-execution.

In meta-execution, the meta-executor never directly looks at the whole system’s input. Instead, it looks at small parts of the input in isolation, and decides for itself how to build those up into a representation of the input.

Similarly, the meta-executor never directly experiences the entire reasoning process; the actual state of the reasoning process is maintained by a large number of meta-executors working in parallel, and each one is responsible for implementing a small part of the reasoning process.

With the exception of the meta-executors who directly observe small pieces of the input, the actual inputs provided to a meta-executor are the product of two forces:

1. The input to the overall system.
2. The optimization applied by the meta-executor as it process that input. The goal of the meta-executor is to ensure that it will be able to correctly handle everything that it sees during the meta-execution, so it is trying to avoid creating states that would exploit vulnerabilities of the meta-executor.

The hope is that an adversary who controls [1] and wants to cause the meta-executor to behave badly must do additional work in order to overcome factor [2].

Realistically, I think that the largest question about this procedure is whether meta-execution can work at all while preserving alignment.If meta-execution can work at all, then I think that there is a reasonable chance that it can also improve security.

That said, even if meta-execution works, it will probably be pretty hard to tell whether it actually reduces security.

As with capability amplification, I think that our goal should be to either convince ourselves that security amplification works, or else to identify an **obstruction**: a type of vulnerability that doesn't become significantly harder to exploit after a step of security amplification.

The rest of this post will try to explain why I think it is plausible that meta-execution could make vulnerabilities more difficult to exploit.

Examples

It seems helpful to consider a bunch of particular examples of vulnerabilities and think about how they behave under meta-execution.

I'll discuss a few examples here: a "magic phrase" which cause the agent **A** to fail badly when encountered, and an "unreasonably compelling argument" which **A** finds compelling even though it should not, and an "appeal to a broken intuition" which exploits a single step of **A**'s reasoning that cannot be broken down further or abstracted. I won't be able to argue convincingly that meta-execution could address any of these hypothetical vulnerabilities. But hopefully I can explain why it seems plausible to me that it would.

(*Warning: the rest of the post is somewhat informal and rambling*.)

Example: magic phrase

Setup. Suppose that the agent **A** fails whenever it sees the phrase "not playing solitaire."

If we had time to do enough capability amplification, then the amplified agent may eventually be able to discover this vulnerability itself and explicitly counteract it. This is not valid security amplification though; it does not help us amplify the security of a low-capability agent **A** in a reasonable amount of time. To amplify security we need to be able to fix vulnerabilities without discovering what they are (both because we occupy the position of the interior, and because we want to be secure against adversaries smarter than we are).

To make the problem harder we could assume that **A** actually fails whenever it merely *thinks of* the phrase "not playing solitaire," and that there is no advance warning or ability to see what is coming. For example, if the letters were revealed one at a time to **A**, then nothing might happen until "not playing so…" is revealed, at which point the agent might imagine several possible completions, chance across "not playing solitaire," and then immediately fail.

Cause for optimism. If a string including "not playing solitaire" is given to the meta-executed system, then **A** won't directly observe it. Instead **A** must process a message like "the string with list of characters [*x*]." **A** could parse this string into words, obtaining a message like "the sentence with list of words [*x*]," where each word is represented by a copy of **A** who has seen that word and can answer questions about it. Then **A** could parse this sentence to form a representation its meaning, e.g. as "the assertion that [*x*] satisfies property [*y*]," with *y* pointing to "does not satisfy the property [*z*]", *z* pointing to "is currently performing action [*w*]" and so on.

In this way, it seems quite plausible that a sentence could be “understood” without any individual agent actually having to look at or even imagine the sequence of letters it contains.

Of course we do have to look at a whole bunch of words in the process of parsing a sentence, and we will have to look at many more if we want to actually do anything with it. So why think the situation is any better?

By hypothesis the agent doesn’t know about the magic words and so can’t actively avoid them; but the agent *can* treat its input with skepticism and try to prevent an adversary from having undue influence over the exact sentences that it produces. To that end, the agent is free to behave stochastically.

If successful, randomization might make it hard to force the agent to say *anyparticular* really unusual phrase with significant probability. (If the magic words aren’t an unusual phrase, then it seems fair for us to give up, based on the initial agent not clearing the required initial level of security. E.g. if the agent fails whenever it sees or thinks the phrase “one more,” we *probably* can’t really help it.)

Hopefully meta-execution has shifted the attacker’s problem from “say the magic phrase” to “engineer a situation which will cause the meta-executor to say the magic phrase with relatively high probability despite its efforts to behave randomly.” It seems likely that this latter problem is quite a bit harder, suggesting that a magic phrase is not a plausible obstruction in the sense described above.

Other remarks. Of course, over the course of a larger computation it becomes more likely that a magic phrase will appear at one point or another, even if the attacker isn’t able to engineer a decisive attack. But as long as the magic phrase is not actually *likely* to appear, then this probability can hopefully be driven back towards zero with reliability amplification. We only need to prevent attacks that succeed with significant probability over the agent’s stochasticity.

This process may sound unreasonably expensive. For example, if parsing a thousand word message, would we need to break it down into individual words and parse them one by one extremely carefully?

The point of *iterating* security amplification is that we don’t have to do this all in one go. When parsing a thousand word message we can break it down into sentences and feed whole sentences to the agent. When processing sentences we can break them down into words. In general, the smaller the task the more careful we can be.

Example: unreasonably compelling arguments

Setup. Suppose that we want to build a system for evaluating arguments about what to do. We think that our starting agent **A** is pretty good—if it sees an argument made by a human, we expect its beliefs to reliably get more accurate rather than less accurate.

But now we would like to evaluate arguments made by more powerful intelligences than humans. We are concerned that some of these arguments may be *unreasonably compelling*—they would cause us to be convinced without being normatively informative. As an extreme case, there may be unreasonably compelling arguments that could convince us to adopt essentially arbitrary positions.

We cannot be protected from an unreasonably compelling argument by changing the way we handle individual words. At some point we are actually going to have to actually evaluate the argument, and that is where we'll run into trouble.

Cause for optimism. In order for meta-execution to handle this problem, we would need to be able to meta-execute the actual evaluation of the argument. For example, rather than inspecting a claimed syllogism and consulting our intuition to determine whether it seems valid, we would need to decide abstractly how to process a question like "does conclusion [*x*] follow from premises [*a*] and [*b*]?" where all of *x*, *a*, and *b* are messages representing parts of the argument.

Of course we could evaluate a proposed syllogism by simply unpacking all of its parts and consulting our intuition to determine whether it seems valid. The first question is: can we can do anything more abstract, that doesn't require looking directly at the whole input? The second question is: if we evaluate an argument in a more abstract way, are we actually more secure?

With respect to the first question: In general I believe that we *can* come up with at-least-slightly abstract procedures for evaluating arguments, which we believe are more accurate than a direct appeal to our intuitions. Although it would obviously be nice to have some convincing theoretical account of the situation, it looks like a largely empirical question. Fortunately, it's an empirical question that can be answered in the short term rather than requiring us to wait until powerful AI systems are available.

With respect to the second question: I think the key property of "unreasonably convincing" arguments is the following. Suppose that you tell me that I will hear an argument from source S, that I will evaluate it *correctly* (knowing that it came from source S), and that I will then come to believe X. After hearing this, I will simply accept X. An evaluation of an argument seems *incorrect* if, given a full understanding of the evaluation process, I wouldn't think that I should have been persuaded.

Now suppose that I find some argument convincing. And suppose that after lightly abstracting my evaluation process it still seems convincing—that is, I look at a sequence of steps like "I concluded that [*x*] followed from [*a*] and [*b*]." and I feel like, in light of that sequence of steps, I was correct to be convinced. It seems to me that then one of two things could be going wrong:

- One of these individual steps was wrong—that is, I asked "Does [*x*] follow from [*a*] and [*b*]?" and got back the answer "It sure does," but only because this step had unreasonably convincing aspects inside of it. It seems like this problem can be fixed by further secure amplification operating on the reasoning with a single step. (Just like we previously discussed breaking a paragraph into sentences, and then making the handling of sentences more secure by breaking sentences down into words.)
- I was incorrectly evaluating the abstract argument—I was misled about whether that sequence of steps *should have been* convincing.

I think the second category is most interesting, because it suggests the possibility of a kind of fixed point. An attacker could construct an argument which convinces me, and such that when I look at an abstracted version of my evaluation process I think that I ought to have been convinced, and when I look at an abstracted version of *that* evaluation process, I think that it *also* was convincing, and so on down the line.

If there is really such a fixed point, then that would be an obstruction to security amplification. After any number of iterations of security amplification, the difficulty of finding an attack would still be upper bounded by the difficulty of finding this fixed point.

I am actually not quite sure what we should make of such a fixed point. Certainly it is conceivable that there could be a plainly wrong argument which had this character. But I have a really hard time constructing a plausible example, and this fixed point does quite strongly suggests that an argument is “convincing for endorsed reasons.”

(As mentioned earlier, it seems much more likely that we simply won’t be able to evaluate arguments using meta-execution, and will be reduced to evaluating them directly.)

If an argument continues to be convincing in light of careful meta-execution, I am not totally sure whether we should call it “unreasonably” convincing. For now it seems like we need a better understanding of the situation, but it is certainly not obvious that there is a problem.

Example: appeal to a broken intuition

Setup. When evaluating an argument or making a decision, we sometimes seem to make a “bare” appeal to intuition—we pose a question to ourselves and discover our intuitive response, without any ability to break the process down further our scrutinize the source of the intuition.

Meta-execution is only feasible if sufficiently “big” intuitions can reliably be broken down into smaller pieces. But nevertheless this process may bottom out at some scale, e.g. perhaps there are questions that can be posed to a human in twenty seconds, where the human has an intuitive answer, but can’t provide any additional detail.

It may be that for the most part these intuitive responses are correct, but that some of them are extremely unreliable. For example, in the process of considering a thought experiment about simulations and torture a person may freak out and give an irrational answer.

If we want to get the benefits from consulting human intuition, then it seems like an adversary could set up a situation where the most important intuitive queries exploit some vulnerability.

Cause for optimism. Even if the internal structure of an intuitive reaction is opaque, there are a number of ways that the surrounding argument can be changed in order to improve security:

- We can query the agent on random variants of an input, with many details varied, in order to avoid problems specific to a particular formulation. In some sense we are left only with cases where the core intuition is itself a vulnerability.
- We can inspect the input (and its meaning) abstractly, to search for any reasons why this particular appeal to intuition may be unreliable.
- We can treat the agent’s reaction as just one piece of evidence that bears on a case. If we learn there is a thought experiment such that considering it causes

the agent to conclude “doing X is extraordinarily important” then that constitutes compelling but not overwhelming evidence.
- We can consider the agent’s response to a range of similar inputs, and make inferences from those responses in the aggregate.

If all of these tests are insufficient to protect us from the purported vulnerability, then again I think it’s not really clear on what grounds we are describing it as a “vulnerability.” It’s not obvious that we should rest comfortably at this point, but at least it seems plausible that we should.

Empirical tests

The discussion in the last section was very vague and intuitive, but fortunately the actual claims at issue seem to empirically accessible. It is very easy to implement meta-execution using humans as the meta-executor. As a result:

- We can *just test* whether we can evaluate arguments or make decisions abstractly in a way that seems at least as good, and preferably better, than evaluating them directly.
- We actually pick a simple idea, and see whether a human meta-executor can abstractly make decisions without ever encountering that idea (even on adversarial inputs).

Mostly I think that many of these issues will become quite obvious as we get some practical experience with meta-execution (and hopefully it will also become clear how to get a better theoretical handle on it).

Last summer I actually spent a while experimenting with meta-execution as part of a metaprogramming project dwimmer. Overall the experience makes me significantly more optimistic about the kinds of claims in the post, though I ended up ambivalent about whether it was a practical way to automate programming in the short term. (I still think it’s pretty plausible, and one of the more promising AI projects I’ve seen, but that it definitely won’t be easy.)

Conclusion

We can attempt to quantify the security of a policy by asking “how hard is it to find an input on which this policy behaves badly?” We can then seek security amplification procedures which make it harder to attack a policy.

I propose meta-execution as a security amplification protocol. I think that the single biggest uncertainty is whether meta-execution can work at all, which is currently an open question.

Even if meta-execution *does* work, it seems pretty hard to figure out whether it actually amplifies security. I sketched a few types of vulnerability and tried to explain why I think that meta-execution might help address these vulnerabilities, but there is clearly a lot of thinking left to do.

If security amplification could work, I think it significantly expands the space of feasible control strategies, offers a particularly attractive approach to running a

massive search without compromising alignment, and makes it much more plausible that we can achieve acceptable robustness to adversarial behavior in general.

This was first published here on 26th October, 2016.

The next post in sequence will be released on Friday 8th Feb, and will be 'Meta-excution' by Paul Christiano.

Meta-execution

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

This post describes meta-execution, my current proposal for capability amplification and security amplification.

(*Meta-execution is annotated functional programming + strong HCH + a level of indirection. It is implemented in the amplify module of my ALBA repository.*)

Goal

We are given an efficient agent **A** which competently pursues some values. We'd like to use a bunch of copies of **A** in order to implement a more powerful and robust agent Meta(**A**) with the same values.

Outline

Our basic plan is to build a machine out of copies of the agent; instead of asking the agent to make a decision directly, we ask it to implement the decision-making process by answering a sequence of questions of the form "what should happen next?"

The basic object in meta-execution is a **message**, which consists of text along with pointers to other messages or to agents.

Each box is a message. **A** is an agent who can respond to queries like "which of X and Y is larger?"

We can represent arbitrarily large objects as giant trees of messages and agents.

Meta-execution first forms a tree representing the question "what should be done?" It then asks the agent **A** to perform a sequence of operations on the tree that eventually lead to an answer. Then it executes that answer.

The initial tree might look something like this:

If you can answer this question, you can implement an agent.

At any given time, an agent who is operating on this tree can only "see" a few messages: it can read the text of those messages, and see pointers like [red] and [blue]. Initially the agent can see only the root of the tree.

If you are an agent tasked with processing a message, there are a few basic operations you can perform. You specify the "targets" of the action by specifying pointers you want to follow:

- Look at another part of the tree which is not currently visible.
- Spawn a new agent, and see a pointer to that agent.
- Send a message to an agent, and see its reply. You can compose a message by writing it out with sub-messages in parentheses (); for example, "What is the smallest element in (the list with first element [green] and remaining elements [purple]) according to [blue]" would produce the message in the first image above, if [green], [purple], [blue] had appropriate values.
- Terminate the current computation by composing a reply. This reply gets sent to the "parent" who initiated the current computation. In the case of the very first agent, who was created in order to answer the original question "what should an agent in state [red] do after receiving input [blue]?", the reply specifies what the overall system should do.

An example execution is illustrated here.

I made a quick demo of the execution process, you can find it here.

And that's basically it. We spawn a new agent, and hand it the "what should we do?" message. It can then take any of the basic actions listed above and see the result. We

repeat that process until the agent returns a message indicating what should be done. We parse the message as an action and new state (see the section on parsing below), we execute the action, and we update the system's state.

The details

Hopefully for most purposes that outline tells you everything you need to know. If not, the easiest way to learn *exactly* how this works is probably just to look at the code.

Meta-execution is implemented as `lambda A : Meta(HCH(A, n))` in the package amplify.__init__, where `n` is the computational budget and `A` is the meta-executor. You can experience *being* the meta-executor by calling `examples.meta.act("test")`. The available commands are described in the README.

Everything is immutable

I assume that we have a digital implementation of **A**, and so we can snapshot and copy it freely. Each time an agent makes an observation and then takes an action, we keep around the original copy of the agent as well as the new copy.

For example, suppose that I send **A** the question *q*, and she replies with the answer *a*. The original agent **A** will not be changed, but I will get back a pointer to a new agent **A**′. There are now two different agents to whom I could send my next question. **A**′ remembers seeing the question *q* and replying with *a*; **A** does not.

Both messages and agents are immutable.

You can't modify a tree of messages, you can only compute a new tree which happens to share some subtrees. If I send a message like "sort the list [purple]," I should expect to get back a reply like "the sorted version is [black]." The original pointer [purple] will never change. This should be familiar to anyone who has experience with functional programming, but might be a little weird to people who aren't used to it.

Parsing outputs

After we run our agent on the "what should we do?" tree, we obtain a message. We need to parse this message as an action. We can do this with the help of an agent, but it requires asking somewhat more specific questions.

First we ask "What string should the agent output?", and then only after we ask "And what state should the agent end up in?"

In response to "what string should the agent output?" we receive a message. To parse it as a string, we can ask "What is the first character of the string?" and then "What is the string obtained by removing the first character?" To parse a character we can ask "What is the ASCII code of the character?" and get a message. To parse an integer we can ask questions like "Is the integer zero?" and then "Is the integer even or odd?"

Everything can bottom out with yes/no questions, to which the response ought to be either the message “yes” or the message “no.” If it’s not, then we can ask again. (The question can make it clear to the answer that the response ought to be yes or no.)

The answer to “what state should the agent end up in” is simpler—the state of the agent is itself a message, so we can just take this at face value.

Budgeting

We can modify the scheme to include a computational budget.

When we create the first agent, we assign it some initial budget, like 10^9 operations.

Each time an agent views an object or sends a message, it uses up 1 operation from its computational budget. When an agent runs out of computational budget, it must immediately compose a reply, it can’t do any other operations. (It is free to return an error message requesting additional resources and explaining why it ran over budget.)

Moreover, each time any agent sends a message and expects a reply, it has to also transfer some of its computational budget to the recipient. The recipient may then spend that budget in order to come up with its answer. When the recipient responds, it may return whatever computational budget remains unused.

Why be functional?

Meta-execution as I’ve described it is purely functional, and the computational model very closely resembles Lisp.

At face value it looks like there is a huge range of universal models that would all serve equally well.

But once we are interested in alignment-preserving computation, I think there isn’t that much room to vary this scheme without making things much uglier. If anyone feels like a different approach to meta-execution would be comparably natural I would be curious to hear about it.

The basic problem is that it’s completely unclear when you “should” mutate part of a mutable program’s state. Every imperative program has a different set of contracts and expectations for how different objects or parts of memory will be used. For meta-execution, any such expectations need to be baked into the agent that we start with. That means that we are stuck with whatever we have at the beginning, and so in some sense we need to start with a “universal” contract.

We have an implicit notion of “accurate” and “helpful” for questions and dialogs. For the version of meta-execution I’ve described, that’s all you need—you never mutate an object, and so you never need to know how an object is used, you just need to figure out what message you should return in response to a question. I don’t see any comparably simple approach in any other computational model.