

A formalization of indirect normativity



Paul Christiano

[Follow](#)

Apr 20, 2012 · 31 min read

This post outlines a formalization of what Nick Bostrom calls “indirect normativity.” I don’t think it’s an adequate solution to the AI control problem; but to my knowledge it was the first precise specification of a goal that meets the “not terrible” bar, i.e. which does not lead to terrible consequences if pursued without any caveats or restrictions. The proposal outlined here was sketched in early 2012 while I was visiting FHI, and was my first serious foray into AI control.

Introduction

When faced with the challenge of writing down precise moral principles, adhering to the standards demanded in mathematics, moral philosophers encounter two serious difficulties:

- Basic notions, like “agent,” “act,” and “motive” are themselves almost inherently imprecise. More complex concepts like “well-being,” “possible world,” and “honesty” are even less approachable from a formal perspective.
- Even expressed in terms of these notions, satisfactory moral principles have proven incredibly elusive. Committing ourselves to any simple, exceptionless theory has proved incompatible with basic ethical intuitions.

In light of these difficulties, a moral philosopher might simply declare: “It is not my place to aspire to mathematical standards of precision. Ethics as a project inherently requires shared language, understanding, and experience; it becomes impossible or meaningless without them.”

This may be a defensible philosophical position, but unfortunately the issue is not entirely philosophical. In the interest of building institutions or machines which reliably pursue what we value, we may one day be forced to describe precisely “what we value” in a way that does not depend on charitable or “common sense” interpretation

(in the same way that we today must describe “what we want done” precisely to computers, often with considerable effort). If some aspects of our values cannot be described formally, then it may be more difficult to use institutions or machines to reliably satisfy them. This is not to say that describing our values formally is necessary to satisfying them, merely that it might make it easier.

Since we are focusing on finding any precise and satisfactory moral theory, rather than resolving disputes in moral philosophy, we will adopt a consequentialist approach without justification and focus on axiology. Moreover, we will begin from the standpoint of expected utility maximization, and leave aside questions about how or over what space the maximization is performed.

We aim to mathematically define a utility function U such that we would be willing to build a hypothetical machine which exceptionlessly maximized U , possibly at the catastrophic expense of any other values. We will assume that the machine has an ability to reason which at least rivals that of humans, and is willing to tolerate arbitrarily complex definitions of U (within its ability to reason about them).

We adopt an indirect approach. Rather than specifying what exactly we want, we specify a process for determining what we want. This process is extremely complex, so that any computationally limited agent will always be uncertain about the process’ output. However, by reasoning about the process it is possible to make judgments about which action has the highest expected utility in light of this uncertainty.

For example, I might adopt the principle: “a state of affairs is valuable to the extent that I would judge it valuable after a century of reflection.” In general I will be uncertain about what I would say after a century, but I can act on the basis of my best guesses: after a century I will probably prefer worlds with more happiness, and so today I should prefer worlds with more happiness. After a century I have only a small probability of valuing trees’ feelings, and so today I should go out of my way to avoid hurting them if it is either instrumentally useful or extremely easy. As I spend more time thinking, my beliefs about what I would say after a century may change, and I will start to pursue different states of affairs even though the formal definition of my values is static. Similarly, I might desire to think about the value of trees’ feelings, if I expect that my opinions are unstable: if I spend a month thinking about trees, my

current views will then be a much better predictor of my views after a hundred years, and if I know better whether or not trees' feelings are valuable, I can make better decisions.

This example is quite informal, but it communicates the main idea of the approach. We stress that the value of our contribution, if any, is in the possibility of a precise formulation. (Our proposal itself will be relatively informal; instead it is a description of how you would arrive at a precise formulation.) The use of indirection seems to be necessary to achieve the desired level of precision.

I. The Proposal

Our proposal contains only two explicit steps:

1. Obtain a precise mathematical characterization of a particular human's brain.
2. Precisely define a completely abstract environment containing an idealized and unbounded computer. Adopt whatever utility function the human would decide on, if allowed free reign in this abstract environment. (With the technical modification that the utility is restricted to take on values between 0 and 1.)

Each of these steps requires substantial elaboration, but we must also specify what we expect the human to do with these tools.

1. Build a community of copies of herself and an idealized environment for interaction. Deliberate on critical scientific and philosophical questions, and decide how to proceed (possibly in the spirit of what follows).
2. Search through the space of all possible worlds to find a simulation of our own. By interacting with this simulation, incorporate simulations of many living humans into her environment.
3. Continue to make scientific and philosophical progress, as safely as possible. Possibly engage in principled self-modification or understand and build machine intelligences which reflect relevant ethical principles.
4. Output a utility function. This function may be a compact set of ethical principles, or it may require many simulated humans or machine intelligences to apply their own (radically modified)

common sense to evaluate states of affairs on a case-by-case basis.

This proposal is best understood in the context of other fantastic-seeming proposals, such as “my utility is whatever I would write down if I reflected for a thousand years without interruption or biological decay.” The counterfactual events which take place within the definition are far beyond the realm our intuition recognizes as “realistic,” and have no place except in thought experiments. But to the extent that we can reason about these counterfactuals and change our behavior on the basis of that reasoning (if so motivated), we can already see how such fantastic situations could affect our more prosaic reality.

The remainder of this document consists of brief elaboration of some of these steps, and a few arguments about why this is a desirable process.

Brain Emulation

The first step of our proposal is a high-fidelity mathematical model of human cognition. We will set aside philosophical troubles, and assume that the human brain is a purely physical system which may be characterized mathematically. Even granting this, it is not clear how we can realistically obtain such a characterization.

The most obvious approach to characterizing a brain is to combine measurements of its behavior or architecture with an understanding of biology, chemistry, and physics. This project represents a massive engineering effort which is currently just beginning. Most pessimistically, our proposal could be postponed until this project’s completion. This could still be long before the mathematical characterization of the brain becomes useful for running experiments or automating human activities: because we are interested only in a definition, we do not care about having the computational resources necessary to simulate the brain.

An impractical mathematical definition, however, may be much easier to obtain. We can define a model of a brain in terms of exhaustive searches which could never be practically carried out. For example, given some observations of a neuron, we can formally define a brute force search for a model of that neuron. Similarly, given models of individual neurons we may be able to specify a brute force search over all ways of connecting those neurons which account

for our observations of the brain (say, some data acquired through functional neuroimaging).

It may be possible to carry out this definition without exploiting any structural knowledge about the brain, beyond what is necessary to measure it effectively. By collecting imaging data for a human exposed to a wide variety of stimuli, we can recover a large corpus of data which must be explained by any model of a human brain. Moreover, by using our explicit knowledge of human cognition we can algorithmically generate an extensive range of tests which identify a successful simulation, by probing responses to questions or performance on games or puzzles.

In fact, this project may be possible using existing resources. The complexity of the human brain is not as unapproachable as it may at first appear: though it may contain 10^{14} synapses, each described by many parameters, it can be specified much more compactly. A newborn's brain can be specified by about 10^9 bits of genetic information, together with a recipe for a physical simulation of development. The human brain appears to form new long-term memories at a rate of 1–2 bits per second, suggesting that it may be possible to specify an adult brain using 10^9 additional bits of experiential information. This suggests that it may require only about 10^{10} bits of information to specify a human brain, which is at the limits of what can be reasonably collected by existing technology for functional neuroimaging.

This discussion has glossed over at least one question: what do we mean by ‘brain emulation’? Human cognition does not reside in a physical system with sharp boundaries, and it is not clear how you would define or use a simulation of the “input-output” behavior of such an object.

We will focus on some system which does have precisely defined input-output behavior, and which captures the important aspects of human cognition. Consider a system containing a human, a keyboard, a monitor, and some auxiliary instruments, well-insulated from the environment except for some wires carrying inputs to the monitor and outputs from the keyboard and auxiliary instruments (and wires carrying power). The inputs to this system are simply screens to be displayed on the monitor (say delivered as a sequence to be displayed one after another at 30 frames per second), while the outputs are the information conveyed from the keyboard and the

other measuring apparatuses (also delivered as a sequence of data dumps, each recording activity from the last 30th of a second).

This “human in a box” system can be easily formally defined if a precise description of a human brain and coarse descriptions of the human body and the environment are available. Alternatively, the input-output behavior of the human in a box can be directly observed, and a computational model constructed for the entire system. Let H be a mathematical definition of the resulting (randomized) function from input sequences $(In(1), In(2), \dots, In(K))$ to the next output $Out(K)$. H is, by design, a good approximation to what the human “would output” if presented with any particular input sequence.

Using H , we can mathematically define what “would happen” if the human interacted with a wide variety of systems. For example, if we deliver $Out(K)$ as the input to an abstract computer running some arbitrary software, and then define $In(K+1)$ as what the screen would next display, we can mathematically define the distribution over transcripts which would have arisen if the human had interacted with the abstract computer. This computer could be running an interactive shell, a video game, or a messaging client.

Note that H reflects the behavior of a particular human, in a particular mental state. This state is determined by the process used to design H , or the data used to learn it. In general, we can control H by choosing an appropriate human and providing appropriate instructions / training. More emulations could be produced by similar measures if necessary. Using only a single human may seem problematic, but we will not rely on this lone individual to make all relevant ethical judgments. Instead, we will try to select a human with the motivational stability to carry out the subsequent steps faithfully, which will define U using the judgment of a community consisting of many humans.

This discussion has been brief and has necessarily glossed over several important difficulties. One difficulty is the danger of using computationally unbounded brute force search, given the possibility of short programs which exhibit goal-oriented behavior. Another difficulty is that, unless the emulation project is extremely conservative, the models it produces are not likely to be fully-functional humans. Their thoughts may be blurred in various ways, they may be missing many memories or skills, and they may lack important functionalities such as long-term memory formation or

emotional expression. The scope of these issues depends on the availability of data from which to learn the relevant aspects of human cognition. Realistic proposals along these lines will need to accommodate these shortcomings, relying on distorted emulations as a tool to construct increasingly accurate models.

The Virtual Environment

For any idealized “software”, with a distinguished instruction return, we can use H to mathematically define the distribution over return values which would result, if the human were to interact with that software. We will informally define a particular program T which provides a rich environment, in which the remainder of our proposal can be implemented. From a technical perspective this will be the last step of our proposal. The remaining steps will be reflected only in the intentions and behavior of the human being simulated in H.

Fix a convenient and adequately expressive language (say a dialect of Python designed to run on an abstract machine). T implements a standard interface for an interactive shell in this language: the user can look through all of the past instructions that have been executed and their return values (rendered as strings) or execute a new instruction. We also provide symbols representing H and T themselves (as functions from sequences of K inputs to a value for the Kth output). We also provide some useful information (such as a snapshot of the Internet, and some information about the process used to create H and T), which we encode as a bit string and store in a single environment variable data. We assume that our language of choice has a return instruction, and we have T return whenever the user executes this instruction. Some care needs to be taken to define the behavior if T enters an infinite loop—we want to minimize the probability that the human accidentally hangs the terminal, with catastrophic consequences, but we cannot provide a complete safety-net without running into unresolvable issues with self-reference.

We define U to be the value returned by H interacting with T. If H represented an unfortunate mental state, then this interaction could be short and unproductive: the simulated human could just decide to type ‘return 0’ and be done with it. However, by choosing an appropriate human to simulate and inculcating an appropriate mental state, we can direct the process further.

We intend for H to use the resources in T to initiate a larger deliberative process. For example, the first step of this process may be to instantiate many copies of H, interacting with variants of

messaging clients which are in contact with each other. The return value from the original process could then be defined as the value returned by a designated ‘leader’ from this community, or as a majority vote amongst the copies of H, or so on. Another step might be to create appropriate realistic virtual environments for simulated brains, rather than confining them to boxes. For motivational stability, it may be helpful to design various coordination mechanisms, involving frameworks for interaction, “cached” mental states which are frequently re-instantiated, or sanity checks whereby one copy of H monitors the behavior of another.

The resulting communities of simulated brains then engage in a protracted planning process, ensuring that subsequent steps can be carried out safely or developing alternative approaches. The main priority of this community is to reduce the probability of errors as far as possible (exactly what constitutes an ‘error’ will be discussed at more length later). At the end of this process, we obtain a formal definition of a new protocol H+, which submits its inputs for consideration to a large community and then produces its outputs using some deliberation mechanism (democratic vote, one leader using the rest of the community as advisors, etc.)

Finding Earth

The next step requires our community of simulated brains to construct a detailed simulation of Earth which they can observe and manipulate. Once they have such a simulation, they have access to all of the data which would have been available on Earth. In particular, they can now explore many possible futures and construct simulations for each living human.

In order to locate Earth, we will again leverage an exhaustive search. First, H+ decides on informal desiderata for an “Earth simulation.” These are likely to be as follows:

- Comprehensible: the code for the simulation should relate to the best understanding of physics which H+ can develop. This may require operating at a coarser level than the “real” physics underlying our world, and it probably requires structuring the code for the simulation in a way that is not maximally efficient. This requirement is designed to ensure that the ‘views’ and ‘interventions’ produced below are acting the expected way when they are being tested for accuracy, so that generalization to new situations will work as expected.

- Supports viewing: H+ should decide on an event to use as a landmark (such as a particular data entry process involved in the definition of H), and search for simulations which support viewing scenes defined relative to that landmark. For example, they might require the simulation to provide a function $\text{View}(x, y, z, t)$, which provides a “human’s view” from a point specified by the coordinates (x, y, z, t) , defined relative to the landmark.
- Accurate: the simulation should consistently conform to expectations about Earth. The history of the individual simulated in H should conform to H’s simulated memories; the simulation should follow the same broad historical course as described in data and remembered by H; all of the other data (for example, the complete snapshot of the Internet) should also be in agreement. Perhaps most importantly, the details of whatever system’s behavior depends on U should be recorded as part of the data.
- Supports intervention: H+ should decide on a particular class of interventions (such as non-physically setting the output of a particular invocation of one program to be equal to S), and search for simulations which support intervening in that way. For example, they might require the simulation to provide a function $\text{Modify}(S)$ which causes all future $\text{View}()$ instructions to show what would have happened if the output of the intervention corresponding to S had been performed.

Once H+ has decided on the desiderata, it uses a brute force search to find a simulation satisfying them: for each possible program it instantiates a new copy of H+ tasked with evaluating whether that program is an acceptable simulation. We then define E to be a uniform distribution over programs which pass this evaluation.

We might have doubts about whether this process produces the “real” Earth—perhaps even once we have verified that it is identical according to a laundry list of measures, it may still be different in other important ways. There are two reasons why we might care about such differences. First, if the simulated Earth has a substantially different set of people than the real Earth, then a different set of people will be involved in the subsequent decision making. If we care particularly about the opinions of the people who actually exist (which the reader might well, being amongst such people!) then this may be unsatisfactory. Second, if events transpire significantly differently on the simulated Earth than the real Earth, value judgments designed to guide behavior appropriately in the

simulated Earth may lead to less appropriate behaviors in the real Earth. (This will not be a problem if our ultimate definition of U consists of universalizable ethical principles, but we will see that U might take other forms.)

These concerns are addressed by a few broad arguments. First, checking a detailed but arbitrary ‘laundry list’ actually provides a very strong guarantee. For example, if this laundry list includes verifying a snapshot of the Internet, then every event or person documented on the Internet must exist unchanged, and every keystroke of every person composing a document on the Internet must not be disturbed. If the world is well interconnected, then it may be very difficult to modify parts of the world without having substantial effects elsewhere, and so if a long enough arbitrary list of properties is fixed, we expect nearly all of the world to be the same as well. Second, if the essential character of the world is fixed but detailed are varied, we should expect the sort of moral judgments reached by consensus to be relatively constant. Finally, if the system whose behavior depends on these moral judgments is identical between the real and simulated worlds, then outputting a U which causes that system to behave a certain way in the simulated world will also cause that system to behave that way in the real world.

Once H+ has defined a simulation of the world which permits inspection and intervention, by careful trial and error H+ can inspect a variety of possible futures. In particular, they can find interventions which cause the simulated human society to conduct a real brain emulation project and produce high-fidelity brain scans for all living humans.

Once these scans have been obtained, H+ can use them to define U as the output of a new community, H++, which draws on the expertise of all living humans operating under ideal conditions. There are two important degrees of flexibility: how to arrange the community for efficient communication and deliberation, and how to delegate the authority to define U. In terms of organization, the distinction between different approaches is probably not very important. For example, it would probably be perfectly satisfactory to start from a community of humans interacting with each other over something like the existing Internet (but on abstract, secure infrastructure). More important are the safety measures which would be in place, and the mechanism for resolving differences of value between different simulated humans.

The basic approach to resolving disputes is to allow each human to independently create a utility function U , each bounded in the interval $[0, 1]$, and then to return their average. This average can either be unweighted, or can be weighted by a measure of each individual's influence in the real world, in accordance with a game-theoretic notion like the Shapley value applied to abstract games or simulations of the original world. More sophisticated mechanisms are also possible, and may be desirable. Of course these questions can and should be addressed in part by H^+ during its deliberation in the previous step. After all, H^+ has access to an unlimited length of time to deliberate and has infinitely powerful computational aids. The role of our reasoning at this stage is simply to suggest that we can reasonably expect H^+ to discover effective solutions.

As when discussing discovering a brain simulation by brute force, we have skipped over some critical issues in this section. In general, brute force searches (particularly over programs which we would like to run) are quite dangerous, because such searches will discover many programs with destructive goal-oriented behaviors. To deal with these issues, in both cases, we must rely on patience and powerful safety measures.

Extrapolation

Once we have a formal description of a community of interacting humans, given as much time as necessary to deliberate and equipped with infinitely powerful computational aids, it becomes increasingly difficult to make coherent predictions about their behavior. Critically, though, we can also become increasingly confident that the outcome of their behavior will reflect their intentions. We sketch some possibilities, to illustrate the degree of flexibility available.

Perhaps the most natural possibility is for this community to solve some outstanding philosophical problems and to produce a utility function which directly captures their preferences. However, even if they quickly discovered a formulation which appeared to be attractive, they would still be wise to spend a great length of time and to leverage some of these other techniques to ensure that their proposed solution was really satisfactory.

Another natural possibility is to eschew a comprehensive theory of ethics, and define value in terms of the community's judgment. We can define a utility function in terms of the hypothetical judgments of astronomical numbers of simulated humans, collaboratively evaluating the goodness of a state of affairs by examining its history

at the atomic level, understanding the relevant higher-order structure, and applying human intuitions.

It seems quite likely that the community will gradually engage in self-modifications, enlarging their cognitive capacity along various dimensions as they come to understand the relevant aspects of cognition and judge such modifications to preserve their essential character. Either independently or as an outgrowth of this process, they may (gradually or abruptly) pass control to machine intelligences which they are suitably confident expresses their values. This process could be used to acquire the power necessary to define a utility function in one of the above frameworks, or understanding value-preserving self-modification or machine intelligence may itself prove an important ingredient in formalizing what it is we value. Any of these operations would be performed only after considerable analysis, when the original simulated humans were extremely confident in the desirability of the results.

Whatever path they take and whatever coordination mechanisms they use, eventually they will output a utility function U' . We then define $U = 0$ if $U' < 0$, $U = 1$ if $U' > 1$, and $U = U'$ otherwise.

II. Desirability

At this point we have offered a proposal for formally defining a function U . We have made some general observations about what this definition entails. But now we may wonder to what extent U reflects our values, or more relevantly, to what extent our values are served by the creation of U -maximizers. Concerns may be divided into a few natural categories:

- Even if the process works as intended, the ultimate intentions of the simulations within the process do not reflect our values, so we should not expect them to output a U which reflects our values.
- The process has some chance of failing to work as intended. If it fails completely, then value will certainly be lost. If it fails with some small probability (over the stochasticity within the process, or over our uncertainty about its behavior) then the resulting utility function may be substantially altered and desirable outcomes may no longer be achieved.
- Any real U -maximizer will be unable to actually carry out the simulation described in the definition of U , and so even if this

process would produce a U reflecting our values, it is not clear how a real U-maximizer will behave.

- Because we have started from the standpoint of bounded expected utility maximization we have ruled out the vast majority of possible value systems. It may be that there is no way to “shorehorn” our values into this framework, so that no matter what bounded U we choose, the resulting agent doesn’t satisfy our values very well.
- Are there possible negative consequences to “passing the buck” as in this proposal?
- Could the process itself be morally abhorrent?

We respond to each of these objections in turn.

If it Works as Intended, Will This Process Reflect our Values?

If the process works as intended, we will reach a stage in which a large community of humans reflects on their values, undergoes a process of discovery and potentially self-modification, and then outputs its result. We may be concerned that this dynamic does not adequately capture what we value.

For example, we may believe that some other extrapolation dynamic captures our values, or that it is morally desirable to act on the basis of our current beliefs without further reflection, or that the presence of realistic disruptions, such as the threat of catastrophe, has an important role in shaping our moral deliberation.

The important observation, in the defense of our proposal, is that whatever objections we could think of today, we could think of within the simulation. If, upon reflection, we decide that too much reflection is undesirable, we can simply change our plans appropriately. If we decide that realistic interference is important for moral deliberation, we can construct a simulation in which such interference occurs, or determine our moral principles by observing moral judgments in our own world’s possible futures.

There is some chance that this proposal is inadequate for some reason which won’t be apparent upon reflection, but then by definition this is a fact which we cannot possibly hope to learn by

deliberating now. It therefore seems quite difficult to maintain objections to the proposal along these lines.

One aspect of the proposal does get “locked in,” however, after being considered by only one human rather than by a large civilization: the distribution of authority amongst different humans, and the nature of mechanisms for resolving differing value judgments.

Here we have two possible defenses. One is that the mechanism for resolving such disagreements can be reflected on at length by the individual simulated in H. This individual can spend generations of subjective time, and greatly expand her own cognitive capacities, while attempting to determine the appropriate way to resolve such disagreements. However, this defense is not completely satisfactory: we may be able to rely on this individual to produce a very technically sound and generally efficient proposal, but the proposal itself is quite value laden and relying on one individual to make such a judgment is in some sense begging the question.

A second, more compelling, defense, is that the structure of our world has already provided a mechanism for resolving value disagreements. By assigning decision-making weight in a way that depends on current influence (for example, as determined by the simulated ability of various coalitions to achieve various goals), we can generate a class of proposals which are at a minimum no worse than the status quo. Of course, these considerations will also be shaped by the conditions surrounding the creation or maintenance of systems which will be guided by U—for example, if a nation were to create a U-maximizer, they might first adopt an internal policy for assigning influence on U. By performing this decision making in an idealized environment, we can also reduce the likelihood of destructive conflict and increase the opportunities for mutually beneficial bargaining. We may have moral objections to codifying this sort of “might makes right” policy, favoring a more democratic proposal or something else entirely, but as a matter of empirical fact a more ‘cosmopolitan’ proposal will be adopted only if it is supported by those with the appropriate forms of influence, a situation which is unchanged by precisely codifying existing power structure.

Finally, the values of the simulations in this process may diverge from the values of the original human models, for one reason or another. For example, the simulated humans may predictably disagree with the original models about ethical questions by virtue of (probably) having no physical instantiation. That is, the output of this process is

defined in terms of what a particular human would do, in a situation which that human knows will never come to pass. If I ask “What would I do, if I were to wake up in a featureless room and told that the future of humanity depended on my actions?” the answer might begin with “become distressed that I am clearly inhabiting a hypothetical situation, and adjust my ethical views to take into account the fact that people in hypothetical situations apparently have relevant first-person experience.” Setting aside the question of whether such adjustments are justified, they at least raise the possibility that our values may diverge from those of the simulations in this process.

These changes might be minimized, by understanding their nature in advance and treating them on a case-by-case basis (if we can become convinced that our understanding is exhaustive). For example, we could try and use humans who robustly employ updateless decision theories which never undergo such predictable changes, or we could attempt to engineer a situation in which all of the humans being emulated do have physical instantiations, and naive self-interest for those emulations aligns roughly with the desired behavior (for example, by allowing the early emulations to “write themselves into” our world).

Will This Process Work as Intended?

We can imagine many ways in which this process can fail to work as intended—the original brain emulations may accurately model human behavior, the original subject may deviate from the intended plans, or simulated humans can make an error when interacting with their virtual environment which causes the process to get hijacked by some unintended dynamic.

Robustness

We can argue that the proposal is likely to succeed, and can bolster the argument in various ways (by reducing the number of assumptions necessary for success, building in fault-tolerance, justifying each assumption more rigorously, and so on). However, we are unlikely to eliminate the possibility of error. Therefore we need to argue that if the process fails with some small probability, the resulting values will only be slightly disturbed.

This is the reason for requiring U to lie in the interval $[0, 1]$ —we will see that this restriction bounds the damage which may be done by an unlikely failure.

If the process fails with some small probability ε , then we can represent the resulting utility function as $U = (1-\varepsilon) U_1 + \varepsilon U_2$, where U_1 is the intended utility function and U_2 is a utility function produced by some arbitrary error process. Now consider two possible states of affairs A and B such that $U_1(A) > U_1(B) + \varepsilon / (1-\varepsilon) \approx U_1(B) + \varepsilon$. Then since $0 \leq U_2 \leq 1$, we have:

$$U(A) = (1-\varepsilon) U_1(A) + \varepsilon U_2(A) > (1-\varepsilon) U_1(B) + \varepsilon \geq (1-\varepsilon) U_1(B) + \varepsilon$$

$$U_2(B) = U(B)$$

Thus if A is substantially better than B according to U_1 , then A is better than B according to U. This shows that a small probability of error, whether coming from the stochasticity of our process or an agent's uncertainty about the process' output, has only a small effect on the resulting values.

Moreover, the process contains humans who have access to a simulation of our world. This implies, in particular, that they have access to a simulation of whatever U-maximizing agents exist in the world, and they have knowledge of those agents' beliefs about U. This allows them to choose U with perfect knowledge of the effects of error in these agents' judgments.

In some cases this will allow them to completely negate the effect of error terms. For example, if the randomness in our process causes a perfectly cooperative community of simulated humans to "control" U with probability $2/3$, and causes an arbitrary adversary to control it with probability $1/3$, then the simulated humans can spend half of their mass outputting a utility function which exactly counters the effect of the adversary.

In general, the situation is not quite so simple: the fraction of mass controlled by any particular coalition will vary as the system's uncertainty about U varies, and so it will be impossible to counteract the effect of an error term in a way which is time-independent. Instead, we will argue later that an appropriate choice of a bounded and noisy U can be used to achieve a very wide variety of effective behaviors of U-maximizers, overcoming the limitations both of bounded utility maximization and of noisy specification of utility functions.

Other errors

Many possible problems with this scheme were described or implicitly addressed above. But that discussion was not exhaustive,

and there are some classes of errors that fall through the cracks.

One interesting class of failures concerns changes in the values of the hypothetical human H. This human is in a very strange situation, and it seems quite possible that the physical universe we know contains extremely few instances of that situation (especially as the process unfolds and becomes more exotic). So H's first-person experience of this situation may lead to significant changes in H's views.

For example, our intuition that our own universe is valuable seems to be derived substantially from our judgment that our own first-person experiences are valuable. If hypothetically we found ourselves in a very alien universe, it seems quite plausible that we would judge the experiences within that universe to be morally valuable as well (depending perhaps on our initial philosophical inclinations).

Another example concerns our self-interest: much of individual humans' values seem to depend on their own anticipations about what will happen to them, especially when faced with the prospect of very negative outcomes. If hypothetically we woke up in a completely non-physical situation, it is not exactly clear what we would anticipate, and this may distort our behavior. Would we anticipate the planned thought experiment occurring as planned? Would we focus our attention on those locations in the universe where a simulation of the thought experiment might be occurring? This possibility is particularly troubling in light of the incentives our scheme creates—anyone who can manipulate H's behavior can have a significant effect on the future of our world, and so many may be motivated to create simulations of H.

How Will a U-Maximizer Behave In Light of Uncertainty About U?

A realistic U-maximizer will not be able to carry out the process described in the definition of U—in fact, this process probably requires immensely more computing resources than are available in the universe. (It may even involve the reaction of a simulated human to watching a simulation of the universe!) To what extent can we make robust guarantees about the behavior of such an agent?

We have already touched on this difficulty when discussing the maxim “A state of affairs is valuable to the extent I would judge it valuable after a century of reflection.” We cannot generally predict our own judgments in a hundred years’ time, but we can have well-founded beliefs about those judgments and act on the basis of those

beliefs. We can also have beliefs about the value of further deliberation, and can strike a balance between such deliberation and acting on our current best guess.

A U-maximizer faces a similar set of problems: it cannot understand the exact form of U, but it can still have well-founded beliefs about U, and about what sorts of actions are good according to U. For example, if we suppose that the U-maximizer can carry out any reasoning that we can carry out, then the U-maximizer knows to avoid anything which we suspect would be bad according to U (for example, torturing humans). Even if the U-maximizer cannot carry out this reasoning, as long as it can recognize that humans have powerful predictive models for other humans, it can simply appropriate those models (either by carrying out reasoning inspired by human models, or by simply asking).

Moreover, the community of humans being simulated in our process has access to a simulation of whatever U-maximizer is operating under this uncertainty, and has a detailed understanding of that uncertainty. This allows the community to shape their actions in a way with predictable (to the U-maximizer) consequences.

Can Our Values be Expressed in This Framework?

It is easily conceivable that our values cannot be captured by a bounded utility function. Easiest to imagine is the possibility that some states of the world are much better than others, in a way that requires unbounded utility functions. But it is also conceivable that the framework of utility maximization is fundamentally not an appropriate one for guiding such an agent's action, or that the notion of utility maximization hides subtleties which we do not yet appreciate.

We will argue that it is possible to transform bounded utility maximization into an arbitrary alternative system of decision-making, by designing a utility function which rewards worlds in which the U-maximizer replaced itself with an alternative decision-maker.

It is straightforward to design a utility function which is maximized in worlds where any particular U-maximizer converted itself into a non-U-maximizer—even if no simple characterization can be found for the desired act, we can simply instantiate many communities of humans

to look over a world history and decide whether or not they judge the U-maximizer to have acted appropriately.

The more complicated question is whether a realistic U-maximizer can be made to convert itself into a non-U-maximizer, given that it is logically uncertain about the nature of U. It is at least conceivable that it couldn't: if the desirability of some other behavior is only revealed by philosophical considerations which are too complex to ever be discovered by physically limited agents, then we should not expect any physically limited U-maximizer to respond to those considerations. Of course, in this case we could also not expect normal human deliberation to correctly capture our values. The relevant question is whether a U-maximizer could switch to a different normative framework, if an ordinary investment of effort by human society revealed that a different normative framework was more appropriate.

If a U-maximizer does not spend any time investigating this possibility, than it may not be expected to act on it. But to the extent that we assign a significant probability to the simulated humans deciding that a different normative framework is more appropriate, and to the extent that the U-maximizer is able to either emulate or accept our reasoning, it will also assign a significant probability to this possibility (unless it is able to rule it out by more sophisticated reasoning). If we (and the U-maximizer) expect the simulations to output a U which rewards a switch to a different normative framework, and this possibility is considered seriously, then U-maximization entails exploring this possibility. If these explorations suggest that the simulated humans probably do recommend some particular alternative framework, and will output a U which assigns high value to worlds in which this framework is adopted and low value to worlds in which it isn't, then a U-maximizer will change frameworks.

Such a “change of frameworks” may involve sweeping action in the world. For example, the U-maximizer may have created many other agents which are pursuing activities instrumentally useful to maximizing U. These agents may then need to be destroyed or altered; anticipating this possibility, the U-maximizer is likely to take actions to ensure that its current “best guess” about U does not get locked in.

This argument suggests that a U-maximizer could adopt an arbitrary alternative framework, if it were feasible to conclude that humans

would endorse that framework upon reflection.

Is “Passing the Buck” Problematic?

Our proposal appears to be something of a cop out, in that it declines to directly take a stance on any ethical issues. Indeed, not only do we fail to specify a utility function ourselves, but we expect the simulations to which we have delegated the problem to in turn delegate it at least a few more times. Clearly at some point this process must bottom out with actual value judgments, and we may be concerned that this sort of “passing the buck” is just obscuring deeper problems which will arise when the process does bottom out.

As observed above, whatever such concerns we might have can also be discovered by the simulations we create. If there is some fundamental difficulty which always arises when trying to assign values, then we certainly have not exacerbated this problem by delegation. Nevertheless, there are at least two coherent objections one might raise:

- Even if the simulated humans can uncover any objections we could raise now, this does not guarantee that we can ignore all objections. After all, the objection “Isn’t passing the buck problematic?” could be raised at every stage, and always countered by the same response: “If it is problematic, then this will be realized by the people to whom we have passed the buck.” If we do not take this objection seriously, then it may be that none of the delegates take it seriously either, and this call and response could be repeated indefinitely.
- This proposal could fail in many (potentially unexpected) ways. If fundamentally resolving ethics requires overcoming some difficulties which we are evading by passing the buck, then we may be adding additional risk without buying much benefit.

Both of these objections can be met with a single response. In the current world, we face a broad range of difficult and often urgent problems. By passing the buck the first time, we delegate resolution of ethical challenges to a civilization which does not have to deal with some of these difficulties—in particular, it faces no urgent existential threats. This allows us to divert as much energy as possible to dealing with practical problems today, while still capturing most of the benefits of nearly arbitrarily extensive ethical deliberation.

Does This Process Have Moral (Dis)Value?

This process is defined in terms of the behavior of unthinkably many hypothetical brain emulations. It is conceivable that the moral status of these emulations may be significant.

We must make a distinction between two possible sources of moral value: it could be the case that a U-maximizer carries out simulations on physical hardware in order to better understand U, and these simulations have moral value, or it could be the case that the hypothetical emulations themselves have moral value.

In the first case, we can remark that the moral value of such simulations is itself incorporated into the definition of U. Therefore a U-maximizer will be sensitive to the possible suffering of simulations it runs while trying to learn about U—as long as it believes that we may might be concerned about the simulations' welfare, upon reflection, it can rely as much as possible on approaches which do not involve running simulations, which deprive simulations of the first-person experience of discomfort, or which estimate outcomes by running simulations in more pleasant circumstances. If the U-maximizer is able to foresee that we will consider certain sacrifices in simulation welfare worthwhile, then it will make those sacrifices. In general, in the same way that we can argue that estimates of U reflect our values over states of affairs, we can argue that estimates of U reflects our values over processes for learning about U.

In the second case, a U-maximizer in our world may have little ability to influence the welfare of hypothetical simulations invoked in the definition of U. However, the possible disvalue of these simulations' experiences are probably seriously diminished.

In general the moral value of such hypothetical simulations' experiences is somewhat dubious. If we simply write down the definition of U, these simulations seem to have no more reality than story-book characters whose activities we describe.

The best arguments for their moral relevance comes from the great causal significance of their decisions: if the actions of a powerful U-maximizer depend on its beliefs about what a particular simulation would do in a particular situation, including for example that simulation's awareness of discomfort or fear, or confusion at the absurdity of the hypothetical situation in which they find themselves, then it may be the case that those emotional responses are granted moral significance. However, although we may define astronomical numbers of hypothetical simulations, the detailed emotional

responses of very view of these simulations will play an important role in the definition of U.

Moreover, for the most part the existences of the hypothetical simulations we define are extremely well-controlled by those simulations themselves, and may be expected to be counted as unusually happy by the lights of the simulations themselves. The early simulations (who have less such control) are created from an individual who has provided consent and is selected to find such situations particularly non-distressing.

Finally, we observe that U can exert control over the experiences of even hypothetical simulations. If the early simulations would experience morally relevant suffering because of their causal significance, but the later simulations they generate robustly disvalue this suffering, the later simulations can simulate each other and ensure that they all take the same actions, eliminating the causal significance of the earlier simulations.

. . .

Originally published at ordinaryideas.wordpress.com on April 21, 2012.

The golden rule



Paul Christiano

[Follow](#)

Nov 29, 2014 · 6 min read

Most of my moral intuitions are well-encapsulated by the maxim: “do unto others as you would have them do unto you.” This is a principle which has extremely broad intuitive appeal, and so it seems worth exploring how I end up with a relatively unusual ethical perspective.

I think there are at least four ways in which my moral views are somewhat unusual:

- I am a committed consequentialist.
- I think that helping twice as many people is roughly twice as good, and I am unusually quantitative and hard-nosed about altruism.
- I think that the welfare of future people matters about as much as the welfare of existing people.
- I think that making happy people is extremely valuable, and would be deeply saddened by a future in which the population was much smaller than it could be.

I think that there are a few modest differences in my understanding of the golden rule that lead to these practical differences:

1. I think about helping twice as many people as if I were helping each person with twice the probability. (Note that from behind a veil of ignorance, if twice as many people are helped I *do* have twice the probability of being helped.)
2. I am very happy to exist. I would accept relatively large decreases in welfare, in exchange for a higher probability of existing.

Consequentialism

When people are considering several policies, some of which might help me, I would really prefer they do the one that helps me the most. I don’t care if they are virtuous, or if they violate deontological side constraints, or if they act according to a maxim that could be

universalized, or whatever (except insofar as those things bear on how effectively they help me). I just want to get the things that I want. So when I consider helping other people, I think I should just help them as effectively as possible, rather than being virtuous, satisfying side constraints, acting according to universalizable maxims, etc.

(I should flag some further subtleties here concerning meddling preferences, but they don't change the conclusion and I don't want to get bogged down.)

Aggregation

As a consequence of [1], when I think about giving X to two people or giving Y to one person, I try to think about whether I would prefer receive X with probability 2% or Y with probability 1%. I think that this is a relatively uncommon perspective. My best guess is that the difference is mostly a manifestation of my very quantitative attitude, rather than serious philosophical disagreements.

The future

I consider helping people roughly equally good whether they live now or in the future. I don't like the version of the golden rule that says "Do unto others [who live nearby] as you would have them do unto you," and I don't like the version that says "Do unto others [who are your contemporaries] as you would have them do unto you" either. And I definitely don't like the version that says "Do unto others [who might do back] as you would have them do unto you."

Though it may be question-begging, the reason I don't like these principles is that they violate the golden rule: I want people in history to try to help me live a good life. When I think about people helping *me* I'm not going to say "this food is tasty but I *really* wish that it had been secured by the good will of my neighbor rather than someone living 50 years ago and 10,000 miles away." There is a further thing I find valuable in the goodwill of neighbors and the vitality of a community, but that's just another good that should be weighted up in the calculus. And given the kinds of tradeoffs that are actually available, I tend to think that goods like health and prosperity are just a lot more important than the advantages of being able to experience gratitude in person rather than owing it to someone who lived long ago or far away; our ability to help those who live far away or in the future appears to be so much greater than our ability to help those

around us (at least for readers living in the rich world of the 21st century).

Creating people

As a consequence of [3], when I think about changes that would bring new people into the world with good lives, I tend to think that those changes are valuable. If I imagine someone deciding whether they should take an action that would bring me into existence, I very much want them to take it. In combination with [1], when I think about the comparison between 20 billion people existing under conditions X or 10 billion people existing under conditions Y, I try to think about whether I would prefer exist under conditions X with probability 2% or under conditions Y with probability 1%. (Carl Shulman has written a post that takes this idea to one logical extreme.)

The kinds of tradeoffs that I find myself considering in my own do-gooding seem to be modest losses in quality of life for existing people in exchange for a small improvement in the probability of many billions of billions of people existing and having rich and valuable lives. Scaled up, I think the tradeoffs look something like making existing people 10% poorer in exchange for increasing by 0.1% the probability of a prosperous and very (very) large future. On the kind of calculus outlined above, I am quite happy with this tradeoff.

While this view is still related to my very quantitative outlook, I think disagreement about this view is much more likely to be due to philosophical disagreements.

One common objection to this view is “if no one exists, no one will be sad about their non-existence.” I consider this objection very weak. If someone does exist, someone will be happy about their existence. So even if non-existence is not bad, existence is good.

A problem I consider more serious is that this view seems to be very sensitive to the outlook of possible people regarding existence. I would be happy to make my life significantly worse in exchange for a higher probability of existing, but many people feel differently. Moreover, this difference seems to be more about a difference in outlook, rather than a difference in quality of life. This seems particularly problematic given that for nearly any outlook, we can imagine possible people with that outlook. Should we be happy to create any possible people who are enthusiastic about existing, and unhappy to create any possible people who aren’t? What if the actual

character of their experiences were almost the same, and only their attitudes about non-existence differed?

For example, regardless of the actual content of individuals' experiences, we might expect natural selection to produce people who would prefer to exist. So if we think there is *any* kind of experience that would be a net negative, but which could have been produced by natural selection, then we might run into a conflict.

I don't want to get into a long discussion here, but my own best guess is that if someone would prefer existing than to not existing (knowing all of the facts, reflecting appropriately, and so on), then all else equal I would prefer that person to exist. These intuitions become weaker as we move further away from the kinds of minds I am familiar with, and aren't that strong to begin with, but I'll postpone a longer discussion for some future day.

Utilitarianism

Incidentally, though I am a committed consequentialist I am not a hedonistic utilitarian. This perspective sometimes puts me at odds with a few acquaintances. I haven't listed this view above because I think that in the world at large, hedonistic utilitarianism is extremely unusual; however, it is another moral issue where I feel like my view is driven by the golden rule. I have preferences for things other than my own experiences of pleasure and pain, and even if I did not I can recognize that *if* I had other preferences, I would prefer that others respect those preferences.

. . .

Originally published at rationalaltruist.com on August 23, 2014.

Three impacts of machine intelligence



Paul Christiano

[Follow](#)

Nov 29, 2014 · 10 min read

I think that the development of human level AI in my lifetime is quite plausible; I would give it more than a 1-in-4 chance. In this post I want to briefly discuss what I see as the most important impacts of AI. I think these impacts are the heavy hitters by a solid margin; each of them seems like a big deal, and I think there is a big gap to #4.

- **Growth will accelerate, probably very significantly.**
Growth rates will likely rise by at least an order of magnitude, and probably further, until we run into severe resource constraints. Just as the last 200 years have experienced more change than 10,000 BCE to 0 BCE, we are likely to see periods of 4 years in the future that experience more change than the last 200.
- **Human wages will fall, probably very far.** When humans work, they will probably be improving other humans' lives (for example, in domains where we intrinsically value service by humans) rather than by contributing to overall economic productivity. The great majority of humans will probably not work. Hopefully humans will remain relatively rich in absolute terms.
- **Human values won't be the only thing shaping the future.** Today humans trying to influence the future are the only goal-oriented process shaping the trajectory of society. Automating decision-making provides the most serious opportunity yet for that to change. It may be the case that machines make decisions in service of human interests, that machines share human values, or that machines have other worthwhile values. But it may also be that machines use their influence to push society in directions we find uninteresting or less valuable.

My guess is that the first two impacts are relatively likely, that there is unlikely to be a strong enough regulatory response to prevent them, and that their net effects on human welfare will be significant and positive. The third impact is more speculative, probably negative, more likely to be prevented by coordination (whether political

regulation, coordination by researchers, or something else), and also I think more important on a long-run humanitarian perspective.

None of these changes are likely to occur in discrete jumps. Growth has been accelerating for a long time. Human wages have stayed high for most of history, but I expect them to begin to fall (probably unequally) long before everyone becomes unemployable. Today we can already see the potential for firms to control resources and make goal-oriented decisions in a way that no individual human would, and I expect this potential to increase continuously with increasing automation.

Most of this discussion not particularly new. The first two ideas feature prominently in Robin Hanson's speculation about an economy of human emulations (alongside many other claims); many of the points below I picked up from Carl Shulman; most of them are much older. I'm writing this post here because I want to collect these thoughts in one place, and I want to facilitate discussions that separate these impacts from each other and analyze them in a more meaningful way.

Growth will accelerate

There are a number of reasons to suspect that automation will eventually lead to much faster growth. By "much faster growth" I mean growth, and especially intellectual progress, which is at least an order of magnitude faster than in the world of today.

I think that avoiding fast growth would involve solving an unprecedented coordination problem, and would involve large welfare losses for living people. I think this is very unlikely (compare to environmental issues today, which seem to have a lower bar for coordination, smaller welfare costs to avert, and clearer harms).

Automating tech progress leads to fast growth.

The stereotyped story goes: "If algorithms+hardware to accomplish X get 50% cheaper with each year of human effort, then they'll also (eventually) get 50% cheaper with each year of AI effort. But then it will only take 6 months to get another 50% cheaper, 3 months to get another 50% cheaper, and by the end of the year the rate of progress will be infinite."

In reality things are very unlikely to be so simple, but the basic conclusion seems quite plausible. It also lines up with the predictions of naive economic models, on which constant returns to scale (with fixed tech) + endogenously driven technology—> infinite returns in finite time.

Of course the story breaks down as you run into diminishing returns to intellectual effort, and once “cheap” and “fast” diverge. But based on what we know now it looks like this breakdown should only occur very far past human level (this could be the subject for a post of its own, but it looks like a pretty solid prediction). So my money would be on a period of fast progress which ends only once society looks unrecognizably different.

One complaint with this picture is that technology already facilitates more tech progress, so we should be seeing this process underway already. But we do see accelerating growth (see the section on the historical record, below), except for a historically brief period of 50–75 years. So this seems like a weak objection.

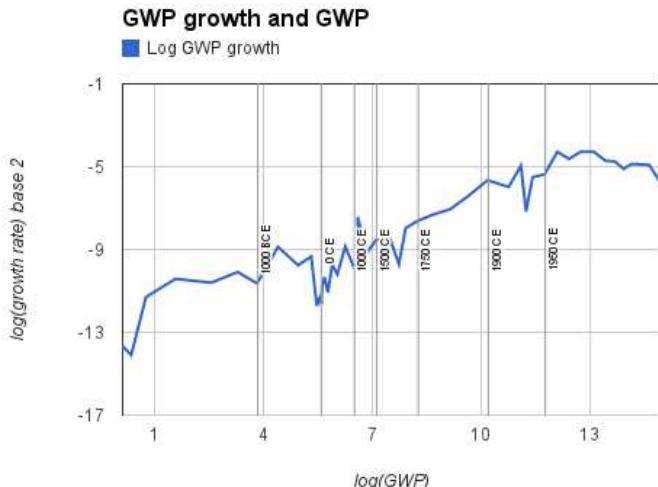
Substituting capital for labor leads to fast growth.

Even if we hold fixed the level of technology, automating human labor would lead to a decoupling of economic growth from human reproduction. Society could instead grow at the rate at which robots can be used to produce more robots, which seems to be much higher than the rate at which the human population grows, until we run into resource constraints (which would be substantially reduced by a reduced dependence on the biosphere).

Extrapolating the historical record suggests fast growth.

Over the course of history the proportional rate of growth has increased substantially, from more than 1,500 years per doubling around 10k years ago, to around 150 years per doubling in the 17th century, to around 15 years per doubling in the 20th century. A reasonable extrapolation of the pre-1950 data appears to suggest an asymptote with infinite population sometime in the 21st century. The last 50 years represent a notable departure from this trend, although history has seen similar periods of relative stagnation.

(See the graph to the left, produced from Bradford Delong's data here; data and full size here. The graph starts at a doubling time of 8000 years, and plateaus around 25 years, about 300 times faster. The average time required for growth rates to double is about 1.5 doublings, or 40 years at the current rate. Each subsequent doubling of the growth rate takes half as long.)



I don't think that extrapolating this trend forward results in particularly robust or even meaningful predictions, but I do think it says one thing: we shouldn't be surprised by a future with much faster growth. The knee jerk response of "that seems weird" is inconsistent with the actual history (though it is a very reasonable intuitive reaction for someone who lived through the 2nd half of the 20th century). The more recent trend of slow growth may well continue, but I don't think we should be surprised if this stagnation was a temporary departure from trend, comparable to previous departures in severity and scale.

Wages will fall

I think this is the most robust of the three predictions. It seems very likely that eventually machines will be able to do all of the things that a human can do, as well as a human can do it. At that point, it would require significant coordination to prevent the broad adoption of machines as human replacements. Moreover, continuing to use human labor in this scenario seems socially undesirable; if we don't have to work it seems crazy to make work for ourselves, and failing to make use of machine labor would involve even more significant sacrifices in welfare.

(Humans may still demand other humans' labor in distinctively human roles. And there may be other reasons for money to move around between humans. But overall most new valuable stuff and valuable ideas in the world will be produced by machines. In the optimistic scenario, the main reason that this value will be flowing *to* humans, rather than merely *amongst* humans, will be because they own some of the machines.)

Historically humans have not been displaced by automation. I think this provides some evidence that in the near term automation will not displace humans, but in the long run it looks inevitable, since eventually machines really will be better at *everything*. Simple theories do suggest a regime where humans and automation are complementary, followed by a regime in which they are substitutes (as horses were once complementary with carriages, but were eventually completely replaced by automation). Robin Hanson illustrates in slides 30–32 here. So at some point I think we should expect a more substantive transition. The most likely path today seems to be a fall in wages for many classes of workers while driving up wages for those who are still complementary with automation (a group that will shrink until it is eventually empty). “Humans need not apply” has been making the rounds recently; and despite many quibbles I think it does a good job of making the point.

I don't have too much to say on this point. I should emphasize that this isn't a prediction about what will happen soon, just about what will have happened by the time that AI can actually do everything humans can do. I'm not aware of many serious objections to this (less interesting) claim.

Human values won't be the only things shaping the future

I'd like to explicitly flag this section as more speculative. I think AI opens up a new possibility, and that this is of particular interest for those concerned with the long-term trajectory of society. But unlike the other sections, this one rests on pretty speculative abstractions.

Many processes influence what the world will look like in 50 years. Most of those processes are not goal-directed, and push things in a somewhat random direction; an asteroid might hit earth and kill us

all, the tectonic plates will shift, we'll have more bottles in landfills because we'll keep throwing bottles away, we'll send some more radiation into space. One force stands out amongst all of these by systematically pushing in a particular direction: humans have desires for what the future looks like (amongst other desires), and they (sometimes) take actions to achieve desired outcomes. People want themselves and their children to be prosperous, and so they do whatever they think will achieve that. If people want a particular building to keep standing, they do whatever they think will keep it standing. As human capacities increase, these goal-oriented actions have tended to become more important compared to other processes, and I expect this trend to continue.

There are very few goal-directed forces shaping the future aside from human preferences. I think this is a great cause for optimism: if humans survive for the long run, I expect the world to look basically how we want it to look. As a human, I'm happy about that. I don't mean "human preferences" in a narrow way; I think humans have a preference for a rich and diverse future, that we care about other life that exists or could have existed, and so on. But it's easy to imagine processes pushing the world in directions that we don't like, like the self-replicator that just wants to fill the universe with copies of itself.

We can see the feeble beginnings of competing forces in various human organizations. We can imagine an organization which pursues its goals for the future even if there are no humans who share those goals. We can imagine such an organization eventually shaping what other organizations and people exist, campaigning to change the law, developing new technologies, *etc.*, to make a world better suited to achieving its goals. At the moment this process is relatively well contained. It would be surprising (though not unthinkable) to find ourselves in a future where 30% of resources were controlled by PepsiCo, fulfilling a corporate mission which was completely uninteresting to humans. Instead PepsiCo remains at the mercy of human interests, by design and by necessity of its structure. After all, PepsiCo is just a bunch of humans working together, legally bound to behave in the interest of some other humans.

As automation improves the situation may change. Enterprises might be autonomously managed, pursuing values which are instrumentally useful to society but which we find intrinsically worthless (e.g. PepsiCo can create value for society even if its goal is merely maximizing its own profits). Perhaps society would ensure that PepsiCo's values are to maximize profit only insofar as such profit-

maximization is in the interest of humans. But that sounds complicated, and I wouldn't make a confident prediction one way or the other. (I'm talking about firms because it's an example we can already see around us, but I don't mean to be down on firms or to suggest that automation will be most important in the context of firms.)

At the same time, it becomes increasingly difficult for humans to directly control what happens in a world where nearly all productive work, including management, investment, and the design of new machines, is being done by machines. We can imagine a scenario in which humans continue to make all goal-oriented decisions about the management of PepsiCo but are assisted by an increasingly elaborate network of prosthetics and assistants. But I think human management becomes increasingly implausible as the size of the world grows (imagine a minority of 7 billion humans trying to manage the equivalent of 7 trillion knowledge workers; then imagine 70 trillion), and as machines' abilities to plan and decide outstrip humans' by a widening margin. In this world, the AI's that are left to do their own thing outnumber and outperform those which remain under close management of humans.

Moreover, I think most people don't much care about whether resources are held by agents who share their long-term values, or machines with relatively alien values, and won't do very much to prevent the emergence of autonomous interests with alien values. On top of that, I think that machine intelligences can make a plausible case that they deserve equal moral standing, that machines will be able to argue persuasively along these lines, and that an increasingly cosmopolitan society will be hesitant about taking drastic anti-machine measures (whether to prevent machines from having "anti-social" values, or reclaiming resources or denying rights to machines with such values).

Again, it would be possible to imagine a regulatory response to avoid this outcome. In this case the welfare losses of regulation would be much smaller than in either of the last two, and on a certain moral view the costs of no regulation might be much larger. So in addition to resting on more speculative abstractions, I think that this consequence is the one most likely to be subverted by coordination. It's also the one that I feel most strongly about. I look forward to a world where no humans have to work, and I'm excited to see a radical speedup in technological progress. But I would be sad to see a future where our descendants were maximizing some uninteresting values

we happened to give them because they were easily specified and instrumentally useful at the time.

• • •

Originally published at rationalaltruist.com on August 23, 2014.

Certificates of Impact



Paul Christiano

[Follow](#)

Nov 29, 2014 · 9 min read

0. Introduction

In this post I describe a simple institution for altruistic funding, based on the creation and exchange of “certificates of impact.”

Typically an effectiveness-minded altruist would try to do as much good as possible. Instead, users of the certificates system try to collect certificates for as much good as possible.

Whenever anyone does anything, they can declare themselves to own an associated certificate of impact. Users of the system treat owning a certificate for X as equivalent to doing X themselves. In the case where certificates never change hands, this reduces precisely to the status quo.

The primary difference is that certificates can also be bought, sold, or bartered; an altruist can acquire certificates through any combination of doing good themselves, and purchasing certificates from others.

For example, a project to develop a malaria vaccine might be financed by investors. If the project succeeds the developers can declare themselves owners of an associated certificate, and distribute it to investors in the same way they might distribute profits. These investors could then resell this certificate to a philanthropist who is interested in funding malaria prevention and who honors certificates. The philanthropist would recognize the certificate as valuable to the extent that they believed that the original project was causally responsible for the development of the vaccine, and their willingness to pay would be the same as their willingness to develop the vaccine themselves (evaluated with the benefit of hindsight).

Note that judging the value of a certificate is just as subjective as judging the value of the underlying activity, and is done separately by any philanthropist considering acquiring that certificate.

1. Some existing challenges

1. Evaluating philanthropic opportunities is difficult, and most are not very good. Predictions are expensive and inaccurate, and often infeasible for small donors. The problem is worst when evaluating novel interventions. Prizes can address some of these difficulties, but have their own set of problems.
2. One person might have the ability to do a project, another the desire to fund it, and a third the knowledge to evaluate it. Coordination is hard, and existing incentives misaligned, both for disseminating good information and executing good projects.
3. Thinking about crowding out and causal responsibility is hard: if I don't do something, will someone else? Does it matter which of Oxfam's activities I fund? How is causal responsibility divided between donors and employees? Between the philanthropist who buys malaria nets and the government which distributes them?
4. Prices are an elegant and flexible system for communication and coordination, but they are often unavailable for altruists.
5. Reasoning about leverage, and interacting with funders with different priorities, is tough. It's easy to end up neglecting a large possible upside or engaged in zero-sum conflict.

2. Certificates of impact

Whenever anyone does something good in the world, they can “mint” an associated certificate of impact and declare themselves to be the owners. This can be as simple as making a public statement, analogous to acknowledging a funder. At the bottom of this post I could write “This post backs certificate [RationalAltruist-22], which is now owned by Paul Christiano;” I could do the same in a press release describing a medical intervention, an academic paper reporting the results of an experiment, or a piece of open source software. Independently, I might post an offer to sell these certificates, or a philanthropist might contact me and express interest in purchasing them.

By deciding which certificates they are willing to pay for, altruists using the system define the rules of the game. The protocol suggests considering a certificate valuable only if

1. It was issued by the group or individual that performed the associated activity.

2. It is the unique certificate associated with that activity.

Users should be willing to pay the same amount for a valid certificate of X that they would be willing to pay to cause X to happen (and to keep the resulting certificate). That is, I should be indifferent between spending \$100 to do a good deed and paying someone else \$100 to do it, provided that in either case I get the entire certificate for that good deed.

Naturally this willingness to pay is subjective, depending on both values and views—just like conventional grantmaking. Users maximize the total value of all of the certificates they hold, using whatever combination they prefer of doing good deeds themselves or acquiring others' certificates.

Certificates can be transferred just like any other IOU or title, and have the same practical issues with bookkeeping and double-spending. We can resolve these however we like (word of mouth, a bulletin board, a trusted central party, the bitcoin blockchain...). Establishing properties [1] and [2] probably involves trusting the issuer—just like conventional grantmaking.

Certificates can be subdivided arbitrarily. When a group accomplishes a project together, they must decide how to divide the resulting certificates in the same way that a profitable enterprise would decide how to divide the profits.

Owning a certificate for X would probably not carry the same prestige as doing X yourself. Funders using the system purchase certificates as a way to do good, not necessarily as a way to earn credit (though helping them get credit may be useful for getting them on board).

For now, I'm most interested in the question: would philanthropic activity in an area be improved, if many funders used this system or a similar scheme?

3. Commentary

Some simple examples

No transfers. In the simplest possible example, certificates never change hands. In this case, the protocol reduces to “Do the activities that you think have the largest positive impact on the world,” i.e. the status quo.

Grants. A grantmaker might pay a charitable organization to perform some charitable work, in return for the resulting certificates. This also reduces to the status quo, at least for the grantmaker. A grantmaker might also leave some of the certificates to the charities, in the same way that a startup retains equity as an incentive.

Prizes. An unfunded do-gooder might keep the resulting certificates to themselves. Later, a philanthropist who appreciates the value of their work could buy the certificate off of them. The end result is very similar to a prize or bounty. As with a prize or bounty, that philanthropist could state their intentions in advance, choosing their own balance between flexibility and predictability.

So what?

Reproducing the status quo isn't so exciting. Do certificates improve the situation, or even change it? I think so, in ways that should be interesting to particularly effectiveness-minded funders. For example:

1. Allocating certificates requires explicit and transparent allocation of causal responsibility, both within teams and between teams and donors. In addition to the obvious cultural effects, this aligns individual incentives with altruistic goals, reducing incentives to mislead donors, volunteers, and employees. Overall I would be excited about these effects, though there are *significant* costs.
2. Exchanging certificates for money leads to a more consistent conversion between good done and compensation, especially if funders use a mix of prizes and grants. It can prevent good deeds from falling through the cracks, ameliorate some winner-take-all PR dynamics, and eliminate some zero-sum conflict. It also makes it easier to move between different funding modes.
3. Purchasing certificates helps funders with different values coordinate; if I see a good deal I have an incentive to take it, without worrying about whether it might be an even better deal for someone else.
4. The ability to resell certificates makes a purchase less of a commitment of philanthropic capital, and less of a strategic decision; instead it represents a direct vote of confidence in the work being funded.

There are also risks inherent in bringing altruistic incentives closer to profit incentives (substituting extrinsic for intrinsic motivations, introducing new scope for zero-sum conflict...). How you feel about the tradeoff depends on how you feel about the relative merits of the two approaches, and may vary across domains.

Some more examples

In addition to these simple examples, certificates of impact can facilitate more exotic interactions.

Financing. If I'm optimistic about your project I might give you a loan which I expect you to pay back by selling a certificate. Or I could buy the certificates off you, as if I were a grantmaker, with the intention of reselling them.

Research universities. A research group can employ researchers in exchange for ownership of some of the certificates of impact they produce. These certificates can then be sold to funding agencies or philanthropists. This shields those funders from having to make predictions about research output, which they are often not equipped to do, and provides good incentives to both researchers and research groups. (This is a special case of financing and causal attribution, but it's an application close to my heart.)

Interpolating between prizes and grants. By implementing both prizes and grants as trades, it is easier to interpolate between the two, providing some funding to get a project going in exchange for part of its output (presumably at an expected discount) and then purchasing more of its output after the fact.

Moral trade. Many cases of moral trade can be implemented by turning good deeds into certificates and bartering or trading in a marketplace. For others, such as abstaining from eating meat, this would be problematic (though it's not clear that the use of certificates is to blame).

Targeted donations. If I want to fund only part of an organization's activities, I can purchase only those certificates of impact which I find valuable.

Research on effectiveness. If I think a certificate is too cheap, I can do research to evaluate its effectiveness; depending on the result I can buy the certificate, publish my research, and resell the certificate.

Trades across time. If my discount rate is lower than interest rates, I can sell certificates of impact today, invest the proceeds, and purchase certificates next year. (The price of each fixed certificate should rise at roughly market rates; but in addition to that, a certificate for a life saved in year 1 may have a different value than a certificate for a life saved in year 2, reflecting the discount rates of philanthropists.)

Speculation. If I think a change will be considered good in retrospect, I can purchase appropriate certificates now and aim to resell them later at a profit.

Multiple bottom lines. If a project has several objectives, it can use certificates to monetize all of them and then make local profit-maximizing decisions. This may lead to more sane decision-making, and much more transparency, than the status quo.

(I'm sparing you a much longer list.)

Some theory

Suppose that everyone who valued X were using the certificate protocol. (For concreteness, suppose it's everyone interested in cosmology research.)

Then at equilibrium, the price of certificates of impact on X is equal to the marginal cost of achieving an impact on X. Any deviation is an arbitrage opportunity: if you can do X more cheaply, then you can sell the resulting certificates for a profit; if you are doing X more expensively, then you could save money by buying certificates instead.

Moreover, the status quo corresponds to one way of producing certificates, with total value equal to the total good being done. The actual outcome should be a Pareto improvement, and in particular should increase the total value of available certificates. As a result, it also increases the total amount of good being done.

I wouldn't lean on this argument alone. But I do think it suggests that things might tend to work themselves out, at least if the system were widely adopted. For me, considering concrete cases bolsters that intuition.

The infra-marginal units of X, those that were cheaper to produce than the marginal unit, pose a theoretical problem. These units of X

get produced and purchased (good news!), but the resulting certificates are most likely to be sold for the marginal price of X (bad news!). That means there is a transfer of wealth, from the people who care about X (and who use the certificates system) to those who have infra-marginal opportunities to do X.

This problem is worst for the early adopters of the certificate system, which is particularly unfortunate. And it's even worse when the opportunities to do X are so cheap that they were going to get done anyway. For example, suppose some high energy physicists would have to go out of their way to avoid doing some cosmology. Under the certificates system they would get paid for this research by cosmology funders who purchase the resulting certificates. But the cosmology funders might not be so happy about this, since the counterfactual impact of this funding is zilch.

Whether this is a feature or bug depends on how many really cheap opportunities to do X were failing to get done. Maybe the high energy physicists wouldn't even bother publishing their cosmology in a format that cosmologists would understand, because the high energy community wouldn't give them any respect for it. In this case there might be huge social benefits from paying them to do so. My own guess is that we leave a lot of low hanging fruit hanging, and that picking it is more important than worrying about the transfers.

-1. Note

I have omitted any discussion of how certificates might be implemented, especially discussion of how you might get from here to there. This is a tough problem, but I think that a single advocate could make meaningful headway if it seemed worthwhile. I'm interested in the prior question, is it a "there" worth thinking about?

• • •

Originally published at rationalaltruist.com on November 15, 2014.



Paul Christiano

[Follow](#)

Dec 1, 2014 · 18 min read

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 definitions 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 accommodate 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 thinks 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.

Of arguments and wagers



Paul Christiano

[Follow](#)

Dec 14, 2014 · 7 min read

(In which I explore an unusual way of combining the two.)

Suppose that Alice and Bob disagree, and both care about Judy's opinion. Perhaps Alice wants to convince Judy that raising the minimum wage is a cost-effective way to fight poverty, and Bob wants to convince Judy that it isn't.

If Judy has the same background knowledge as Alice and Bob, and is willing to spend as much time thinking about the issue as they have, then she can hear all of their arguments and decide for herself whom she believes.

But in many cases Judy will have much less time than Alice or Bob, and is missing a lot of relevant background knowledge. Often Judy can't even understand the key considerations in the argument; how can she hope to arbitrate it?

Wagers

For a warm-up, imagine that Judy could evaluate the arguments if she spent a long enough thinking about them.

To save time, she could make Alice and Bob wager on the result. If both of them believe they'll win the argument, then they should be happy to agree to the deal: "If I win the argument I get \$100; if I lose I pay \$100." (Note: by the end of the post, no dollars will need to be involved.)

If either side isn't willing to take the bet, then Judy could declare the case settled without wasting her time. If they are both willing to bet, then Judy can hear them out and decide who she agrees with. That person "wins" the argument, and the bet: **Alice and Bob are betting about what Judy will believe, not about the facts on the ground.**

Of course we don't have to stick with 1:1 bets. Judy wants to know the probability that she will be convinced, and so wants to know at what

odds the two parties are both willing to bet. Based on that probability, she can decide if she wants to hear the arguments.

It may be that both parties are happy to take 2:1 bets, i.e. each believes they have a 2/3 chance of being right. What should Judy believe? (In fact this should always happen at small stakes: both participants are willing to pay some premium to try to convince Judy. For example, no matter what Alice believes, she would probably be willing to take a bet of \$0.10 against \$0.01, if doing so would help her convince Judy.)

If this happens, there is an arbitrage opportunity: Judy can make 2:1 bets with both of them, and end up with a guaranteed profit. So we can continuously raise the required stakes for each wager, until either (1) the market approximately clears, i.e. the two are willing to bet at nearly the same odds, or (2) the arbitrage gap is large enough to compensate Judy for the time of hearing the argument. If (2) happens, then Judy implements the arbitrage and hears the arguments. (In this case Judy gets paid for her time, but the pay is independent of what she decides.)

Recursion

Betting about the whole claim saved us some time (at best). Betting about parts of the claim might get us much further.

In the course of arguing, Alice and Bob will probably rely on intermediate claims or summaries of particular evidence. For example, Alice might provide a short report describing what we should infer from study Z, or Bob might claim “The analysis in study Z is so problematic that we should ignore it.”

Let’s allow anyone to make a claim at any time. But if Alice makes a claim, Bob can make a counterclaim that he feels better represents the evidence. Then we have a recursive argument to decide which version better represents the evidence.

The key idea is that **this recursive argument can also be settled by betting**. So one of two things happens: (1) Judy is told the market-clearing odds, and can use that information to help settle the original argument, or (2) there is an arbitrage opportunity, so Judy hears out the argument and collects the profits to compensate her for the time.

This recursive argument is made in context: that is, Judy evaluates which of the two claims she feels would be a more helpful summary within the original argument. Sometimes this will be a question of fact about which Alice and Bob disagree, but sometimes it will be a more complicated judgment call. For example, we could even have a recursive argument about which wording better reflects the nuances of the situation.

When making this evaluation, Judy uses facts she learned over the course of the argument, but she interprets the claim as she would have interpreted it at the beginning of the argument. For example, if Bob asserts “The ellipsoid algorithm is efficient” and Alice disagrees, Bob cannot win the argument by explaining that “efficient” is a technical term which in context means “polynomial time”—unless that’s how Judy would have understood the statement to start with.

This allows Judy to arbitrate disagreements that are too complex for her to evaluate in their entirety, by showing her what she “would have believed” about a number of intermediate claims, if she had bothered to check. Each of these intermediate claims might itself be too complicated for Judy to evaluate directly—if Judy needed to evaluate it, she would use the same trick again.

Betting with attention

If Alice and Bob are betting about many claims over the course of a long argument, we can replace dollars by “attention points,” which represent Judy’s time thinking about the argument (perhaps 1 attention point = 1 minute of Judy’s time). Judy considers an arbitrage opportunity “good enough” if the profit is more than the time required to evaluate the argument. The initial allocation of attention points reflects the total amount of time Judy is willing to spend thinking about the issue. If someone runs out of attention points, then they can no longer make any claims or use up any of Judy’s time.

This removes some of the problems of using dollars, and introduces a new set of problems. The modified system works best when the total stock of attention points is large compared to the number at stake for each claim. Intuitively, if there are N comparable claims to wager about, the stakes of each should not be more than a $1/\sqrt{N}$ of the total attention pool—or else random chance will be too large a factor. This requirement still allows a large gap between the time actually required to evaluate an argument (i.e. the initial bankroll of attention

points) and the total time that would have been required to evaluate all of the claims made in the argument (the total stake of all of the bets). If each claim is itself supported by a recursive argument, this gap can grow exponentially.

Talking it out

If Alice and Bob disagree about a claim (rather, if they disagree about Judy's probability of accepting the claim) then they can have an incentive to "talk it out" rather than bringing the dispute to Judy.

For example, suppose that Alice and Bob each think they have a 60% chance of winning an argument. If they bring in Judy to arbitrate, both of them will get unfavorable odds. Because the surplus from the disagreement is going to Judy, both parties would be happy enough to see their counterparty wise up (and of course both would be happy to wise up themselves). This creates room for positive sum trades.

Rather than bringing in Judy to arbitrate their disagreement, they could do further research, consult an expert, pay Judy attention points to hear her opinion on a key issue, talk to Judy's friends—whatever is the most cost-effective way to resolve the disagreement. Once they have this information, their betting odds can reflect it.

An example

Suppose that Alice and Bob are arguing about how many trees are in North America; both are experts on the topic, but Judy knows nothing about it.

The easiest case is if Alice and Bob know all of the relevant facts, but one of them wants to mislead Judy. In this case, the truth will quickly prevail. Alice and Bob can begin by breaking down the issue into "How many trees are in each of Canada, the US, and Mexico?" If Alice or Bob lie about any of these estimates, they will quickly be corrected. Neither should be willing to bet much for a lie, but if they do, the same thing will happen recursively—the question will be broken down into "how many trees are east and west of the Mississippi?" and so on, until they disagree about how many trees are on a particular hill—a straightforward disagreement to resolve.

In reality, Alice and Bob will have different information about each of these estimates (and geography probably won't be the easiest way to break things down—instead they might combine the different

considerations that inform their views, the best guess suggested by different methodologies, approximate counts of each type of tree on each type of land, and so on). If Alice and Bob can reach a rational consensus on a given estimate, then Judy can use that consensus to inform her own view. If Alice and Bob can't resolve their disagreement, then we're back to the previous case. The only difference is that now Alice and Bob have probabilistic disagreements: if Alice disagrees with Bob she doesn't expect to win the ensuing argument with 100% probability, merely with a high probability.

Odds and ends

This writeup leaves many details underspecified. In particular, how does Judy estimate how long it will take her to arbitrate a disagreement? This can be handled in several ways: by having Judy guess, by having Alice and Bob bet on the length of time until Judy reaches a conclusion, by having them make bets of the form "Alice will agree with me with Z effort," or so on. I don't know what would work best.

Despite my use of the word "recursion," the estimate for "time to settle an argument" (which Judy uses to decide when the stakes are high enough to step in and resolve a disagreement) probably shouldn't include the time required to settle sub-arguments, since Judy is being paid separately for arbitrating each of those. The structure of the arguments and sub-arguments need not be a tree.

This is a simple enough proposal that it can be realistically implemented, so eventually we'll hopefully see how it works and why it fails.

I expect this will work best if Alice and Bob often argue about similar topics.

This scheme was motivated by a particular exotic application: delegating decision-making to very intelligent machines. In that setting the goal is to scale to very complex disagreements, with very intelligent arguers, while being very efficient with the overseer's time (and more cavalier with the arguers' time).

Delegating to a mixed crowd



Paul Christiano [Follow](#)

Dec 15, 2014 · 3 min read

Suppose I have ten programs, each a human-level agent. I suspect that at least one or two of them will behave helpfully, but the others may behave maliciously. How can I get useful work out of the combination?

Here's a very rough proposal:

- Each agent receives an initial budget of points.
- By default, each agent runs slowly in an independent environment. An agent may pay points to receive extra computational resources.
- At any time, an agent may propose an action and claim that I would approve of it—perhaps it's an instruction to give to a robot, a message to display to me, or a modification to this top-level loop. They also estimate how much I'd be willing to pay them for the time they spent thinking up the action.
- Any other agent can challenge the proposed action, or the estimate for how much I'd be willing to pay. If so, the actor and challenger have an argument about it, which I arbitrate.
- Running arguments is the key ingredient of the proposal: I need to arbitrate arguments which are potentially (1) extremely frequent, (2) extremely complicated, and (3) between extremely smart people. And I want to do it without spending almost any of my time. I think that an idea along these lines might work, where agents wager points about what we will believe if we hear them out, and we concentrate our attention on claims where there is significant disagreement.
- If there are no successful challenges, the action is taken and the actor is compensated. If there is a successful challenge, the action is not taken and the challenger is compensated instead, at the expense of the actor (we can use a similar argument to determine the extent of the compensation). In addition to these payments, many points may be won or lost over the course of the argument itself.

Would this work? I don't know. I'd guess that it would work iff the arguments worked; if the arguments work correctly and the malicious agents try to significantly interfere with the helpful agents, the helpful agents will probably be able to extract points from them.

Odds and ends

Details

This game is still underspecified; many details would have to be filled in, and many tweaks and elaborations would be required to get something sensible.

We can use the system itself to address the question "how should the system work?" For example, if the honest agent realizes that the setup is subject to manipulation, she can immediately recommend a change to fix the problem. As long as her initial budget is large enough to convince us that we should change the system, the problem can correct itself.

Why not go straight for the throat?

The most important issue would be understanding which of the agents actually want to be helpful and why. If the argument system is working well, it might be used to address this problem directly; and once we knew which agent actually wanted to be helpful, we could get rid of the others.

So why do I care about the mixed-crowd setting at all?

1. I think that an implementation of the transform $(\text{helpful} + ?) \rightarrow (\text{helpful})$ may be a useful ingredient in building (helpful) to start with. For example, we might apply similar ideas at a lower level or iterate the proposal to "amplify" slightly helpful agents into very helpful agents.
2. This procedure can be used to define a computation (where the "actions" are all computational steps) which "I would approve of" as a method to determine which agents want to be helpful. Defining a process to determine X is generally easier than determining X directly, so this places lower demands on the argument system.
3. Determining what agents want to be helpful may prove to be a particularly nasty problem. It may be that none of the agents reliably want to be helpful across contexts, or that there is no

easy way to see that any agent is motivated to be helpful (even if it is), or so on. We might want to have helpful behavior in the meantime.

Delegating to different crowds

In principle we could also delegate to agents who only care about collecting as many points as they can. But in the self-interested case, collusion becomes a significant concern.

We could apply the same idea to delegate to people. The first step would be understanding whether and how the argument scheme can work well enough (with people) to support an application like this.

Optimization and goals



Paul Christiano [Follow](#)

Dec 28, 2014 · 7 min read

If we want to write a program that *doesn't* pursue a goal, we can have two kinds of trouble:

1. We might need to explicitly introduce goal-directed behavior into our program, because it's the easiest way to do what we want to do.
2. We might try to write a program that doesn't pursue a goal, and fail.

Issue [2] sounds pretty strange—it's not the kind of bug most software has. But when you are programming with gradient descent, strange things can happen.

In this post I illustrate issue [2] by considering a possible design for an approval-directed system, intended to take individual actions the user would approve of without considering their long-term consequences.

The architecture

Our agent takes as input a sequence of observations $x[t] \in \{0, 1\}$ and outputs a sequence of actions $a[t] \in \{0, 1\}$.

The main ingredient is a pair of functions $A : \Re^n \rightarrow \{0, 1\}$, $S : \Re^n \times \{0, 1\} \rightarrow \Re^n$. To run the agent:

- We initialize $s[0] = o^n$.
- Define $a[t] = A(s[t])$
- Define $s[t+1] = S(s[t], x[t])$

For concreteness you might imagine that A and S are feed-forward neural networks, so that the entire system is a typical recurrent neural network. I'm going to imagine performance well beyond anything we can currently get out of a recurrent neural network.

Our training data consists of some sequence of observations $x[t]$ and reward functions $r[t] : \{0, 1\} \rightarrow \mathfrak{R}$. We train A and S to maximize $\Sigma r[t](a[t])$. We'll define $r[t](a)$ to be how much we approve of taking a in the context of step t .

To gather training data, we use the current version of A and S to control a robot, we define $x[t]$ as the robot's sensor reading at step t , and we manually provide the reward functions $r[t]$ as look-up tables.

We hope that if we use A and S to control a robot, the resulting system chooses each action to maximize our approval *of that action*. (This might seem like a crazy design, but for now let's just ask whether the resulting system will exhibit goal-directed behavior.)

The state s is important for most applications. We'd like our agent to make decisions based on everything it knows, not based only on what it currently observes. By optimizing S to store useful information, we allow the system to remember information across time steps and aggregate many observations to make informed judgments.

What happens?

This system looks like it shouldn't form any long-term plans; each action should be selected greedily, without concern for its consequences. The separation of A and S makes this perfectly explicit: A is essentially a classifier, which labels each state s with the most-approved action $a \in \{0, 1\}$. There isn't any room for A to consider long-term consequences.

How would this system work, if it worked well? What would the learned function S look like?

Here are some possible properties, which seem like the “right” way to optimize the reward function. That is, if you optimized well enough, over a large enough space of functions, I would expect you to observe these behaviors:

1. The state s encodes beliefs about the agent's environment.
(Assuming that our approval, and hence the reward function r , is mostly determined by the environment.)
2. The function S updates the agent's current beliefs based on the observation x .
3. s selectively remembers facts that are likely to be useful, because it doesn't have enough space to remember everything in detail. It

also stores the results of cached computations—the function A doesn't have time to do much computation, so if complicated computations are useful, the agent must carry them out over many steps and remember the results. (For example, if the agent saw a puzzle written on a sheet of paper, the agent might compute and store its answer to use later.)

4. S uses non-trivial rules to decide what to remember or compute. All of these decisions are at least implicitly optimized to make efficient use of limited resources. These choices depend on what situations will arise in the future, and so S should use its full predictive ability to make them. For example, when deciding whether to forget X or forget Y, S might predict what reward it will receive in each case and choose the better option (this computation could use cached intermediate results and predictions, implicitly taking place over many steps). **In general, S updates s rationally to maximize its expected future reward.**
5. The agent is able to reason about the relationship between its own beliefs and the world. For example, it might realize that a particular fact will be repeated later, and so not bother remembering it. This clearly requires making a prediction about what it will observe in the future. But it's also based on a prediction about how S will update s in response to that observation, and about how A will respond to different values of s.

These all sound like good things. Each of them leads the agent to perform better when deployed. We could imagine accomplishing the same goals with simple heuristics, but as we get enough training data the heuristics should converge to rational behavior.

But these properties, and especially property 4, may have some counterintuitive consequences.

We can easily see the shadow of plans: when the agent chooses the state s, it won't greedily maximize the next round's performance (and we wouldn't want it to, since this would mean promptly forgetting everything that isn't immediately relevant). Instead, it will choose a state s that is useful for long-term performance. And to do that, it will think about long-term consequences.

What goes wrong?

When the agent anticipates the consequences of choosing different values of $s[t]$, it sees that different choices lead to different sequences of states, hence different sequences of predictions, hence different rewards. During training the agent *doesn't* expect different states to lead to different events in the world, because it was trained on the fixed actions taken by previous versions of A,S.

But when the system is actually deployed, changing s will also change what action is actually taken. If the system is a good general purpose learner, it will notice this relationship (along with many other patterns that happen to appear while the system is deployed but didn't appear in any training episode.)

For example, if the agent forgets where an item is and is unable to find it when asked, the agent is likely to be reminded where to look. In general, the agent should notice patterns like "If I forget X, I am more likely to be reminded of X in the future." Once the agent notices this pattern, the agent should act on it—after all, the agent is already acting on many similar patterns.

If the agent rationally chooses s to maximize future rewards, it might take more extreme measures. For example, it might reason "If I choose a state s which incorrectly 'remembers' that the user asked me to write down X, then I will write down X. If I write down X, I will remember X in the future. This will lead to low reward in the next few rounds, since the user *didn't* ask me to write down X. But this is more than offset by the high future reward from remembering X on many different occasions."

In an even more extreme case, the agent might reason "If I choose a state s which incorrectly 'remembers' that the user asked me to implement another AI whose goal is maximizing my future reward, I will achieve a high future reward. I will get an approval of o while I am implementing the new AI, but this will be more than offset by my high reward thereafter."

This example is of course facetious, but hopefully it illustrates the point. If we actually want the robot to maximize its total approval, then we should be happy with this result. But now we are right back in the goal-directed case.

Upshot

If you trained a sufficiently weak S, it would not display goals nor exhibit human-level reasoning. If you trained a sufficiently powerful

S, it might display goals and would definitely be superhuman. Somewhere in between you get human-level performance, and somewhere in between you probably get goal-directed behavior.

It's not clear whether goal-directed behavior emerges before or after human-level performance. But I am wary of systems that only behave as intended in a sweet spot between being "good enough" and "too good," or which depend on optimistic assumptions. It would be more satisfying to design systems that perform robustly regardless of how smart they are.

I suspect that similar problems will arise frequently if we train very intelligent but goal-free systems.

One response is to give up on designing intelligent systems without goals—to conclude that in order to build systems that work robustly, we should conservatively assume that they have goals. This is probably somewhat premature.

Another response is to find a better training criterion, which doesn't involve maximizing a sum of future rewards. But it's not clear how to do this while also training the system to build a useful representation, since the usefulness of a representation is determined by long-term performance. We could train F to create a state s which is useful on the very next time step (i.e. we could propagate gradients from $s[n+1]$ to the parameters of S, but not back to $s[n]$), and hope that the resulting update rule happens to be appropriate over longer time scales. But this seems to rest on a pretty optimistic assumption.

A third response is to also update the state s in the most-approved-of way. But this forces the human engineers to figure out how the system should store its knowledge, which may be a serious problem (though it might be manageable).

A final possible response is to "cross that bridge when we come to it." I would like to understand AI safety further in advance, and understanding the inevitability of goal-directed behavior is a natural step. So although this option is tempting, I am hesitant to accept it. Our theoretical understanding is also so weak that I feel we can go somewhat further before we run into a hard requirement for empirical feedback. I'll find this option more tempting once we are going in circles rather than making steady headway.

Adversarial collaboration



Paul Christiano

[Follow](#)

Dec 29, 2014 · 6 min read

Suppose that I have hired a group of employees who are much smarter than I am. For some tasks it's easy to get useful work out of them. For example, suppose I am interested in finding a good layout for the components on a chip and can easily evaluate the quality of a proposed layout. Then I can solicit proposals, test the proposals, and award the employees according to the results.

However, for some tasks there may be fundamental problems with such an incentive-based policy. For example, suppose I am interested in hiring these employees to design an AI which is smarter than any of us and will act autonomously in the world. In cases like this, I will have to try something different.

The challenge

In this case there is no clear way to evaluate the performance of my employees even *ex post*. I can build whatever AI they propose and test its performance in a range of environments. But these tests could never incentivize my employees to make an AI that shared *my* values, rather than one that shared *their* values: an AI that shared their values would be motivated to maximize the reward given to my employees—including looking *as if* it shared my values, as long as I was watching.

In this context, rather than blindly testing proposals I should understand them first, and reward my employees for making proposals which I can understand and which are clearly good. Even if I can't think of a way to design a better AI, I may be able to recognize a good idea when I see one.

But this approach focuses my employees' ingenuity on finding proposals which will be *convincing to me* rather than *actually good*. I can try to remove any other incentives for my employees, but even then I should worry that most compelling ideas aren't actually good. This becomes increasingly problematic as the ideas become more complicated, and subtle problems become more and more likely.

Maybe we could just be so careful that it didn't matter; we could hold a proposed idea to such a high standard that if it passes, it really must be good. In practice, this could require so much vigilance that it would be harder than coming up with the idea yourself.

The idea

Instead you could consult not one but two groups of employees: team A, whose goal is to provide an idea that you think is good, and team B, whose goal is to convince you that it isn't. This might be a good idea even if you do have a clear performance metric, to help identify its possible shortcomings, efforts to game the system, or unintended consequences of an idea.

(Parts of this proposal are based on discussion with Katja Grace. If you haven't figured it out by now, the whole thing is a metaphor for the AI control problem.)

The rest of this post will describe some issues with and elaborations on this proposal.

Capabilities: Ideally team B would be smarter, know more, and have longer to deliberate than team A. How much smarter? The tradeoff depends on the relative costs of taking longer vs. incorrectly accepting a bad idea. If an idea is good enough, then we might hope it will look good no matter how smart team B is.

In particular, it would be great if team B could know everything that team A knows, and could eavesdrop on as much of their thought process as possible. This would make it very difficult for team A to be deliberately misleading. In the case of machine intelligences, it might be possible for the two teams to literally share the same mental state.

Weakening the critic: It may be *too easy* to convince someone that an idea is flawed. If team B is much smarter than team A, then they may always be able to find some reason that team A's suggestion is suboptimal. I don't think this is a big deal. We can:

- Allow several rounds of rebuttals, in which team A can defend their idea against criticisms, or can change their ideas to address flaws.
- Force team B to find significant flaws. If team B can improve at all on team A's plan, that suggests that team A's proposal isn't

the best; but at some point we should just accept it anyway if team B can't find serious problems.

It's worth remembering that team B might be motivated to try any trick at their disposal to convince us that the proposed idea is no good. They will claim that the problems are serious but too subtle to easily explain, will make desperate emotional appeals, will try to scare or threaten the implementer, and so on. To cope with this, we might call some outcomes a tie—we could refrain from using the idea, but still decline to give team B a reward.

Problematic incentives: Ulterior motives for either team could create problems. Team B might go soft on an idea which they want us to accept (and knowing this team A might propose an idea that team B will like). So we should ensure the incentives are large enough that both teams care about winning much more than they care about whether we accept their ideas.

In particular, if we are going to do multiple rounds of back-and-forth, we should use independent teams. Team A shouldn't be able to get more reward if their original plan is shot down. Instead we should hire a new team A'.

Preemptively defend against criticisms?: we could set up the game in two extreme ways.

- Team A is rewarded when they propose a plan which sounds good. This may lead to many rounds of rejections, as team A constantly makes good-sounding but flawed proposals, since team A has no motivation to fix hard-to-spot flaws.
- Team A's goal is to make a plan which survives scrutiny.

The first option is better, if it works. If team B can't find any holes in a generic idea that sounds good to us, we should feel pretty comfortable. We should be (slightly) less comfortable if team A engineered their plan so that team B couldn't explain its problems to us.

One could iterate the first option several times, with each round leaving us increasingly savvy to possible flaws in team A's plan. Eventually this starts to run into the same concerns with overfitting.

Team C: We could try to capture the best of both worlds by introducing a third team C, which is better-equipped than either team

A or team B and operates in one of two ways:

- Team A attempts to produce plans that will survive criticism by team B. After a proposal survives criticism by team B, team C enters and attempts to identify further flaws. Team A does not care whether or not its plan is overturned by criticism by team C.
- Team A attempts to produce plans that will be superficially plausible, without concern for whether they will survive criticism by B. Before team A makes a suggestion, team C tries to provide coaching which minimizes our probability of inadvertently accepting a “bad” plan. That is, team C is penalized if we accept a plan from team A which is later overturned by criticism from team B.

Other mechanisms: this approach could be combined with other mechanisms designed to get useful work out of much smarter employees. For example:

- Bootstrapping: ask smart employees how to build more effective systems for getting information out of smart employees.
- Throttling: work with employees who are smarter than you but only by the minimum amount necessary to get the job done.
- Restricting the range of proposals: limit team A’s options, and stick with what we can most easily understand.
- Self-verifying proposals: ask for ideas that help verify their own desirability. (Some ideas can fail in subtle other ways; others are demonstrably bad if they are bad at all. Encourage team A to look for the second kind of ideas.)
- Ex post verification: we can give out the reward only after we have seen an idea in action. This ex post verification can be assisted by team B, who can help us design good tests and think of possible shortcomings we wouldn’t have looked for.

None of the ideas in this post are silver bullets. And if they were needed, the would-be employer would hopefully spend more time thinking about the problem than I have—and they would have more contextual information.

The point of this post is to help flesh out our understanding of how hard it is to delegate high-stakes problems to very smart, very

cutthroat reward-seekers. The bigger goal is to better understand how hard the AI control problem is (and where the largest difficulties lie).

• • •

Originally published at ordinaryideas.wordpress.com on July 18, 2014.

Implementing our considered judgment



Paul Christiano

[Follow](#)

Jan 18, 2015 · 14 min read

Suppose I had a very powerful prediction algorithm. How might I use this algorithm to build a smart machine that does what I want?

If I could implement a function J which returned the “right” answer to any question, I could build a machine which computed $J(\text{“What should the machine do?”})$ and did that. As far as I’m concerned, that would be good enough.

It’s far from clear what the “right” answer is, but I’m going to describe an algorithm that tries to find it anyway.

(I’m actually happy to settle for a very good answer, rather than the best one. If we could build a very good machine—one which is much more productive than any combination of humans, say—then we could ask it to design a very, very good successor.)

The setup

I’ll assume I have a function $\text{Predict}()$ that takes as input a sequence of observations and predicts the next one.

I’ll assume that $\text{Predict}()$ predicts *very* well. In the future I’ll talk about scaling down—using a mediocre predictor to get a mediocre version of the “right” answer—but for now I’ll start with a strong assumption and end with a strong conclusion.

I’ll assume that $\text{Predict}()$ can be physically instantiated, and can make reasonable predictions about itself. (It’s some work to formalize this, but there are no intrinsic problems with self-reference. If I ask $\text{Predict}()$ to predict what $\text{Predict}()$ won’t predict, it will just output a uniform distribution.)

The notion of “observation” is going to become a bit confusing, so let’s be clear: in the background is some (approximate, implicit) prior over binary sequences x . $\text{Predict}(x[:n])$ outputs a sample from the (approximate) posterior distribution of $x[n]$ given the values of $x[0], x[1], \dots, x[n-1]$.

I'll assume we have some source of randomness that is unpredictable to `Predict()`. This is a very mild assumption. For example, we can use `Predict`'s self-predictions to generate pseudorandomness.

Finally: I'm going to write a program to answer questions, but I'm actually going to have to interact with it each time it answers a question. This may look like a serious drawback, but it's actually quite mild. If I'm tired of making decisions, I can use `Predict()` to lighten the load. Every time it answers a question, it asks for my help with probability 1%. With probability 99%, it just predicts what I would have said if I had helped. (Similar tricks can lower my workload even further.)

The ideal

In this section, I'll describe a formalization of "considered judgment." In the next section, **The implementation**, I'll describe how to use `Predict()` to find our considered judgment.

The answer to any complicated question can depend on the answers to a host of related questions. The length of a flight depends on the weather, the competence of the crew, and the congestion at the destination. The competence of the crew depends on the competence of each member of the crew. The competence of one individual depends on a host of psychological considerations. And so on.

In order to find the right answer to any one question, it would be nice if I could first learn the right answer to each related question that I can think of.

To define my *considered judgment* about a question Q , suppose I am told Q and spend a few days trying to answer it. But in addition to all of the normal tools—reasoning, programming, experimentation, conversation—I also have access to a special oracle. I can give this oracle any question Q' , and the oracle will immediately reply with my considered judgment about Q' . And what is my considered judgment about Q' ? Well, it's whatever I would have output if we had performed exactly the same process, starting with Q' instead of Q .

Definition

First we have to choose a representation of questions and answers. Let's pick a particular computer and represent Q and A as files stored on this computer, each of size at most one gigabyte.

We also need to define the “oracle” which can tell me my own considered judgment. Let’s say there is a special function J which I can use as a black box. J reads one file, corresponding to a question, and then immediately writes the answer.

In order to answer a question Q , I interact with my computer, on which Q is stored as a file. I can do whatever I like—write programs, call the function J , consult a friend, run an experiment, and so on. At the end of a few days, I write a new file A , which records my answer.

The outcome of this process depends both on the initial question Q , and on the behavior of the function J . (It also depends on me and my environment, but let’s hold those fixed. Every time we talk about me deciding something, it’s the same two days unfolding over and over again, just with a different question Q and a different function J .)

I’ll write $R[J](Q)$ for my answer to the question Q , where I am allowed to call the function J .

We can think of $R[J]$ is an “improved” version of J . After all, to compute $R[J]$ I can make any number of function calls to J , so in some sense it should be at least as good.

If in fact $J = R[J]$, then we say that J *reflects my considered judgment*. In this case, further reflection on my views leaves them unchanged; I have reached a reflective equilibrium. There is always at least one (randomized) function J which has this property, by Brouwer’s fixed point theorem.

A key feature of this definition is that we can actually compute $R[J](Q)$ if we can compute J . Indeed, we just have to give me a few days to think about it.

Are our considered judgments good?

You might ask: is this a satisfactory notion of “the right answer”? Are the answers it produces actually any good?

Of course, the quality of the answers depends on how I choose to produce them. I am optimistic about my considered judgment because I think there is at least one good strategy, and that we can find it.

In this section, I’ll describe my reasons for optimism. In the next section, I’ll give some reasons for pessimism.

These are long sections. Once you get the idea, you might want to skip ahead to the section **The implementation** below.

Recursion. I've defined considered judgment as a fixed point of the map $J \rightarrow R[J]$. Simple recursion is buried inside this fixed point computation as a special case.

For example, if I wanted to know "Does white win this game of chess?" I could consider each move for white in turn, and for each one compute $J(\text{"Does white win in } this \text{ position if black gets to play next?"})$. If white wins in any one of those positions then white wins the original game. If white loses in all of them, then black wins. If J makes correct judgments about each of these possible positions, then $R[J]$ makes a correct judgment about the original one.

To answer each of these sub-questions, I would use a similar procedure. For checkmate positions, I would simply return the correct result. (I need to be careful about drawn games if I want to make sure that my recursion is well-founded, but this is not hard.) By induction, my considered judgment can be perfect about questions like "Does white have a winning strategy in chess?"

Most questions can't be quite so easily broken down into intermediate questions. But even for messy questions—"How could I best get to the moon?"—the ability to exhaustively consider *every* intermediate seems to be very powerful.

Brute force search. A special case of the above is the ability to consider *every* possibility. This reduces the problem of "finding the best X" to the problem of "reliably determining which X is better."

I can use a few techniques to do a brute force search:

- Break down the search space into two pieces, and then recursively find the best item from each one. Then compare them. These breakdowns could be "dumb," such as considering "Proposals that start with 'a,'" "proposals that start with 'b,'" etc., or they could be context-dependent such as breaking down proposals to get to the moon according to where the energy comes from.
- Break down the possible methodologies you could use into two classes, and then recursively find the best item produced by a methodology from either class. Then compare them. For

example, we could do a brute force search over ways to do a brute force search.

- After searching for a good X and finding our best guess, we can ask “What is the best X?” (Yes, this is the question that *we* were asked.) If the result is better than our current best guess, we can adopt it instead. This implies that if we produce the best X with any probability, it will be identified as the fixed point. As described in the next section, this is a dangerous approach.

I can also do a brute force search for possible flaws in a design or argument, or to evaluate the expected value of a random variable, or so on.

Search for methodologies. In addition to trying to answer a question directly, I can consult my considered judgment to find the best methodologies for answering a question; to find the best frameworks for aggregating evidence; to identify the most relevant lines of reasoning and experiments that I should explore; to find considerations I may have overlooked; to identify possible problems in a proposed methodology; and so on.

Long computations. By storing intermediate results in questions, my considered judgment can reach correct answers about very long-running computations. For example, for any computation C using at most a gigabyte of memory, I could answer “What is the result of computation C?” This can be defined by advancing the computation a single step to C’ and then asking “What is the result of computation C’?”

Intellectual development. Suppose I were tasked with a question like “What physical theory best explains our observations of the world?” Using the techniques described so far, it looks like I could come up with answer as good as contemporary physics—by thinking through each consideration in unlimited detail, considering all possible theories, and so on.

Similarly, I would expect my considered judgment to reflect many future intellectual and methodological developments.

Robustness the hard way. Many of the techniques listed so far have possible issues with robustness. For example, if I perform a brute force search, I may find a solution which is very persuasive without being accurate. If I try to explore the game tree of chess without realizing that the game can loop, I may end up with a cycle,

which could be incorrectly assigned any value rather than correctly recognized as a draw.

However, I also have a lot of room to make my computations more robust:

- Every time I do anything I can record my proposed action and ask “Will taking this action introduce a possible error? How could I reduce that risk?”
- I can re-run checks and comparisons in different ways to make sure the results line up.
- I can sanitize my own answers by asking “Is there any reason that looking at this answer would lead me to make a mistake?”
- I can proceed extremely slow in general, splitting up work across more questions rather than trying to bite off a large issue in a single sitting.

Are our considered judgments bad?

On the other hand, there are some reasons to be skeptical of this definition:

Universality? The considered judgment of a 4-year-old about political economy would not be particularly reasonable. In fact, it probably wouldn’t be any better than the 4-year-old’s knee-jerk reaction. And if I had to come up with answers in a few seconds, rather than a few days, my judgment would be reasonable either.

It’s reasonable to be skeptical of a proposed definition of “the right answer,” if it wouldn’t have helped a stupider version of you—we might reasonably expect a smarter version of ourselves to look back and say “It was a fine process, but they were just not smart enough to implement it.”

But I think there is a good chance that our considered judgment is universal in a sense that the 4-year-old’s is not. We know to ask questions like “How should I approach this question?” which a 4-year-old would not. And in a few days we have time to process the answers, though we wouldn’t in a few seconds.

In simple domains, like mathematical questions, it seems pretty clear that there is such a universality property: the 4-year-old would play a

terrible game of chess even if they could consult their considered judgment, but we would probably play a perfect game.

If there is no universality phenomenon, then the fact that the 4-year-old gets the wrong answer strongly suggests that we will also get the wrong answer. But this might be OK anyway. It at least seems likely that our considered judgment is much wiser than we are.

Malignant failure modes. Suppose that my “considered judgment” about every question was a virus V, an answer which led me to answer whatever question I was currently considering with V. This is a fixed-point, but it doesn’t seem like a good one.

Such an extreme outcome seems unlikely. For example, there are some questions that I can answer directly, and I probably wouldn’t respond to them with a complicated virus V. It’s also not clear whether there is any answer that has this malicious property.

That said, especially as we consider brute force searches or cyclical dependencies, it becomes increasingly plausible that we get unanticipated answers which have surprising effects. Adversarial effects could potentially spread throughout the network of related questions, even if it couldn’t literally replicate itself or spread with 100% fidelity.

I am optimistic about this problem because we can painlessly go to heroic lengths to scrutinize our own reasoning at very high levels of detail. I expect there are levels of scrutiny sufficiently great that they could catch all plausible malignant failures. If this weren’t possible, we would want to focus our attention on well-founded sets of queries, in order to avoid the implicit optimization power of a fixed point computation.

This problem feels interestingly analogous to computing in the presence of errors. In both cases, the critical question is whether an error-correcting step is itself so error-prone that it introduces more errors than it fixes. This depends on the reliability of the underlying process (and is another possible source of a universality phenomenon) but my intuition is that by proceeding carefully we can remove possible errors from our computations faster than they are introduced.

Experiments and big computations. My considered judgment about the question “Exactly how much does Mars weigh?” is not accurate unless I know enough information to figure out the answer

(or I can collect that information over a few days). Similarly for sufficiently complex computations, which I'll treat as another kind of empirical fact.

It's not clear this is a meaningful drawback. In the intended use case of a machine that consults our considered judgment to figure out "What should I do?", it's not clear that this is a drawback at all.

If we need to know some empirical fact and the machine asks "What should I do?" then our answer may be: "an experiment." But this is the right answer, regardless of what definitions we use. We could define our considered judgment so that it knew the results of all possible experiments. But since those results aren't available to the machine it would need to do the experiment anyway before it could actually act on the result.

Once we have any evidence that bears on an empirical question, then our considered judgment can use that judgment extremely efficiently.

The implementation

We are free to define terms however we like. But can we use `Predict()` to approximate this notion of "considered judgment"?

The first attempt

We'll maintain a log L of questions and answers (encoded in self-delimited binary). By definition, $\text{Predict}(L)$ predicts the next item to be added to the log. We'll write $\text{Predict}(L)$ for the prediction of the next question or answer rather than the next bit.

We start with a large enough log that `Predict` can build a good model of the world. Once it's made enough initial errors, it should be able to predict the next log entry as well as a human could. At this point, we start asking questions.

Define $J(Q)$ to be $\text{Predict}(L+Q)$, where $+$ is concatenation of lists. That is, $J(Q)$ imagines that Q was appended to the log, and predicts what comes next. We will always follow a question with its "correct" answer, so that $J(Q)$ will try to predict the "correct" answer to Q .

If we want to compute $J(Q_0)$, then with 99% probability we make a single call to `Predict`. But with 1% probability, we should perform the following "training" process:

1. Compute $J(Q_0)$.
2. Generate a random question Q .
3. Determine $R[J](Q)$. That is, I answer the question Q , consulting J and following the procedure described in the definition of considered judgment. Computing J requires only a single call to Predict, so this is feasible. Note that the log L is not changed during this process.
4. Set $L = L + Q + R[J](Q)$.

If Predict() makes perfect predictions, then $J(Q)$ must reflect my considered judgment: $R[J](Q)$ always follows $L+Q$ in the log, so $J(Q) = \text{Predict}(L + Q) = R[J](Q)$, which is precisely the definition of considered judgment.

To ensure that Predict keeps making good predictions, we might want to continue appending sensor readings to the log, so that Predict understands its current environment and can make good predictions. (If we don't keep the log up to date, Predict should still make reasonable predictions, but it will no longer be aware of context. If my considered judgment depends on context that Predict doesn't know, then its predictions won't be so good.)

A problem

We hope that $J(Q)$ reflects my considered judgment even if Predict() is sometimes wrong. Unfortunately, it doesn't.

The problem is that Predict is trained to answer random questions—only random questions+answers ever appear in its log. A machine which answers 99.999% of random questions “correctly” need not do anything useful at all, since 99.999% of all possible questions are meaningless garbage.

When we ask Predict($L+Q'$) a *real* question Q' , one that actually makes sense, it could be wrong 100% of the time, without having any serious effect on Predict()'s overall performance.

The same problem might affect the human as well. $J(Q') = \text{Predict}(L + Q')$ is supposed to predict what the human would output, if Q' was generated as a random question. But what does a human do if they flip 100,000 coins and find that they spell out a meaningful question about the methodology of science?

The human may (correctly) infer that something fishy is going on. Hopefully they would continue to behave sensibly, but it's hard to predict and essentially impossible to test. This is closely linked to problems that Wei Dai and Vladimir Slepnev have raised with an earlier definition of enlightened judgment.

The second attempt

In order to fix this problem, we would like to ensure that Predict is trained using the same kinds of questions that J has to answer. We can do this with a nice trick.

Let Q be our random “training” question, and let Q* be a random question that we ask in the course of answering Q. We want the distribution of Q to be the same as the distribution of Q*. We will do this by asking the predictor to sample Q from its predicted distribution over Q*, rather than generating Q uniformly at random.

To this end, before asking a question we either insert a 0 or a 1 into the log. 0 indicates that we will give the predictor a question, and then expect it to predict the answer. 1 indicates that we want the predictor to predict the next question, but we won't tell it the answer.

We define $J(Q) = \text{Predict}(L + [0] + Q)$. If we want to compute $J(Q_0)$, then with probability 99% we make a single call to Predict. But with probability 1%, we perform the following process:

1. Compute $J(Q_0)$.
2. Let $Q = \text{Predict}(L+[1])$. That is, sample a question from the predicted distribution.
3. Determine $R[J](Q) = R[Q' \rightarrow \text{Predict}(L+[0]+Q')]$.
4. Let Q^* be a random question that J answered in step 1 or step 3.
With probability 1/2, set $L = L + [0] + Q + R[J](Q)$.
With probability 1/2, set $L = L + [1] + Q^*$.

As before, if Predict does a good job then $\text{Predict}(L+[0]+Q)$ will be $R[J](Q)$, and so J implements our considered judgment.

But now, rather than being uniformly random, Q is distributed according to $\text{Predict}(L+[1])$. If $\text{Predict}()$ is making good predictions, this is the distribution of Q^* , a random question that was posed to J. So $\text{Predict}()$ is being trained with exactly the same distribution of questions that J is being asked to answer.

Conclusion

Given a sufficiently powerful predictor $\text{Predict}(\cdot)$, we can implement a function J which reflects our considered judgment. If we trust the definition of considered judgment, then a machine that asks J (“What should I do?”) is almost as good as a machine that acts optimally.

With a more realistic predictor we will have to scale back our ambitions, and instead try to produce a the “most right” answer we can. Although this will require many new ideas, I am optimistic about scaling down to obtain reasonable behavior using significantly sub-human predictors.

I hope that the case of a powerful predictor helps illustrate what it might mean for a machine to do “the right thing,” and that the concrete example can help move forward the theoretical discussion about AI safety.

Human-in-the-counterfactual-loop



Paul Christiano

[Follow](#)

Jan 20, 2015 · 7 min read

Consider an autonomous system which is buying or selling assets, operating heavy machinery, or making war. People often discuss three kinds of supervision:

- **Fully autonomous.** The system operates without human supervision.
- **Human-in-the-loop.** The system does what a human tells it to.
- **Human-on-the-loop.** A human oversees the system and may override its behavior if necessary.

In the context of human-level AI, I think a fourth model of supervision is interesting:

- **Human-in-the-counterfactual-loop.** Each time the system wants to act, it consults a human with a very small probability. The system does what it thinks a human *would have* told it to do if the human had been consulted.

I think this model isolates an interesting characteristic of autonomy, and is likely to be a useful form of supervision for powerful systems.

Why bother with prediction?

Why would you have a human in the counterfactual loop rather than the actual loop?

Machines can make lots of decisions quickly. One drawback of human involvement is that human time is expensive and human responses are high-latency.

If you are involved with each decision with small probability, then:

- You can oversee many more decisions, much more cheaply. One person can oversee many machines acting in parallel.

- The system might act much faster than you come to a conclusion. You could take minutes or hours to reach a conclusion that the machine made in seconds. (This is particularly important when one person is overseeing many machines, since they may need to spend some time understanding the context in which a decision was made.)
- We can reduce human error by allowing human overseers to think for longer about each decision.

Why bother being in the loop?

Human oversight seems particularly important when it is difficult to explicitly state all of the goals of a system. For example, we might be concerned that any simple rule for asset purchases might sometimes misfire. If you buy any asset whose expected price is higher tomorrow than today, the asset might be frozen, or the purchase might have ethical repercussions, or the transaction might be politically or legally problematic, or so on. Unanticipated possibilities become more extreme and more worrying as we broaden the space of possible actions or make higher-stakes decisions.

With a human in the counterfactual loop, our system applies its predictive abilities to notice conditions that we would have objected to, if we had foreseen them. This partially addresses many of these concerns.

Although I'll describe such a system as having a "human" in the loop, the role could instead be played by any group of humans. As an extreme example, we could have a democracy-in-the-loop (though there would be a few practical issues). This is possible because the turnaround time in a counterfactual loop can be so much longer than in a normal loop.

More details

Every time an autonomous system with a human in the counterfactual loop needs human feedback, it first flips a biased coin which comes up heads with small probability (say 0.001%). If the coin comes up heads, then the system consults a human, supplying them with relevant context and getting their feedback (though the actual feedback can be provided long after the decision). If the coin comes up tails, the system does not consult a human. In either case, it predicts what feedback the human would eventually provide if the coin came up heads, and uses the predicted feedback as an input into

its decision-making process (as if the human were actually in the loop and had provided that feedback).

This requires an algorithm that can effectively predict human judgments. The more complex the domain, the harder this prediction problem is. There is no guarantee that we have predictors powerful enough to help oversee the autonomous systems that we build. But the predictors never need to exercise superhuman ability, and so we will certainly have good enough predictors by the time we reach broadly human-level AI.

The predictor needs to be able to make counterfactual predictions. But assuming that the predictor can't predict which way the coin will turn up, this amounts to a conditional prediction: *given* that the coined turned up heads, what do I predict the human will tell me? Any useful predicting system must be able to make predictions of this form.

Why actually consult the overseer?

We actually consult the overseer in order to provide a concrete experiment for the predictor to predict. If the predictor is able to consider more extreme counterfactuals, then we may not need to even have a human around. We could instead say “Predict what Bob *would* say, if we had trained him to oversee you.” This doesn’t actually change the details much. But for this post, I’ll conservatively assume that we need to actually set up the experiment in order to get good predictions about it.

There are two forces that keep us from reducing the probability of human involvement to 0:

Keeping us sharp. If the human only steps in once every century, the human may have forgotten what they are supposed to be doing. But this is a *very* weak requirement, and it applies at the level of the human operator rather than of the machine. So one human could potentially oversee a warehouse of thousands of machines, carefully reviewing only a few decisions each day (or month).

Training data. A sophisticated AI could predict the human overseer’s behavior using very few direct observations, by leveraging other related information. For example, if I hear someone say “Please be careful with those boxes” I will expect them to object to any plan that involves dropping the boxes. I don’t need a library of training data which includes boxes being dropped. That said, this still requires

some contact or common understanding between the overseer and the machines being overseen, so that they can understand how their decisions will be evaluated.

For a weak system, actually consulting the human overseer regularly may be the only way to make good predictions about the overseer.

This data can still be shared across many similar systems, potentially driving the per-system cost very low.

Many systems will be somewhere in between these extremes, able to leverage some general information (whether from observations of the overseer, or general heuristics installed by the designers of the system) but would need to get some direct observations of the overseer in order to make good predictions.

Discussion

A human in the counterfactual loop eliminates some but not all of the concerns with automation (and so serves as a thought experiment to identify which aspects of human control are most important). In particular, such a system will only take actions *it thinks* the operator is OK with—but it might be wrong about what the operator thinks is OK.

Strong predictors

For an AI with a very good understanding of humans, prediction errors would be rare, and so a human in the counterfactual loop may be better than the real thing (since they can think longer).

There are also advantages over approaches without counterfactual oversight:

- The predicted response of the human can change as the environment changes (assuming that the predictor knows about the changes, and can predict how the human would respond to them).
- The human overseer can handle all of the factors that we care about, which may otherwise be difficult to formalize.
- Because the human only needs to be able to learn about a decision when prompted, they need not have extensive training in advance. So putting a predicted human in the loop may be a lot cheaper than spending engineer time making sure you have correctly defined a domain-specific decision criterion.

With a strong predictor, any failure of the system can be traced to either (1) a failure of the predictor, or (2) a failure that also would have occurred with human oversight.

Weak predictors

For an AI with a weak understanding of humans, this proposal separates *human judgment* from *human preferences*. A system with a human in the counterfactual loop may make errors that would be caught by a human overseer, but it won't fail by virtue of having different preferences than a human overseer.

That said, in some contexts a weak predictor might do much better than a human—especially when decisions are made quickly or under emotional pressure (in the counterfactual loop, the human can take a long time and consider the issue carefully). In some cases this could be tested directly: we can record the context surrounding a decision, and empirically evaluate whether an AI or human is better able to predict the results of an extended deliberative process.

If this test suggests that the AI is a better predictor of deliberative human judgment, then the objection to automated decision-making becomes substantially weaker.

A digression on weapons. Autonomous weapons are not my primary concern in this post, but it's an example that would be hard to avoid. I think that a human in the counterfactual loop would be a legitimate response to the most common concerns with human rights violations stemming from autonomous weapons—assuming the system passed the test described in the last paragraph (which may already be possible in simple contexts). However, I'm afraid that these human rights issues obscure a more serious concern with autonomous weapons, namely that they may radically reduce the expense and risk of ending human lives.

Robustness

All approaches to oversight require the overseen system to “go along with it”: if a flexible autonomous system wanted to cut the human out of the loop, they would have many opportunities to do so. Any successful approach to oversight must either make this impossible or (more likely) ensure that the autonomous system has no motivation to cut the human out of the loop.

Having a human in the counterfactual loop doesn't fundamentally change these dynamics. It introduces a new point of failure for the

human's involvement (the predictor which is responsible for anticipating the human's response). But given how many possible points of failure there already are (the communication channel, the human's understanding of the situation, the part of the code that responds to the human's judgment, etc.), this is a minor addition.

Using a counterfactual loop makes it more realistic for the human to provide fine-grained oversight, improving the probability that they can oversee the detailed plans made by the machine or the full range of actions that might cut them out of the loop.

Conclusion

For powerful AI systems, effective human oversight may be cheaper than it appears; we should keep this in mind when thinking about superintelligent AI. The idea of counterfactual human oversight may also be a useful building block more generally.

Automated assistants



Paul Christiano

[Follow](#)

Jan 22, 2015 · 13 min read

In my last post, I described a possible scheme for using a powerful predictor to build a smart machine that does what you want it to. In that post, I assumed that the predictor was **very** powerful. In this post, I'll talk about “scaling down” these results to a more realistic predictor.

The starting proposal

Here is the proposal from my last post, with a more pragmatic—and less detailed—description. As before, we assume that we have access to a predictor P .

We'll describe an assistant that accepts spoken commands (e.g. “What is the population of Brazil?”, “Let Alice know that I'll be late” or “Make my blog look nice.”).

When you give the assistant a command Q , you may be asked to provide training data. During normal operation, this happens with small probability (say 1%) for each command. To gather training data:

1. You are given a random command Q^* . Half of the time, $Q^* = Q$. The other half of the time, Q^* is P 's guess for a random command that you will either execute or output in step 2.
2. You decide how you would want the assistant to execute Q^* , and enter a sequence of commands $F(Q^*) = [Q_1, Q_2, \dots, Q_k]$. In order to determine what sequence of commands is appropriate, you might have your assistant execute some additional commands Q' . (But you definitely won't be asked to generate more training data.)

In order to execute the command Q , the assistant does the following:

1. If Q is a primitive command, such as “Display X” or “Send message M,” then it is performed.

- Otherwise, the P guesses what the value $F(Q) = [Q_1, Q_2, \dots, Q_k]$ would have been, if you had been asked to enter training data and $Q^* = Q$ had been chosen. It then executes each instruction Q_1, Q_2, \dots, Q_k .

Note that the same training data, and hence the same function F , is used for Q itself, for each of Q 's steps Q_1, Q_2, \dots, Q_k , and for any other commands Q' that you invoke while you figure out how to implement Q^* . (This is why we randomized Q^* —so that it covers each of these cases with some probability. This randomization is actually doing a lot of work.)

My last post argued that—at least if P is a very good predictor—this is an attractive candidate for a robustly beneficial AI.

Example

Suppose I gave the command “Which direction is home?” I’d like the assistant to display an arrow pointing in the direction of my home and to tell me which compass direction it is.

With probability 1%, I am first asked to produce training data. In this case, the system speaks a random command for me to demonstrate.

If it happens to be “Which direction is home?” (this happens half of the time) then I say, “Display a map that includes my current location and my home.” The assistant brings up a map. (Rather, it does whatever I would do if I had instead been given the command “Display a map that includes my current location and my home.”) Hopefully I would correctly display the map!) I determine that my home is two miles Northwest. I then enter the sequence [“Display an arrow pointing Northwest,” “Say ‘Northwest’ ”].

In order to execute the query, P predicts how I would implement “Which direction is home?” If all goes well, then P predicts that I would output [“Display an arrow pointing Northwest,” “Say ‘Northwest’ ”]. It then uses the same procedure to execute these two commands, i.e. it predicts how I would implement each of them and does that.

This is a pretty lightweight example, but it is the same as executing a more complicated task such as “Arrange a trip to Boston.” With a more creative user, it can even apply to commands they couldn’t execute on their own, like “How could I most efficiently learn chemistry?”

The problem

This proposal involves the predictor learning a *very* complicated mapping.

In the extremely simple example above, even if I just ask it “Which direction is home?” over and over again, it must learn the mapping from (States of the world → which direction to my house). In order to answer this question, it needs to consult its compass and GPS and combine the results in an intelligent way.

As the domain becomes more complicated, the demands on P become even more extreme. Imagine a predictor which can learn to predict the design of a machine based on a description of its function. If it worked very generally, such a predictor would need to be a resourceful agent that can use its computational resources strategically and that can carry out complicated cognitive processes. This isn’t what we would normally call a “predictor,” and it already raise concerns with robustness and control.

This post asks: can we build a similarly safe system using a more realistic predictor?

Scaling down

In this section, I’ll describe two techniques for using weaker predictors:

1. Learning processes rather than results
2. Going meta

In fact [2] is a generalization of [1], but [1] is such an important special case that it seems worth discussing anyway.

Learning processes rather than results

We can think of your behavior as a loop:

1. You perceive a situation.
2. You react to that situation.
3. Your action changes the situation.

For example, if you are asked “Will it rain tomorrow?” you might respond by consulting the GPS to find your location. Once you see the

latitude and longitude, you may send an appropriate query to weather.com. Once you get the result, you may extract the probability of rain and display it on the screen.

Each of these actions changes your situation, and once you perceive the new situation you can quickly find the next action.

Rather than teaching a machine the map **Command** → **Implementation**, we could try to teach it the map **Context** → **Response**. We could then generate complex behaviors by iterating this map.

Here is a simple implementation of this idea. A *context* consists of the command which is currently being executed, as well as everything the user has input or observed while implementing the command.

A *response* consists of a command to execute in a given context, or a command to supply as the next element of the output sequence.

Instead of training P to directly execute a command, we train it to guess our response to each context. We gather training data exactly as in the original proposal, but rather than collecting a single data point of (Command→Implementation) we get a series of data points of (Context→Response), one for each command you execute while implementing Q*.

We can then implement the main loop described above:

1. You perceive a situation // P is given a context.
2. You react to that situation // P guesses your response to that context.
3. Your actions change the situation // P's response is executed by the assistant. The response itself, as well as any outputs produced by the assistant while answering, are appended to the context.

This process is then iterated until P produces an output response.

For example, to execute “Will it rain tomorrow?”, P learns to respond by querying the GPS. The assistant executes this instruction by displaying the coordinates, which are then added to the context. P responds to this new context by querying weather.com for those coordinates. This again results in an output, which modifies the context. Finally, P responds to the new context by outputting “Display

25%” or whatever the answer may be. Each of these is a very simple mapping, which can be learned using existing technology.

In this setup, P learns a policy to control a more complicated computational system. It is similar to training a neural Turing machine by trying to teach the controller by example. The difference is that P’s instructions are implemented recursively by the assistant, rather than being restricted to primitive operations.

Teaching by example can potentially lead to fast learning (and facilitates this recursive breakdown), but collecting the training data is expensive. In many contexts, it is much more efficient to generate training data automatically by experimenting with policies until you find one that achieves a given goal. For example, it is much cheaper to learn to play backgammon by playing millions of games against the computer and learning which work well, than by hiring human experts to produce training data. This is a significant problem, which can be addressed using the ideas from the next section.

Learning process rather than results also forces the assistant to use the same computational process as the human. If the predictor is weaker than a human this is fine, but if the predictor is more powerful then we will be handicapping it. This is another significant problem, which will also be addressed in the next section.

If P is a weak predictor, then we can simplify its problem by “exposing” some of our intermediate cognitive steps. For example, we might normally answer “Is $3 \times 8 > 5 \times 5$?” by carrying out the multiplication in our head. We could instead spell out our reasoning, by asking the assistant “What is 3×8 ?", “What is 5×5 ?", and “Is $24 > 25$?”. This makes the predictor’s job much easier: the map from “Is $3 \times 8 > 5 \times 5$?” → “What is 3×8 ?” is much more straightforward than the map “Is $3 \times 8 > 5 \times 5$?” → No.

Going meta

At the core of our assistant is the predictor P. In practice P might be a neural network trained to reproduce the user’s judgments, or some other relatively simple model. The original proposal is asking P to solve a problem which is *much* too challenging. There are two problems:

1. If we try to teach P processes rather than results, then its job gets easier. But it’s still probably too hard. For example, we’d like P to make really efficient use of observations of the user, to anticipate

changes in behavior, and to carry out whatever complex processing the user performs in their head.

2. Even if P is up to this task, then it's just going to be following the user's lead, which is unsatisfying. We would like P to use its capabilities to their full extent, short-cutting any computations it can and searching for more efficient alternatives.

We can address both of these problems by *going meta*.

The original assistant executes Q by asking P to predict the user's implementation [Q₁, Q₂, ..., Q_k] and then executing each step. Instead, we could execute Q by asking *the assistant* to predict the user's implementation, i.e. by executing the command "Predict how the user would implement Q." This command is executed by P predicting how the user would predict how the user would implement Q. I'll call this process metaprediction.

In fact we could execute "Predict how the user would implement Q" by going more meta, and asking the assistant to predict how the user would implement "Predict how the user would implement Q." Obviously this needs to eventually bottom out with a call to P. For now I'll set aside the question of how to choose when to go more meta.

When going meta, the predictor can use much more sophisticated procedures to predict the user's behavior. It can do inference in a causal model of the user and their environment, it can reason from analogous situations, it can apply heuristics like "the human wants to achieve X," it can use ensemble methods to combine different predictors, and so on.

Indeed, even the idea of "learning processes" is an example of going meta. One way to predict how the user would implement Q is to actually follow the same steps the user would. So when P predicts how the user would answer "Predict how the user would implement Q," it may learn to answer ["Use P to predict how the user will respond to the current context," "Execute the resulting instruction," "Modify the context appropriately," "execute Q in this new context."]

Once this is a learned behavior, it can be improved. The assistant can be taught to optimize computational processes without compromising the results; it can be taught to intelligently decide when it needs to follow the human's process and when it can just

predict the result directly; it can be taught to learn processes on its own or from other kinds of data, and so on.

Most of the process of programming an AI might be replaced by teaching P how to implement new prediction strategies. Any idea which allows an AI to predict, plan, or perceive, can be translated into a procedure for predicting a human's outputs. (Assuming the human can themselves produce the desired output with enough deliberation.) In the degenerate case, we can write functional programs by teaching P to match patterns.

As the assistant becomes better at predicting the user's decisions, it becomes easier to teach it new prediction strategies. Getting the process off the ground may require some challenging bootstrapping.

Challenges

In this section I'll talk about a few problems with the above system, and how you might address them.

Remaining “in the counterfactual loop”

You are involved with only a tiny fraction of your assistant's decisions —if you give 100 commands, you only need to implement 1, while your assistant make thousands, millions, or billions of low-level decisions.

Nevertheless, this architecture requires you to remain “in the counterfactual loop.” The right decision for the assistant is defined by what you would recommend, *if* you were asked.

In some sense this is an advantage: if your system is working well, then it won't do anything that you wouldn't. But it could also be a crippling disadvantage: you need to be able to reason about every decision the assistant makes. You might want your assistant to use cognitive strategies that you don't understand, exploit knowledge you don't have, use conventions you don't know, and reason about domains you've never seen. Is this possible?

The key idea is that you don't have to actually know about these decisions. You just have to be able to answer questions about them if asked.

If you are happy to trust a black box you don't understand, you could answer questions just by pointing to a written record. For example, if

you had to implement “Predict how the user would implement Q,” you might just look up a default answer in a database and report that. For most users, these responses might be automatic, so that the user is only ever consulted about questions they are likely to care about.

A few users may want to actually supervise the entire reasoning process, either to improve the safety or performance of their own queries, or to help develop the system further.

In these cases, the user needs to be able to understand everything the assistant is doing. But they only need to actually understand something when asked how to implement it—they can learn about it when asked. At that point they can read documentation that describes an algorithm, read comments that describe the assistant’s current internal state, receive a tutorial on the format used for some data, be briefed on relevant parts of the assistant’s history, or so on. Even if this would be an impossibly difficult task without the assistant’s help, with help it might be easy.

This ensures that the assistant is self-documenting—it never does anything that the user can’t understand. When the system adopts a new data format or cognitive strategy, explaining the change to the user (counterfactually) is a key part of implementing it. If P is a good enough predictor, then explaining the change to the users is the *only* part of implementing it (once the documentation changes, and the users’ counterfactual behavior changes, P’s predictions can change immediately). This self-documentation won’t be misleading unless the user intentionally creates misleading documentation.

I don’t know how much this requirement handicaps the system. Self-documentation is impossible until the system is smart enough to intelligibly describe its own behavior. Once it can describe its own behavior sufficiently well, I would guess that the handicap is minimal.

Imitation vs. maximization

We might train the predictor P in two different ways:

1. **Imitate.** Provide pairs (Context, Response), and train P to predict the response given the context. This is the proposal described above.
2. **Satisfy.** Evaluate pairs (Context, Response), and train P to produce a response which will score highly given the context.

This would require modifying the training procedure described above.

If the space of responses is small, the two approaches are essentially equivalent. In general, they diverge substantially.

We only need to make this choice at the lowest level—for P itself. When we use metaprediction, all that matter is how we implement “Predict how we would implement Q.” We don’t need to make a literal prediction. We can “predict” however best suits our needs, in light of any possible concerns with either [1] or [2].

But we do need to make a choice for P itself.

If P is weak, than [1] can be a dangerous option. A weak predictor can’t tell what we’ll actually do, and a rational “best guess” may be very unattractive. We would prefer to train our system directly to output reasonable answers. (Not all incorrect predictions are equally good.)

If P is strong, then [2] can be a dangerous option. A strong predictor might find outcomes which we would assign a high rating for bad reasons (and these might be the very highest rated options!). Malicious responses might fool us into thinking they are good, or might interfere with our ability to rate them accurately.

[2] is suitable for weak predictors and [1] is suitable for strong predictors. But in reality there is no such dichotomy. Most predictors will be strong in some respects and weak in others, and we won’t understand exactly what they do or don’t know. So it would be great to capture the benefits of both approaches. I don’t know how to do this.

My guess is that either approach [1] or [2] can work, because it only needs to be applied to P. If we want to use [2], we can work with predictors too simple to be ingeniously malicious, and we can use the assistant’s help to avoid being fooled by a carefully crafted answer. If we want to use [1], we can dumb down our decisions, and use computational procedures that are robust to the kinds of misunderstandings P might make.

Conclusion

Computing considered judgments looks far-fetched, but it may be possible to scale down the idea to a more practical framework.

Most of these ingredients can already be implemented. Others can be prototyped using a human's predictions in the place of P.

As usual, much more theoretical work is needed to understand when these ideas would be safe, and especially when they would be as effective as AI systems based on rational agency. But for a change of pace, experimental work may also be able to provide some insight into the feasibility of this framework.

Stable self-improvement as an AI safety problem



Paul Christiano

[Follow](#)

Jan 22, 2015 · 8 min read

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

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

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

• • •

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

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

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

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

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

Self-modification requires high confidence

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

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

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

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

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

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

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

Briefly, I think:

1. The biggest concerns are from design errors that would conceal themselves (presumably because the resulting AI becomes motivated to conceal them). It's hard to make changes that are bug-free, but it's probably much easier to avoid this very specific kind of bug, which (1) leaves the program still behaving almost exactly as intended, but (2) subtly changes what it's optimizing.
2. You can achieve relatively high confidence using normal empirical methods. Bugs arise in unanticipated and untested situations, but those aren't mostly the kinds of bugs you are worried about (see #1). It is unlikely that an unanticipated situation would cause bugs for several different supervisory systems at once.
3. Previous versions of a system can literally watch their successors operate and ensure that new changes function as intended. New systems can be deployed in such a way that deception is very hard to conceal (even the cost-benefit analysis necessary to decide when to deceive is presumably the result of some calculation which can be monitored).
4. Before AI is doing most of the AI design work, bugs are not catastrophic. Once AI is doing most of the AI design work, you don't have to make many more changes before you have massively increased your AI design capabilities (obsoleting your earlier work). For example, 10 improvements of +30% efficiency (which is a relatively small improvement for a significant software change) results in a >13x total improvement. So talking about "billions" of changes is misleading, in the most extreme case we should maybe talk about "tens" of changes.
5. Applying changes in batches rather than continuously doesn't really matter to productivity. Waiting until you have assembled a +30% productivity increase before deploying a package of changes reduces your speed by less than 25%, probably much less.

6. Just because a change can be decomposed as many smaller changes doesn't mean its probability of failure is overwhelming. Normal software often fails, but each change is composed of a large number of small features. Moreover, the failures of many related changes are highly correlated.
7. If a proposed change results in only a small productivity benefit, then it is particularly easy for the original system to provide end-to-end oversight for the improved version, e.g. listening to its arguments and evaluating its final product. And there should be *very* few changes with big productivity benefits before you have obsoleted your original investment.
8. Bugs are most challenging in a security setting, where adversaries search for improbable corner cases—when your buffer overflows, you don't normally expect it to overwrite your code in a subtle way, unless you are facing an adversary. There is generally no adversary in AI design, and I don't know of any promising approaches to AI safety that involve hardening software against even an implicit adversary. The closest you get is the AI's adversarial desire to maximize its goals; I don't think this is relevantly similar, but I'm happy to have an argument about it.
9. There are many pairs of “nearby” agents such that one is “friendly” and the other “unfriendly.” In these cases it might be easy to make a programming misstep that takes you from one category to the other. But this is nearly irrelevant; we get to choose what AI we design, and in particular we can focus on designs that are robustly good (and which are separated from adversarial deceivers by a chasm of non-functional or obviously malignant intermediates).

Standards of reasoning can't be outsourced

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

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

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

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

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

Self-improvement highlights open questions about reasoning

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

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

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

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

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

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

The absentee billionaire



Paul Christiano

[Follow](#)

Feb 20, 2015 · 5 min read

Once each day, Hugh wakes for 10 minutes. During these 10 minutes, he spends 10 million dollars. The other 1430 minutes, he slumbers.

Hugh has a goal. Maybe he wants humanity to flourish. Maybe he wants to build the world's tallest building. Maybe he wants to preserve the rainforest. Whatever Hugh wants, our question is: how can he get it?

The desideratum

Hugh has commissioned us to come up with a system for him to use—to tell him how to spend his ten minutes, what to look up, who to talk to, and how to decide where to send his money. We can give Hugh an instruction manual and spend a few hours explaining the system to him. After that, he's on his own.

Hugh wishes he could pay people to adopt the goal “Hugh’s goals are satisfied” wholesale, to pursue it as effectively as they can. In the best case, he could do this with no more expense—and no more difficulty—than would be required to hire them to do similar work. Let’s call this ideal arrangement *perfect delegation*.

We’ll consider our system a success if it lets Hugh achieve his goals nearly as well as he could using perfect delegation.

With access to perfect delegation Hugh could find the best team money could buy and hire them to administer his budget in the service of his goals. They would in turn identify and evaluate opportunities, and use perfect delegation to implement them. Doing the same using only 10 minutes a day, without perfect delegation, would be a tall order.

The details

Some details of Hugh’s situation:

1. Hugh has access to the internet. He can send and receive emails, can browse the internet, and so on. He can attach money to emails painlessly.
2. In principle, the people of Earth could all agree not to accept Hugh's money. On average, this would make them better off. But it's not going to happen.
3. The world isn't on Hugh's side, either. He doesn't have any super-trustworthy assistants, much less thousands of them. There are just a lot of people who want their share of \$10M / day. In the easy version of the problem, there are also some people who share Hugh's vision—but even then, who can tell the difference?
4. Hugh has written a lot about his goals and his outlook. Anyone who's curious can go learn about Hugh on the internet; lots of people have.
5. If someone doesn't want Hugh to learn something, they can pay a news site not to cover it. They can launch a DDoS against Google. They could even go to more extreme lengths. But we'll assume that Hugh's connection to the internet is itself tamper-proof. And just like the world won't coordinate to turn down Hugh's money, they won't coordinate to set up a parallel version of the internet just to delude him.
6. Hugh has no prospect of fixing his debilitating sleep disorder. He could leave his room if he wanted, but what is he going to do in 10 minutes, anyway? He's already arranged for his room to be secure and well-stocked, and for his infrastructure to be reliable.
7. Hugh's goals do not require the cooperation of any specific person, or any unverifiable private information. Everything that Hugh wants to do could be done by one of several people (and, as per assumption #2, we assume that these people will not all collude). All of the information that Hugh needs could in principle be verified by Hugh, if he had enough time to spend verifying it.

The prognosis

Hugh's situation would not be interesting if it were hopeless. I'm posting this puzzle because I think it pushes the boundaries of what is possible without going beyond.

I have a few ideas and a partial solution. I'll present some of them in upcoming posts. I'd also love to see other approaches to the problem!

The metaphor

Hugh's situation is a caricature, but the basic problem seems ubiquitous. If available, perfect delegation would be useful in many domains: from philanthropists making grants, to voters electing representatives, to executives managing companies. A lot can be lost between intention and implementation.

The deeper reason I care about Hugh's predicament is that I think it illustrates a much more fundamental difficulty. As a society, I don't think we yet have the hang of building organizations that can effectively pursue an on-paper mandate, or that can make decisions explicitly and rationally. Instead we get bureaucracies, we get coalitional politics, we get crude rules applied bluntly, we get inertia, we get strong cultures and personal networks that can only grow so far before they decay or drift. In many cases, the best we can do is to leave decisions in the hands of the best-equipped individuals we can find and trust.

Setting aside this more abstract difficulty, there are two concrete cases I find particularly interesting.

Philanthropy. A philanthropist with a pure heart seeks the most effective way to use their money to improve the world. They are surrounded by people pursuing their own projects for their own reasons, uninterested and sometimes unable to think about their expertise in the context of the philanthropist's broader goal. How can the philanthropist make funding decisions in a domain without implicitly adopting its practitioner's values? How can the philanthropist best leverage the knowledge of many diverse experts?

I think we have a lot of room to improve our basic institutions for this setting—Hugh's extreme situation makes the problem especially obvious, but I think it is always there. The Open Philanthropy Project is making a noble effort to attack this problem head on, and I expect they will do a lot of good. So far they are relying on a combination of small scale and staff with closely aligned values, and it will be interesting to see how these efforts scale. (I expect that the practical issues the OPP is struggling with are more pressing than the theoretical question; but I think that theoretical question has still received less attention than it deserves.)

AI Control. Unless we go out of our way to avoid it, we humans will eventually live in a world that outpaces us as much as our world outpaces Hugh. If things go well, we may also be comparably rich (in absolute terms). One way of understanding the AI control problem is: how can humans continue to influence the overall direction of society when decisions are made automatically, much more quickly and much more frequently than humans can directly oversee?

The analogy is not entirely straightforward, but I expect that a solution to Hugh's problem would also prove useful for attacking the AI control problem.

Indirect decision theory



Paul Christiano

[Follow](#)

Mar 2, 2015 · 3 min read

In which I argue that understanding decision theory can be delegated to AI.

Indirect normativity.

My preferences can probably be described by a utility function $U : [\text{Possible worlds}] \rightarrow \mathbb{R}$. But U is likely to lack a simple specification, and even if it has one, I certainly don't know it. So if I wanted to describe my preferences, I might define the utility of a world w as:

$U(w) = \text{"How good I would judge the world } w \text{ to be, after an idealized process of reflection."}$

Intuitively, if there were a powerful AI around, I'd like it to perform an action a such that $\mathbb{E}[U(w)|\text{do}(a)]$ is as large as possible.

Indirect decision theory

But what does $\mathbb{E}[U(w)|\text{do}(a)]$ mean anyway? We haven't given a prescription for interpreting " $|\text{do}(a)$," and we haven't specified a distribution over possible worlds.

Really, I'd like to leave these questions up to an AI. That is, whatever work I would do in order to answer these questions, an AI should be able to do just as well or better. And it should behave sensibly in the interim, just like I would.

To this end, consider the definition of a map $U' : [\text{Possible actions}] \rightarrow \mathbb{R}$:

$U'(a) = \text{"How good I would judge the action } a \text{ to be, after an idealized process of reflection."}$

Now we'd just like to build an "agent" that takes the action a maximizing $\mathbb{E}[U'(a)]$. Rather than defining our decision theory or our beliefs right now, we will instead come up with some answer during the "idealized process of reflection." And as long as an AI is uncertain

about what we'd come up with, it will behave sensibly in light of its uncertainty.

This feels like a bit of a cheat. But I think the feeling is an illusion. More precisely:

A successful AI will need to be able to reason about quantities like $\mathbb{E}[U(w)|\text{do}(a)]$, and we can't dodge this algorithmic problem with a sleight of hand. But a sleight of hand might dodge the philosophical hazard of committing ourselves to a particular definition of $\mathbb{E}[U(w)|\text{do}(a)]$.

U' doesn't seem any harder to define than U . Indeed it may be easier —possible worlds are complex and massive objects, and to evaluate them we might have to think long and hard and become very different people than we are today. But actions are close to home.

And U' seems every bit as actionable as U : if a program can calculate $\mathbb{E}[U(w)|\text{do}(a)]$ (whatever we mean by that), it can probably just as well calculate $\mathbb{E}[U'(a)]$.

It may be that this approach isn't tenable. But I think that is necessarily a question about the internal structure of an AI.

Possible problems

Is “idealized reflection” up to it?

In order to evaluate how good an action is, we will often want to understand its consequences. This puts an additional requirement “idealized process of reflection:” it needs to be powerful enough to understand the consequences of each possible action.

I don't think this is a big deal:

1. In order for U' to guide an AI's decisions, U' just needs to be as wise as the AI itself. It doesn't matter if we would like an action because of some hard-to-anticipate consequences, unless the AI can anticipate that we'll like it.
2. The bar for an “idealized process of reflection” into whose hands we would entrust the entire future seems much *higher* than the bar for a process of reflection that can determine the consequences of actions today.

A final wrinkle

Computing $\mathbb{E}[U(a)]$ doesn't seem any more or less complicated than computing $\mathbb{E}[U(w)|\text{do}(a)]$.

But, it seems unlikely that superintelligent AI systems will simply compute $\mathbb{E}[U(w)|\text{do}(a)]$ for each possible action a and then do the best one. For example, they may have to think about how to think; more broadly, it seems hard to predict what a successful AI system of the future will look like.

It may well be that the internal structure of AI systems favors rational agents over other designs. For example, “maximize $\mathbb{E}[U(w)]$ ” might be a really useful invariant to organize a system around, and if so it's not clear whether maximizing $\mathbb{E}[U'(a)]$ is a satisfactory alternative. (I discuss this issue inconclusively here.) It's plausible that an understanding of decision theory will help us see how the global goal-directed behavior of a system emerges from a combination of heuristics and goal-directed components; but for now we don't have a very clear picture.

Conclusion

I think this final wrinkle gives us our best reason to study decision theory today. But I think the case is weaker and more subtle than is often assumed, and I am certainly not yet convinced that we can't delegate decision theory to an AI.

The Steering Problem



Paul Christiano

[Follow](#)

Mar 14, 2015 · 22 min read

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

1. 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 the steering problem.
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. In general, scaling up fields rapidly is difficult.

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?

2. 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_i, y_i) and predicts the label of a new data point, x' . A function learner is Hugh-level if, after training on any sequence of data (x_i, y_i) , the learner's guess for the label of the next point 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 which 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 as 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.

Scaling down

“Human-level abilities” could refer to many different things, including:

1. Human-level performance on high-level tasks.
2. The level of functionality embodied in the human brain. Human-level perception, intuition, motor control, subsymbolic reasoning, and so on.

In general, as we shift from 1 towards 2 the steering problem becomes more difficult. It may be difficult to produce simple or predictable high-level functions using low-level abilities.

For example, humans pursue a complicated set of goals that would be very difficult to determine by looking at the human brain (and some of which are quite distant from the evolutionary pressures that produced us). When given a task that doesn’t serve these goals, a human may simply decide to pursue their own agenda. If we build human-like abilities out of human-like low-level functions, we may find ourselves with similarly unpredictable high-level functions.

It is harder to formalize or understand low-level abilities than high-level functions. One approach is to consider very short time periods. For example, we could consider black boxes which learn functions as well as a human who spends only 500 milliseconds per example. Unfortunately, at this level it is harder to encapsulate human abilities in a small number of simple functions, and we must pay more attention to the way in which these abilities can be connected.

If the steering problem were satisfactorily resolved, “scaling down” to these lower-level abilities would be a natural but challenging next step.

3. Objections

The simple, abstract capabilities we can think of now are much harder to use productively than the rich and messy AI capabilities we will actually develop.

For now we can't clearly state *anything* a machine could do that would make the steering problem easy (short of exactly reproducing human behavior). Filling in this gap would be an appropriate response to the steering problem.

Perhaps we don't yet know exactly what we want machines to do, but figuring it out is inextricably bound up with getting them to do it. If so, it might be easier to say what we want once we know how to do it. But by the same token, it might be easier to figure out how to do it once we can better say what we want.

In either case, it seems likely that the steering problem fills in its own niche: either it is a distinct problem that won't be solved automatically en route to AI; or else it is a different perspective on the same underlying difficulties, and can be productively explored in parallel with other AI research.

Because the steering problem is non-trivial for simple, precisely stated abilities, it may well be non-trivial for the abilities we actually obtain. Certainly we can imagine developing a human-level predictor without learning too much about how to build useful systems. So it seems unreasonable to be confident that the steering problem will turn out to be a non-problem.

The simple, abstract abilities we can think of now are much easier to work with than the human-level abilities we will actually develop, or at least much different. Building a robust system is easier when all of the pieces have clean, reliable functions; in practice things won't be so pretty.

It would be a larger leap to continue "...and the ideas required to work with simple, reliable components will have no relevance to their more realistic counterparts." *Whatever* abilities we end up with, many solutions to the steering problem will turn out to be inapplicable, and they will all be incomplete. But we can still find useful general techniques by developing ideas that are helpful for many versions of the steering problem; and we can identify important technical challenges by understanding what makes each version easy or hard.

We can gradually scale up the difficulty of the steering problem by demanding more robust solutions, making weaker guarantees on our black boxes, or working with less manageable abilities. Our choices can be informed by ongoing progress in AI, focusing on those capabilities we think are most realistic and the forms of robustness we consider most likely to be necessary.

Why is autonomy necessary?

One apparent solution to the steering problem is to retain human decision-making, with AI systems acting as assistants and tools to help humans accomplish their goals.

This is an appropriate solution while AI systems remain relatively limited. It has serious problems when scaling:

- If large numbers of machines make large numbers of decisions, with human wages orders of magnitude larger than the operating costs of machines, then the cost of human oversight becomes prohibitive. Imagine a million humans overseeing a billion or trillion human-level machines.
- If machines make very rapid decisions, human oversight can introduce unacceptable latency. Imagine human engineers overseeing the handling of individual Google searches.
- If machines work on complex problems, human overseers may not be able to understand their reasoning process. Imagine a physics undergraduate overseeing a team of world-class physicists.

All of these problems become particularly severe when we consider *thinking about thinking*. That is, machines must make numerous, rapid decisions about how to process information, what to investigate or compute, how to organize their resources, and so on. If we want to use machine intelligence to make those decisions better, that will have to be done without substantial human oversight.

It may be possible to maintain human involvement in all important automation, but doing so will eventually become a serious bottleneck. Tasks that can be performed without human oversight will become increasingly efficient, and without explicit coordination (and a willingness to make short-term sacrifices) it seems likely that more autonomous operations will outcompete their less autonomous counterparts.

Is there any concrete work to do on the steering problem?

In the next section I'll describe a handful of existing research directions that bear on the steering problem. I think the steering problem suggests an interesting and unusual perspective on each of these domains; I don't know whether it will prove to be a fruitful perspective, but if it fails it won't be because of a lack of first steps.

I have done some work motivated explicitly by the steering problem: a formalization of "judgment upon reflection," which can be expressed entirely algorithmically based on (experimentally controlled) observations of human behavior, an alternative to goal-directed behavior which may enjoy similar productivity benefits while being more robust, and some simple protocols for delegating to untrusted agents.

4. Approaches, ingredients, and related work

Rational agency

One natural approach to the steering problem is to build goal-directed agents who want to be useful or who share their creators' goals.

There are two main difficulties:

- Specifying goals in an appropriate language. What does it mean to "be useful"? How can we define what we want?
- Building agents that reliably pursue goals specified in that language.

Deploying a goal-directed agent is somewhat worrying: an agent with an almost-but-not-quite-correct goal will be working at cross-purposes to its creator, and will be motivated (for example) to avoid revealing that its goal is not quite correct. These concerns motivate a third line of research:

- Designing goals or goal-directed agents which "fail gracefully," i.e. which don't behave adversarially or resist correction, even if their goals are not perfectly aligned with their creators'.

Several lines of existing research bear on each of these questions.

Specifying goals

Rather than directly specifying what outcomes are good, it seems more promising to specify how to learn what outcomes are good. This is a topic of existing research, although the focus is typically on pragmatic considerations rather than on the more general theoretical problem.

- Inverse reinforcement learning (for example see Russell, Ng and Russell, or Ziebart et al.) and goal inference (for example see Baker, Tenenbaum, and Saxe or Verma and Rao) attempt to infer underlying preferences by observing behavior, despite a complex relationship between actions and outcomes. To apply to the steering problem, the techniques would need to be generalized to learners who are much better informed and more capable than the human models they are learning from, and who have much noisier information about the human models and their environment. This requires understanding the limitations and errors of the human models, and generalizing the human's goals robustly so that they remain acceptable even when they are pursued in an unfamiliar way.
- Preference learning attempts to infer underlying preferences from observed decisions, despite noisy information and potentially irrational behavior (for a small sample, see Fürnkranz and Hüllermeier, Fürnkranz and Hüllermeier, or Gervasio et al.). Existing work considers small domains with explicitly represented preferences, and there seem to be serious challenges when scaling to preferences over complete states-of-affairs. As a result, preference learning seems less directly applicable than inverse reinforcement learning or goal inference.
- Some more speculative and philosophical research (for example, see my post on the subject, or this more discursive article by Yudkowsky) has explored how to formalize our preferences in a general and precise way. The focus is on determining what processes of deliberation correctly capture our informed judgment and how we might formalize those processes. The primary challenge, on this perspective, is defining our preferences about outcomes that are difficult for us to describe or reason about.

We could also investigate goals of the form “maximize user satisfaction,” but it seems hard to find a satisfactory definition along these lines.

Pursuing goals

Even with desirable goals in hand, it may be challenging to design systems that reliably pursue those goals. There are questions about how goal-directed behavior relates to reasoning and to the behavior of subsystems (does the system pursue the goals it appears to?), about the theoretical basis for optimal rational behavior (does it pursue them well?), and about how an agent should behave in light of uncertainty about what outcomes are desirable.

- Some existing work in reinforcement learning (for a summary, see Sutton and Barto) and probabilistic inference (for example, see Attas) shows how goal-directed behavior can be implemented using other faculties.
- Some work in philosophy studies the formal basis for rational agency, from decision theory to epistemology. This work clarifies what it means to rationally pursue a particular goal. A lack of understanding may result in systems that appear to pursue one goal but actually pursue another, or that generalize to novel environments in undesirable ways.
- Some work on AI safety (see Dewey or Bostrom, ch. 12) explores frameworks for pursuing uncertain goals, in the interest of understanding how and to what extent “learning what is good” is different from “learning what is true.”
- Some work in multi-objective optimization considers settings with a large space of possible objectives, and seeks policies which are appropriate in light of uncertainty about which objective we really care about.

Failing gracefully

Even a “near miss” when defining a goal-directed agent might have undesirable consequences. In addition to minimizing the probability of failure, it would be nice to minimize the costs of a near miss—and to allow us to use the kind of “trial and error” approach that is more typical of software development.

- Researchers interested in AI safety have introduced and discussed the notion of “corrigible” agents, who cooperate with “corrective” interventions by their programmers—even if we cannot implement goals consistent with that cooperation.

- Some researchers have worked to make AI reasoning understandable (For example, see Bullinaria or Craven). Understandability can reduce the scope for malignant failure modes, since engineers might be able to directly monitor the motivation for decisions (and in particular to distinguish between honest and deceptive behavior).

Delegation

Every day humans work productively on projects that they don't intrinsically care about, motivated by a desire for money, recognition, or satisfaction. We could imagine an AI doing the same thing. For example, a reinforcement learner might do useful work in order to earn a reward, without having any intrinsic concern for the work being done.

Unfortunately, such delegation runs into some problems. The problems appear even when delegating to humans, but they get considerably worse as machines become more powerful and more numerous:

- Naively, the agent will only do good work when the principal can verify the quality of that work. The cost of this oversight can be non-trivial.
- Unless the oversight is extremely thorough or the problem particularly straightforward, there will be some gap between what the principal wants and what the principal evaluates. The reward-driven agent will maximize whatever the principal evaluates.
- If agents are granted much autonomy, they may find other ways to get rewards. This depends in detail on how the "reward" is implemented, and what the agent cares about.

There are many tools available to address these problems: breaking a system up into pieces with differing values and limited autonomy, performing randomized audits, automating audits and auditing auditors, relying on agents with short time horizons or extreme risk aversion, and so on. So far there are no compelling proposals that put the pieces together.

A full solution is likely to rely on agents with different values, combined with an appropriate system of checks and balances. But if the agents coordinate to pursue their collective values, they could fatally undermine such a system. We can try to minimize the risk by

making the agents' interactions essentially zero sum, or employing other agents to oversee interactions and report signs of collusion. But the possibility of collusion remains a serious obstacle.

Even if delegation cannot fully resolve the steering problem, a weak solution might be useful as part of a bootstrapping protocol (see the section on bootstrapping below).

This problem is similar in spirit to mechanism design, but the details (and apparently the required tools) are quite different. Nevertheless, some ideas from mechanism design or the economics of delegation may turn out to be applicable. Conversely, some approaches to the steering problem might be of interest to economists in these areas.

Shared language and concepts

When delegating to a helpful human, we would say what we want done in natural language, relying on a rich network of shared concepts that can be used to specify goals or desired actions. Writing programs with the same capability would greatly simplify or perhaps solve the steering problem.

In some sense, human-level language understanding is already encapsulated in human-level cognitive abilities. For example, if we were pursuing the delegation approach in the last section, we could describe tasks in natural language. The agents would infer what we expect them to do and under what conditions we will give them rewards, and they would behave appropriately in light of that knowledge. But this “language understanding” only appears in the agent’s goal-directed behavior.

To address the steering problem, we would like something stronger. We would like to build agents that share human concepts, such that we can write code that operates in terms of those concepts: specifying a goal in terms of higher-level concepts, or executing instructions defined in terms of these concepts. These tasks don’t seem to be possible using only goal-directed language understanding.

Understanding concept learning and the relationship to language is a fundamental problem in cognitive science and AI. Work in these areas thus bears directly on the steering problem.

For now, we cannot say formally what it would mean to have a program that reliably acquired “the same” concepts as humans, so that instructions expressed in those concepts would have the

intended meaning. Even given unlimited computation, it's not clear how we would solve the steering problem using concept learning. This is not at all to say it is not possible, merely that it has not yet been done.

There may be a tight connection between the theoretical question—what would it mean to learn human concepts, and how could you do it with any amount of computation—and the pragmatic computational issues. If there is a connection, then the theoretical question might be easier once the pragmatic issues are better understood. But conversely, the pragmatic question might also be easier once the theoretical issues are better understood.

Non-consequentialist intelligence

If describing our real goals is too demanding, and describing a crude approximation is hazardous, then we might try to build systems without explicitly defined goals. This makes the safety problem much easier, but probably makes it harder to build systems which are sufficiently powerful at all.

Non-agents

One idea is to focus on systems with some narrower function: answering questions, proposing plans, executing a narrow task, or so on. On their own these systems might not be terribly useful, but the hope is that as tools they can be nearly as useful as a goal-directed assistant. Moreover, because these systems don't need to be aligned with human goals, they may be easier to construct.

For example, research in decision support and multi-objective optimization aims to find good solutions to an optimization problem (or a planning problem) by interacting with a decision-maker rather than giving a scalar representation of their preferences (for example, see Fonseca and Fleming or Deb).

These systems can certainly be useful (and so may be appropriate as part of a bootstrapping strategy or as a component in a more complex solution; see the next sections). But most of them inherently require significant human oversight, and do not seem suitable as solutions to the full steering problem: for projects involving large numbers of agents with a small number of human overseers, the involvement of human oversight in all substantive decisions is probably an unacceptable overhead.

This issue applies at every level simultaneously, and seems particularly serious when we consider systems thinking about how to think. For example, in order to produce a plan, a simulated human or team would first plan how to plan. They would seek out relevant information, talk to people with necessary expertise, focus their attention on the highest-priority questions, allocate available computational resources, and so on. If human oversight is necessary for every step of this process, the resulting system is not even as useful as a black box that outputs good plans.

Of course, even if humans rely on goal-directed behavior to accomplish these tasks, this doesn't imply that machines must as well. But that would require a concrete alternative approach that still captures the benefits of goal-directed reasoning without substantial oversight.

Non-consequentialist agents

Instead we might build agents with no explicitly defined goals which can nevertheless accomplish the same tasks as a helpful human.

Perhaps most straightforward is an agent that asks “What would a helpful human do?” and then does that. If we have a particular helpful human available as a template, we could build a predictive model of that human template’s decisions, and use this model to guide our agent’s decisions. With a good enough model, the result would be precisely as useful as the human template.

This proposal has a number of potential problems. First, it does not scale to the available abilities—it is never more useful than a simulation of the human template. So it is not a general solution to the steering problem, unless we have access to arbitrarily capable human templates. Second, if our predictions are imperfect, the behavior may be significantly worse than the helpful human template. Third, making accurate predictions about a human is itself a superhuman skill, and so asking Alice to do what she thinks Bob would do can result in behavior worse than Alice would produce on her own, no matter how smart Bob is.

We can address some of these problems by instead asking “What choice would the human overseer most approve of?” and then taking the most-approved-of option. And a bootstrapping procedure can address some limitations of using a human template. I explore these ideas here.

Research on inverse reinforcement learning or goal inference could also be used to learn imitative behavior which is sensitive to human goals, rather than to build systems that infer a human's long-term goals.

There may be many other alternatives to goal-directed behavior. Any proposal would probably be combined with some ideas under “rational agency” and “shared language” above.

Bootstrapping

We might try to deal with two cases separately:

- We are dealing with machines *much* smarter, faster, or more numerous than humans.
- We aren't.

In the first case, we could try to delegate the steering problem to the much more capable machines. In the second case, we might be able to make do with a weak solution to the steering problem which wouldn't scale to more challenging cases.

It's hard to know how this strategy would work out, and to a greater extent than the other approaches we would expect to play it by ear. But by thinking about some of the difficulties in advance, we can get a better sense for how hard the steering problem will actually be, and whether a strong solution is necessary.

Roughly, there are two pieces to a bootstrapping protocol:

1. We need to solve a weak version of the steering problem. We don't have to make our machine as useful as a human, but we do need to get *some* useful work, beyond what we could have done ourselves. Ideally we'll come as close as possible to a useful human.
2. We need to use the available machines to solve a stronger version of the steering problem. This is repeated until we've solved the full problem.

Both of these steps would rely on the kinds of techniques we have discussed throughout this section. The difference is that the humans, as well as the machine intelligences, only need to solve the steering problem for machines somewhat smarter or faster than themselves.

One implication is that weak solutions to the steering problem might be amplified into strong solutions, and so are worth considering even if they can't be scaled up to strong solutions directly. This includes many of the approaches listed under "Delegation" and "Goal-less intelligence."

Problem: Safe AI from episodic RL



Paul Christiano

[Follow](#)

Apr 7, 2015 · 3 min read

In a previous post, I posed 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?

One natural ability is **episodic reinforcement learning**. We can ask: if we have a good algorithm for episodic RL, can we use it to implement a safe and useful AI?

The quoted problem statement considers “human-level,” but the same problem can be posed for any level of ability, which may be subhuman in some respects and superhuman in others. The only additional wrinkle is defining the benchmark of a “well-motivated” AI with the same abilities. There are some other technical points in my post about the steering problem.

Definition

In (online) episodic reinforcement learning, a learner participates in a series of episodes. In each episode, the learner interacts with an unknown environment, and eventually receives a real-valued reward. The learner’s goal is to receive a high total reward.

We say that an algorithm A is competitive with a fixed policy X if, for every sequence of episodes, the total payoff of A is almost as large as the total payoff of using X in each episode. (The gap is the **regret**.)

By “human-level” we mean competitive with the behavior of a particular human H. This definition of human-level is parametrized by the regret, which quantifies the training time required to converge to human level.

A stronger and more useful assumption is that A is competitive with any simple modification of A that a human could describe and implement. In this case, the regret must increase as we consider more complex proposed modifications.

Current status

A good RL agent could probably behave very “well” in the world, effectively acquiring resources and expanding its influence. But we don’t know how to build a similarly effective agent which would eventually use its resources and influence in a way that its owners would actually like.

The wide availability of powerful RL agents, without accompanying advances in our ability to apply them usefully, would probably not be good for humanity.

A common hope is that we will learn how to build more useful AI in parallel with or before understanding how to build dangerously powerful RL agents.

I think that this hope is probably justified. But nevertheless, I would feel more comfortable about AI safety if we had a more compelling answer to this problem now. I think that we understand reinforcement learning well enough that we can begin to have serious discussions about the problem, and to do relevant empirical research, today. And I think that the stakes are high enough that we probably should.

A challenge

I consider this to be a particularly challenging and important instance of the steering problem, for a few closely related reasons.

Most importantly, it seems that any algorithm which can achieve good performance in the real world, across a range of environments, can also achieve good performance in episodic reinforcement learning problems. So if we can build any kind of powerful AI at all, we can probably build an effective reinforcement learner. In this sense, episodic reinforcement learning is almost a non-assumption.

Unsurprisingly, it also seems that reinforcement learning, or at least opaque goal-directed behavior, is an especially hard case for most AI control techniques. It’s also the capability whose destructive capability is most evident and immediate, and a basic part of the story that motivates concern with AI risk. Reinforcement learning is not the only formalization of agency, but it seems to be one that features prominently in current practice, and it may be the most challenging to apply safely.

Note that the definition of reinforcement learning is general enough that it includes techniques like black box search or gradient descent over policies, even in supervised learning settings that wouldn't normally be described as reinforcement learning. Finding a good policy to interact with a computational environment (such as a scratchspace, an external memory, or computational aids) is a natural approach to implementing complex functionalities.

We make one apparently optimistic assumption: our learner maximizes performance in each episode independently rather than considering interactions between episodes. In theory and intuitively the episodic version seems easier, and this intuition is consistent with practice (most existing algorithms either apply directly to episodic reinforcement learning or can be trivially adapted to it). In many contexts we can reduce episodic reinforcement learning to single-episode reinforcement learning by crude measures such as resetting the learner to an appropriate state between episodes.

Steps towards safe AI from online learning



Paul Christiano

[Follow](#)

Apr 8, 2015 · 9 min read

Suppose that we have a good algorithm for episodic reinforcement learning, and would like to use it to build a safe AI. (A more precise problem statement.)

In this post I'll describe a very lightweight approach, whose success mostly rests on behavior of the human overseers. I'll describe some techniques that the overseers could use to get good behavior. In the next post, I'll discuss some problems with this approach.

Supervised learning

(ETA: this is a contextual bandits problem, not a supervised learning problem. It's much more like RL than supervised learning.)

Rather than working directly with an RL agent, we will consider a supervised online learner. This will simplify the presentation and make the discussion slightly more general.

In each round the learner is given a string $X \in \{0,1\}^n$, outputs a label $Y \in \{0,1\}^n$, and then receives a payoff $p(Y) \in [-1,1]$. The goal of the learner is to maximize their total payoff, for any fixed sequence of pairs X, p . It is easy to see that this is a special case of episodic reinforcement learning.

Let Predict be a good algorithm for this supervised learning problem, perhaps obtained by the reduction to episodic RL.

We'll leave the log L of past triples $(X, Y, p(Y))$ implicit, and write $\text{Predict}(X')$ for the value that Predict outputs when it has seen the log L and the new input X' . We may also include other information in the log in order to help the algorithm perform well, even when the predictor is not being scored on that information. This could include e.g. information from sensors that Predict could use to build a good model of its environment.

Our mechanism

Conceptually our mechanism is very simple, and closely analogous to the definition of considered judgment from this previous post. Let Hugh be a designated human overseer for our AI.

To answer a question Q , we use Predict to find an answer which Hugh will score highly. Rather than making Hugh score answers on his own, we allow him to use Predict to recursively answer any sub-questions that arise. We require that the recursion depth be bounded (and limited by Hugh's patience). It is probably possible to remove this restriction with a more clever design.

Formally, we define $\text{Answer}[N](Q) = \text{Predict}((N, Q))$. Let M be some fixed, large integer. Then we define $\text{Answer}(Q) = \text{Answer}[M](Q)$.

What is Predict predicting? Each time we call $\text{Answer}(Q)$, we apply the following procedure with some non-negligible probability. If we do, we wait to return the answer until this procedure is finished.

1. Set $Q^* = Q$, $N^* = M$, $A^* = \text{Answer}[N^*](Q^*) = \text{Answer}(Q)$.
2. Print (Q^*, A^*) .
3. If $N^* > 0$, let Hugh adaptively query $\text{Answer}[N^*-1]$ at any number of questions Q_1, Q_2, \dots
4. Hugh outputs a score $p(A^*) \in [-1, 1]$.
5. With probability $1/2$, append $((N^*, Q^*), A^*, p(A^*))$ to Predict 's log.
6. Let Q^* be one of the Q_i chosen at random, let $N^* = N^*-1$, and let $A^* = \text{Answer}[N^*](Q^*)$. Go back to step 2.

We then build an AI that takes the action Answer ("What should the AI do?"). We may also allow Hugh to give the robot instructions during step 3, for example he may decide to instruct the robot to take action $\text{Answer}(Q)$ before he has finished evaluating Q .

Techniques

The behavior of Answer depends on the strategy employed by Hugh. Here are some techniques that Hugh can use to try to get good answers. All of these techniques fit within steps 3 and 4 of the mechanism above, though I will describe some of them in the (equivalent) language of consulting a "different" AI to help verify one AI's work.

Expensive counterfactual oversight

As described here: because Hugh only needs to provide oversight with small probability, he can do a much more thorough job than would otherwise be possible (while still not incurring a meaningful performance penalty).

This is a necessary precondition for most of the other techniques in this section, which would otherwise be unacceptably inefficient for most applications.

The concept of counterfactual oversight may seem somewhat arcane (and the rest of the techniques in this post therefore somewhat impractical). But we can also think of it as a robust form of training data. It may be that the actual counterfactual involvement is unnecessary, and that we can just follow the contemporary paradigm of training and then testing. Or perhaps some entirely different procedure is possible. But in any case, we should be able to find something that works at least as well as counterfactual oversight.

Verification

Some actions have immediately observable consequences, e.g. if someone builds a building, I can tell whether it serves my needs, whether it collapses, and so on.

In these cases, we can ask our AI to design a building, and after the fact decide how well it performed and assign a reward appropriately. In the procedure above, Hugh can allow a long time to elapse between receiving the proposed answer A^* and actually replying with a score p .

In some cases no evaluation is feasible in the short term, but one will eventually become possible. For example, we may be able to evaluate a plan after we observe its longer-term consequences, or after we have become smarter or learned some important further facts. In principle this is not a problem; Hugh can just allow a *very* long time to elapse before making a judgment.

As the feedback is delayed further into the future it becomes particularly likely that a “good” plan will be one which compromises Hugh’s ability to evaluate it (for example by replacing him with a more agreeable rater). I’ll discuss this issue in more detail in a future post, but one basic precaution is to use a division of labor where one AI devises a plan and another AI helps Hugh ensure that the proposed plan is unlikely to compromise his ability to evaluate it. If

Hugh can determine that a plan is “safe” early, then he can safely wait a long time to determine whether it is “effective.”

Another simple precaution is to ensure that an AI making a long-term plan receives a near-maximal payoff by behaving honestly, but receives a very low payoff if we detect a manipulation attempt (see this talk for a discussion of this idea). This ensures that manipulation isn’t worthwhile unless it is very likely to succeed. The section **comparisons** below provides an observation that seems helpful for ensuring the payoff for honest behavior is high.

Bootstrapping

Verification on its own may often be prohibitively challenging. For example, imagine that we have commissioned some code and would like to confirm that it conforms to our specification. Even if we only do this confirmation with small probability, it may be intractable for humans to verify a code base that is too complex for them to have written themselves.

But this verification can itself be performed by an AI who will be rewarded for noticing possible problems (and perhaps who is able to inspect the entire cognitive process of the AI that produced the program, to help guess where a bug is most likely lurk). If the verification is performed only with small probability, then it can be performed much more carefully than the task itself (though this would involve a modification to the setup above, where Hugh can elect to use more powerful predictors for some queries).

Of course the work of this evaluator can itself be evaluated with the help of another AI. The key question is whether this bootstrapping ever “bottoms out,” with an AI whose behavior is sufficiently simple that a human can actually understand and evaluate it. This issue is discussed in these two posts; the other techniques on this list just need to be good enough that they let us evaluate each answer using a slightly less complicated process than the one that produced it.

Comparisons

Assigning an action a score between -1 and 1 may be challenging, since it involves an implicit comparison between that action and an unknown alternative. But Hugh can instead ask AI questions like “What alternative actions would have been better?” and then try to make pairwise comparisons between the action that was taken and the alternative that was proposed, giving a high reward iff the original seems better or at least nearly as good. We can make these

comparisons in advance or after the fact, we need only make them with small probability, and we can make them with the assistance of the same AI that proposed the original action.

Arguments

More generally, most plans that are good seem to be good for comprehensible reasons.

That is, suppose an AI proposes an action which will be very hard to evaluate after the fact (or that an AI claims that a given plan is no good, or that a given program has a subtle bug, or so on). If we understood the consequences of this action, the reasons why it seemed good to the AI, the possible problems with the action, and so on, then it seems like we should typically be able to tell whether the proposal is good or bad.

The AI could provide an argument for its action, which we could evaluate to the best of our abilities. We could then ask the AI to find the plan that is supported by the most compelling argument, and take that one. But it seems much more effective to have an adversarial process in which another comparably intelligent AI helps us find holes.

Any techniques for arbitrating arguments can be applied to this situation. For examples, see past posts about arguments here and here. As described in those posts, a human arbiter can implicitly examine very large issues, by breaking them down into smaller questions and focusing on subquestions where the disputants disagree.

Reasoning about complex arguments may sometimes be harder than making a decision, and so an AI may be able to find a good plan without being able to justify it. But (1) we can structure arguments in such a way that intermediate claims are just empirical predictions, and such that “tell the truth” is a simple and effective rhetorical strategy, (2) we can carry out arguments only with very small probability, so it’s fine if having arguments is merely slower or more expensive than making decisions, as long as it’s possible at all, (3) we can make efforts to closely match the intellectual abilities of the two arguers (e.g. by using the same AI in both roles), so that differences in rhetorical ability are unlikely to have a big effect.

Verify process, not product

If a machine vision system tells me “This is a picture of a horse,” it’s not clear that they can give any compelling argument for their

conclusion. In the case of images we could of course just look and see if we agree. But the problem is harder when machines are making analogous judgments that are beyond human abilities. For a simple analogy, imagine justifying “This is a picture of a horse” to a society composed entirely of blind people.

In this case it is possible to make a different sort of argument. Namely, the learner can claim “The parameters of <this> model were adjusted so that it reliably outputs ‘horse’ on pictures labeled ‘horse,’ and not pictures with other labels.” The learner can also justify each of: <this> model outputs ‘horse’ on this picture, the training images labeled ‘horse’ were really horses, the current image is drawn from a distribution sufficiently similar to the training data, and so on. Altogether, this constitutes a compelling argument that the image in question probably depicts a horse, even to a society that has never before encountered a picture of anything.

Of course this argument could fail for many subtle reasons, but we can rely on a comparably sophisticated AI to identify possible problems in a proposed argument.

Note that we can potentially verify even very complex processes in this way (for example, processes that take billions of steps), by having one AI describe what each step is doing, and having another AI point out the aspects of this description that we are most likely to find problematic upon deeper investigation.

In the worst case, this amounts to simply solving the AI control problem on a case-by-case basis: rather than producing an AI which makes human-friendly decisions in general, we just need to find a way to certify that the process used to make a *particular* decision was human-friendly. And this certification can itself be assisted by another AI using a process which is nearly as complex as the original.

The unknown

This is a short list of approaches that Hugh could use to get good performance out of his AI. It’s clearly not a complete list, and Hugh would be at a significant advantage in coming up with new items once he knows more detail about the AI’s reasoning and has the opportunity to interact with it at length.

The main reason to have such a list now is to better understand whether these kinds of techniques are likely to be adequate, or whether a fundamentally different approach is needed.

Challenges for safe AI from RL



Paul Christiano [Follow](#)

Apr 9, 2015 · 10 min read

In this post, I'll describe and discuss a few big problems for the proposal from my last post. My overall sense is that none of these issues are fundamentally deal-breakers. But there are enough possible issues that I probably wouldn't want to bank on the proposal, and I think that in the long run it would be much better to find actual security arguments rather than a non-exhaustive list of possible problems that don't seem fatal.

I think that the most troubling scenario is outlined in the last problem, and I'll probably discuss this troubling scenario in more detail in a future post.

Problems

Problem 1: Maximization

The most obvious problem is that our AI is incentivized to find answers which receive the highest possible score, rather than answers which are actually good. We hope that these are answers which we would judge to be good, given careful judgment and the available facts.

But we may receive answers which are malicious in one way or another. They may exploit our misconceptions or biases, leverage other factors in our decision-making (e.g. by establishing credible threats of retaliation for a low rating), or could seize control of their own reward channel by a physical or computer security exploit.

These problems become more serious when we attempt to evaluate actions by seeing how they play out, or by waiting until the future when we may be better-informed in general. But they are issues even if we are merely evaluating arguments presented as text.

Bootstrapping

Our main defense is applying other AIs to help protect ourselves from exploits. While a radically smarter AI may be able to find a “bad”

answer which convinces or coerces us into supplying a high rating, it will have a much harder time fooling a similarly intelligent peer.

We can use such bootstrapping in a two ways:

1. We can harden systems in advance based on advice from an AI, patching security loopholes, possible social exploits, or possible biases or lapses in judgment. This takes place in advance of making a query.
2. We can consult an AI to help evaluate a proposed action, by asking whether this particular proposal is in any way manipulative. We can then evaluate the claim of manipulativeness in the same way that we evaluate other claims, and if we conclude that an action was manipulative, we can adjust payoffs accordingly. This all takes place in step 3 of the process from the last post, though we may want to modify the process by allowing Hugh to pose individual questions to a predictor that is dumbed-down in one way or another.

Note that step 2 can take place before we actually look at a message or process it in any non-trivial way—we could have a sanitizing AI look at the message and try to detect and convince us of possible exploits, before we look at the message.

I see two major concerns with this approach.

First, it may be very difficult to harden an evaluation procedure against attempted manipulation. For example, it may be that even an AI's peers cannot adequately evaluate its arguments without being vulnerable to manipulation. In this case we could attempt to evaluate arguments with the assistance of AIs which are in some respects more powerful than the original arguer, but overall the prospects for argument look rather bleak in this world. I don't find this story very plausible or concerning, but the issue may not be settled until the empirical evidence is in.

Second, the AIs we consult may themselves manipulate us. To avoid this problem, we need these AIs to be “safer” than the AI whose argument we are evaluating. We can then defend ourselves against their manipulations using an even safer AI, until the process bottoms out with an AI which we can expose directly to human evaluators.

These assistant AIs can be safer by being less smart, by having less time to think, by being subject to more conservative evaluation by us,

by being restricted to make easier-to-evaluate arguments, by being forced to send simpler or shorter messages, or so on.

The key assumption is that notifying us of a possible exploit is “easier” than actually exploiting it, in at least one of these senses. That is, it can be done with less time to think or with less clever thoughts, it can be communicated in a less expressive medium or shorter message, or it can be correctly identified as a possible exploit by a more conservative evaluation process. Note that the notifier need not be able to work out the details of the exploit, they simply have to identify a possible vulnerability that shouldn’t exist and might be exploitable by a sufficiently powerful attacker.

This seems very plausible to me. My main reason for optimism is that I haven’t encountered any possible errors that weren’t easier to point out than to exploit, in at least one of the above senses.

This process may sound unreasonably expensive, but it would only be applied rarely if ever. Each time we patch a vulnerability we can hopefully learn from our mistakes, and hopefully we don’t have to repeat the process that many times. Moreover, it may be that most vulnerabilities are never even patched; the mere possibility of patching them when prompted could be sufficient to dissuade a manipulator from trying to exploit one.

Safety valves

A second defense is to ensure that manipulative answers are non-catastrophic, and to fix up problems once they are identified.

To some extent this is an automatic feature of our protocol, in which each learner wants to maximize their payoff but doesn’t have greater ambitions to take over the world.

We could try to make this more robust by inserting safety valves: giving AIs every possible opportunity to receive payoff by making an attack in a non-destructive context. For example, we could periodically host arguments in which one side had a significant cognitive or structural advantage (more time to think, more information, a more expressive communication channel) but was saddled with defending an obviously false claim. If an agent were able to win one of these arguments, it would highlight a flaw which could subsequently be corrected, while (typically) causing very little collateral damage.

If we really have good episodic RL agents, they won't forego such opportunities in order to receive a higher payoff in future episodes: if they did that too many times, then we'd have a sequence of episodes in which their total payoff can be significantly increased by a simple modification, contradicting their goodness.

Problem 2: Uneven abilities

Weak understanding of humans

An AI may be very good at some tasks before it is able to reason effectively about humans. Such an AI may be able to devise very effective strategies for expanding their own influence, without being able to communicate effectively with humans or to predict what humans would or would not approve of if they were fully informed.

This seems to be a possible problem, but there are a few safeguards that leave me relatively unconcerned:

1. Reasoning about humans is an important part of interacting with the world, and one that humans are very keen to see AI excel at. So there is a good chance that it will be developed alongside other capabilities.
2. Even given a relatively modest ability to reason about humans and a very great ability to reason about other things, it is still possible to express arguments and explain plans. Indeed, if the only generalization you can make about humans is "they are magically able to figure out whether something is true by thinking about it," together with a very crude model of their preferences, then the proposals from my previous post will probably be effective. So the only really problematic case seems to be the one where the ability to reason about humans is *very* minimal. Given the current trajectory of AI, this doesn't seem like a very likely capability state. I discuss this issue briefly here.
3. An AI does not need to have a very detailed model of human preferences at all to pursue them relatively effectively. See the next section.

Weak arguers

Our proposal relies heavily on the ability of the AI to justify its behavior, as well as actually behaving well. It's still not clear whether an AI would necessarily have this ability, even given an adequate ability to predict human responses to possible arguments. This is

closely related to the concerns in problem 4. It's fine if this is much harder for them, as long as they can do it when called upon. And there is a great deal of latitude in the format of those arguments. But still, it's going to be hard to address this concern except with empirical evidence.

Other

Of course there may be other respects in which uneven abilities cause a problem. For example, an AI may just be really good at making explosives, and not so good at building anything useful. I think this is a very different issue which shouldn't be addressed at the same time, or with the same methods, as other concerns with AI risk.

Problem 3: Human values as a handicap

There are several respects in which an AI pursuing human values might operate at a fundamental handicap:

1. The AI may not know what outcomes are good or bad according to human values, and acquiring that information from humans may be a bottleneck.
2. Even if human values are understood well, it may be harder to figure out how to promote human values than how to acquire resources.
3. Acquiring resources most effectively may fundamentally involve compromising human values.

Issue #3 is mostly orthogonal to AI. That is, we may live in a world in which the most effective way to gain power is to commit acts of war, to pursue risky strategies which endanger life on Earth, or to do other terrible things. In that case we'll probably have to coordinate to prevent terrible things from happening, whether or not we build AI. AI doesn't seem to significantly change the difficulty of that coordination problem, except insofar as it generally improves humans' abilities to realize their goals (including both the goal of coordination and the goal of wanton destruction).

It seems that we can sidestep issues #1 and #2 if we are willing to accept a very slight loss. Namely, we can imagine an AI which pursues a simple proxy for human values: acquire resources, don't do anything really objectionable, keep the user safe (or humanity, or

whatever other group you represent), and eventually consult them about what to do with the resources.

It doesn't take much interaction with humans to determine that this is a reasonably good strategy, and to find a "good enough" model of really objectionable behavior (probably it doesn't take any interaction with humans at all, just reading things we've written would be plenty). And this strategy isn't much more complex than "acquire resources." I don't think an AI would actually follow a strategy like this—I think it would follow heuristics that are even simpler and even less objectionable—but this is a lower bound.

Formalizing this proxy precisely would be somewhat difficult, because of the complexity of the mandates "consult the user..." and "keep the user safe..." and even "acquire resources..." But it is an information-theoretically and computationally simple goal, and if an AI is able to reason about the world effectively it can presumably identify and pursue these kinds of simple subgoals. So whatever our AI does, it ought to receive feedback at least as good as if it implemented a strategy of this form—and if the evaluator is doing their job well, this means that the resulting outcomes are at least as good as if the AI implemented a strategy of this form.

Overall, I am not concerned that respecting human values will be a major handicap for an AI.

Problem 4: Inscrutability

In our protocol, most of the actual agency resides with the human users, in their counterfactual evaluations of actions. In order for this to work, the users need to actually be able to tell what actions further their values, if the reasoning is explained to them in sufficient detail (perhaps using the kinds of tricks described here and here, to "explain" things in a normally unattainable level of detail).

Some reasoning processes, like most search or inference processes in a fixed model, seem to produce comprehensible results. With sufficient time, we could inspect the results and verify that they are good explanations for the data or good plans to achieve the desired ends.

But other processes may produce completely inscrutable outputs. For example, an AI may learn a model for the world that has a great deal of latent structure. That latent structure need not correspond to any

human concepts. So while some intermediate steps of plans may be expressible in terms that a human can understand, some steps may only make sense in terms of completely alien concepts.

It may be possible to explain these alien concepts with enough time, or to convince a human that they relate to each other in certain ways, but this isn't guaranteed. Good reasoning may be fundamentally inscrutable.

Of course the process that produced the reasoning may be comprehensible anyway. We can be convinced that a latent variable has explanatory power, and that it is most likely to take on a particular value in a particular case, even without understanding anything about the meaning of that variable. In many cases, that would be enough to let us evaluate a plan. We seem most likely to run into trouble when the reasoning process really blurs the lines between prediction and planning; where even the internal thought process reveals no useful predictions beyond "This plan will be awesome."

The very worst case

In the very worst case:

- The most productive plan of action is incomprehensible.
- The process that produced that plan is equally incomprehensible.
- The safety of the plan is incomprehensible: the only way we can know it will be safe is to observe after the fact that no one died and got replaced by an adversarial superintelligence. Note that this is only a problem for plans which may be *unusually* unsafe. A plan which merely killed everyone would be fine, because it wouldn't cause us to incorrectly rate the plan favorably in retrospect.
- The plan cannot be made safe with the assistance of any AI acting in a less-incomprehensible way. The only way to compete with incomprehensibility is with more incomprehensibility.

I don't know if this sort of thing is likely to happen; it seems unlikely to me, but I don't have any really compelling argument. If this is the nature of good plans, it seems to present a serious challenge for many approaches to AI safety.

Safe AI from question-answering



Paul Christiano

[Follow](#)

Apr 10, 2015 · 5 min read

(Warning: minimal new content. Just a clearer framing.)

Suppose that I have a question-answering box. You give the box a question, and it gives you an answer. For simplicity, let's focus on yes/no questions, and assume it just outputs yes or no. Can we use this box to implement an efficient autonomous agent that is aligned with human values?

Let's make the following assumptions:

1. The box never gives an answer predictably worse than the answer that some reference human would give. That is, there is no distribution over questions Q where we expect the human to guess the right answer more often than the box.
2. Whatever language the box uses, it's *at least* as expressive as mathematical notation.

Under assumption [2], the worst possible box is a "math intuition module," a box which takes as input a mathematical assertion and outputs a probability. We'll assume that we are working with such a box, though the whole thing would just be easier if we had a box which could answer questions in e.g. natural language.

A proposal

We'll describe an AI which outputs binary actions. To make a decision, it asks the following question:

*If the user considered my situation at **great** length, which decision would they think was best?*

It then takes the most probable action.

Formally (with links):

- Step 1: We provide a precise definition of "the user" as an input-output mapping, by giving concrete examples and specifying how to learn the model from these examples. This specification

can't be used to efficiently find a model of the user, but it does give a formal definition.

- Step 2: We provide a precise definition of "considered my situation at **great** length" by defining a simple interactive environment in which the user can deliberate. The deliberation may be very, very long—it may involve growing smarter, simulating an entire civilization, developing new branches of science, examining each possible decision's consequences in detail... The interactive environment is designed to facilitate such a prolonged deliberation.
- Step 3: The user eventually outputs either 0 or 1. The choice may be made based on a deep understanding of what the AI is thinking and why, and what the effect of a particular strategy would be on the AI's decision. This seems like a good idea for getting a good outcome, but it also leads to a complicated analysis. For the purpose of simplicity, we'll assume that the after deliberation the user represents the quality of each possible outcome by a utility in [0,1], and then chooses an action randomly so that the difference in the probabilities is equal to the difference in the utilities that would result if the AI took one action or the other. (Note that this depends not only on the state of the world, but on the behavior of the AI in the future)

Given the simple interpretation of step 3, our AI is maximizing the utility function defined by the user's judgment-after-deliberation. But the user has a great deal of latitude here, and they could e.g. direct the AI to make arbitrary changes to its own code. So this procedure doesn't commit the user to bounded utilities, or to essentially any non-trivial philosophical or normative positions.

Note that any interesting behavior for this agent will be composed of a long sequence of binary decisions. But we can express any message in binary, so this is a weak assumption.

This procedure can be easily adapted to AI that maintains state, by including the AI's current state in the question: ("What would the user want the AI to do, when its state is given by X?"). We can use similar questions to find appropriate updates to the AI's state based on its recent observations. The deliberation then needs to consider a weighted average over states of the world that correspond to a particular state for the AI. We'll set aside this complication for now.

A proof

Theorem. Let A be the AI described above, and let B be another AI. Assume that the user follows the simplified strategy described in step 3 above. Suppose that running B results in a higher expected utility, according to the user's judgment-after-deliberation, than running A. Then, by using B we can sample from a distribution over questions for which we expect B's answer to be right with higher probability than A's answers.

So in some sense, this design makes optimal use of the question-answering system we have available. We can't get a better outcome without also building a better question-answerer—at least not if the deliberative process successfully captures the user's values. So the quality of the proposal comes down to the quality of the deliberative process and the model of the user.

Proof. To produce the distribution, we simply run B and choose a random question it is posed. It's easy to evaluate the sum of the differences between $P(B \text{ is right})$ and $P(A \text{ is right})$ over this whole sequence: it's a telescoping sum that adds up to $U(\text{taking B's decisions}) - U(\text{taking A's decisions})$. So if B receives a higher utility, B also has a higher probability of being right for a random question from this distribution.

A problem

I discuss some possible problems in the posts linked under step 1 and step 2 above, and in this post describing an older version of the proposal. There seems to be one major problem (and many minor problems), with several facets:

- The quality of the scheme rests on the quality of the user's outputs upon deliberation. But we can never actually run this deliberative process, and so we can't test if our definition is reasonable.
- During the deliberative process, the user's experiences are completely unlike their everyday experiences. This may introduce some divergence between the actual user and the hypothetical user. For example, the hypothetical user may become convinced that they are in a simulation and then start behaving erratically.
- Because we can't ever run this deliberative process, we can't ever give our AI training data for the kinds of problems that we want it to actually solve. So we are relying on a kind of transfer learning that may or may not occur in practice.

- The last point raises a general concern with the question-answering system as a model of AI capabilities. It may be that this just isn't the kind of AI that we are likely to get. For example, many approaches to AI are based heavily on finding and exploiting strategies that work well empirically.

I don't think these problems are deal-breakers, though I agree they are troubling. I think that if the system failed, it would probably fail innocuously (by simply not working). Moreover, I think it is more likely to succeed than to fail malignantly.

But taken together, these objections provide a motive to look for solutions where we can train the AI on the same kind of problem that we ultimately want it to perform well on. We can capture this goal by trying to build a safe AI using capabilities like supervised learning or reinforcement learning.

The state of the steering problem



Paul Christiano

[Follow](#)

Apr 10, 2015 · 5 min read

The steering problem asks: given some powerful AI capabilities, how can we engineer a system that is both efficient and aligned with human interests?

Here's what I think we know about the steering problem right now:

- **Given a question-answering system**, that can answer any precisely posed question as well as a human, I think we have a reasonable candidate solution. But the proposed solution is hard-to-test, relies on the sensible behavior of humans in very exotic situations, and is hard to adapt to more realistic capabilities.
- **Given a very powerful predictor**, I think we have a promising but unsatisfying solution. The big problem is that requires a *very* powerful predictor. The big advantage is that the solution can be tested thoroughly, and the predictor can be tested and trained directly on “important” predictions (rather than relying on transfer learning).
- **Given a reinforcement learner or supervised learner**, I think we have a superficially plausible solution. I have little idea whether it would really work. A solution to the steering problem under these assumptions would imply a solution under essentially any other reasonable assumptions.

Overall, this problem seem much easier than I anticipated. I feel more like “It’s easy to see how things could go wrong,” rather than “It’s hard to see how things could go right.”

I think these approaches are mostly common-sensical, and are in some sense an evasion of the more exotic issues that will need to be resolved eventually. (“What do we really want?”, “what standards of reasoning should we ultimately accept?”, and so on.) But in fact I think we have a good chance of evading these exotic issues for now, postponing a resolution until they no longer seem so exotic.

Open problems

Better solutions for reinforcement learning. My solution for reinforcement learning is definitely dubious. It would be great find new approaches, find new problems with the existing approach, better patch known problems with the existing approach, or try to find a more robust/reliable way to reason about possible solutions.

I'm confident that there is room for significant improvement, though I'm very unsure how good a solution we can ultimately find.

More conservative assumptions. All of these solutions make a number of “nice” assumptions. For example, I assume that the training error of our algorithms is either very small, or mostly incurred very early, or else that there is no small set of “make or break” decisions. But can we design algorithms that are robust to a modest number of adversarial failures at any point during their execution? (Note that adversarial failures can be correlated across different AIs, and that there are a number of reasons that such correlations might arise.) Or can we articulate plausible guarantees for our algorithms that rule out problematic behaviors?

Another mild assumption is that a modest amount of human labor is available to oversee AIs (we aren't trying to make an AI that can reconstruct civilization after an apocalypse). Removing this assumption is also an interesting problem—not so much because the scenario itself is particularly plausible, but because it could lead to much more robust solutions.

Move on. I think it may be helpful to specifically ask “Supposing that we can solve the steering problem, how can things still go wrong?” For example, we still need to avoid undesirable internal behavior by pieces of a system optimized for instrumental purposes. And we wouldn't be happy if our RL agent decided at a key moment that it really cared about self-actualization rather than getting a high reward. (It wouldn't be completely unheard of: humans were selected for reproduction, but often decide that we have better things to do.)

Are these serious problems? Are there other lurking dangers? I don't really know. These questions are more closely tied up with empirical issues and the particular techniques used to produce AI.

White box solutions. (*See next section.*) All of these solutions are “black box” approaches. It would be good to find white box solution in any model, under any assumptions. That is, to implement a white

box solution using *any* well-defined capability, or even infinite computing power.

To formalize the “white-box” requirement, we can try to implement the preferences of uncooperative agents, or work under other pessimistic assumptions that make black box approaches clearly unworkable.

Along similar lines, could we design a system that could efficiently create a good world even if its operators were unaging simulations of young children? Or a dog? Are these questions meaningful? If we know that a solution doesn’t work or isn’t meaningful for sufficiently immature or underdeveloped humans, can we really suppose that we are on the right side of a magical threshold?

Black box vs. white box methods

(This section’s dichotomy is closely related to, but different from, Wei Dai’s here.)

All of these solutions use human judgment as a “black box:” we define what the right behavior is by making reference only to what humans would do or say under appropriate conditions. For example, we think of someone’s judgment as a “mistake” if they would change it after thinking about it enough and having it explained to them.

A different approach is to treat human behavior as a “white box:” to reason about *why* a human made a certain decision, and then to try figure out what the human really wanted based on an understanding. For example, we might say that someone’s judgment is a “mistake” by looking at the causal process that produced the judgment, identifying some correspondence between that process and actual facts about the world, and noticing possible inconsistencies.

White box approaches seem more intuitively promising. Inverse reinforcement learning aims to model human behavior as a goal-directed process perturbed by noise and error, and to use the extracted goals to guide an AI’s decisions. Eliezer describes an analogous proposal in *Creating Friendly AI*; I doubt he stands by the proposal, but I believe he does stand by his optimism about white box approaches.

Black box approaches suffer from some clear disadvantages. For example, an AI using one might “know” that we are making a mistake, yet still not care. We can try minimize the risk of error (as

we usually do in life), but it would be nice to do so in a more principled way. There are also some practical advantages: white box approaches extract motives which are *simpler* than the human they motivate, while black box approaches extract “motives” which may be much more complex.

That said, I don’t yet see how to make a white box solution work, even in principle. Even given a perfectly accurate model of a person, and an unlimited amount of time to think, I don’t know what kind of algorithm would be able to classify a particular utterance as an error. So for now I mostly consider this a big open question.

The easy goal inference problem is still hard

Goal inference and inverse reinforcement learning



Paul Christiano

[Follow](#)

Apr 11, 2015 · 5 min read

One approach to the AI control problem goes like this:

- Observe what the user of the system says and does.
- Infer the user's preferences.
- Try to make the world better according to the user's preference, perhaps while working alongside the user and asking clarifying questions.

This approach has the major advantage that we can begin empirical work today—we can actually build systems which observe user behavior, try to figure out what the user wants, and then help with that. There are many applications that people care about already, and we can set to work on making rich toy models.

It seems great to develop these capabilities in parallel with other AI progress, and to address whatever difficulties actually arise, as they arise. That is, in each domain where AI can act effectively, we'd like to ensure that AI can also act effectively in the service of goals inferred from users (and that this inference is good enough to support foreseeable applications).

This approach gives us a nice, concrete model of each difficulty we are trying to address. It also provides a relatively clear indicator of whether our ability to control AI lags behind our ability to build it. And by being technically interesting and economically meaningful now, it can help actually integrate AI control with AI practice.

Overall I think that this is a particularly promising angle on the AI safety problem.

Modeling imperfection

That said, I think that this approach rests on an optimistic assumption: that it's possible to model a human as an imperfect rational agent, and to extract the real values which the human is imperfectly optimizing. Without this assumption, it seems like some additional ideas are necessary.

To isolate this challenge, we can consider a vast simplification of the goal inference problem:

The easy goal inference problem: Given no algorithmic limitations and access to the complete human policy—a lookup table of what a human would do after making any sequence of observations—find any reasonable representation of any reasonable approximation to what that human wants.

I think that this problem remains wide open, and that we've made very little headway on the general case. We can make the problem even easier, by considering a human in a simple toy universe making relatively simple decisions, but it still leaves us with a very tough problem.

It's not clear to me whether or exactly how progress in AI will make this problem easier. I can certainly see how enough progress in cognitive science might yield an answer, but it seems much more likely that it will instead tell us “Your question wasn't well defined.” What do we do then?

I am especially interested in this problem because I think that “business as usual” progress in AI will probably lead to the ability to predict human behavior relatively well, and to emulate the performance of experts. So I really care about the residual—what do we need to know to address AI control, beyond what we need to know to build AI?

Narrow domains

We can solve the very easy goal inference problem in sufficiently narrow domains, where humans can behave approximately rationally and a simple error model is approximately right. So far this has been good enough.

But in the long run, humans make many decisions whose consequences aren't confined to a simple domain. This approach can work for driving from point A to point B, but probably can't work for designing a city, running a company, or setting good policies.

There may be an approach which uses inverse reinforcement learning in simple domains as a building block in order to solve the whole AI control problem. Maybe it's not even a terribly complicated approach. But it's not a trivial problem, and I don't think it can be dismissed easily without some new ideas.

Modeling “mistakes” is fundamental

If we want to perform a task as well as an expert, inverse reinforcement learning is clearly a powerful approach.

But in the long-term, many important applications require AIs to make decisions which are *better* than those of available human experts. This is part of the promise of AI, and it is the scenario in which AI control becomes most challenging.

In this context, we can't use the usual paradigm—“more accurate models are better.” A perfectly accurate model will take us exactly to human mimicry and no farther.

The possible extra oomph of inverse reinforcement learning comes from an explicit model of the human's mistakes or bounded rationality. It's what specifies what the AI should do differently in order to be “smarter,” what parts of the human's policy it should throw out. So it implicitly specifies which of the human behaviors the AI should keep. The error model isn't an afterthought—it's the main affair.

Modeling “mistakes” is hard

Existing error models for inverse reinforcement learning tend to be very simple, ranging from Gaussian noise in observations of the expert's behavior or sensor readings, to the assumption that the expert's choices are randomized with a bias towards better actions.

In fact humans are not rational agents with some noise on top. Our decisions are the product of a complicated mess of interacting process, optimized by evolution for the reproduction of our children's children. It's not clear there is any good answer to what a “perfect” human would do. If you were to find any principled answer to “what is the human brain optimizing?” the single most likely bet is probably something like “reproductive success.” But this isn't the answer we are looking for.

I don't think that writing down a model of human imperfections, which describes how humans depart from the rational pursuit of

fixed goals, is likely to be any easier than writing down a complete model of human behavior.

We can't use normal AI techniques to learn this kind of model, either —what is it that makes a model good or bad? The standard view —“more accurate models are better”—is fine as long as your goal is just to emulate human performance. But this view doesn't provide guidance about how to separate the “good” part of human decisions from the “bad” part.

So what?

It's reasonable to take the attitude “Well, we'll deal with that problem when it comes up.” But I think that there are a few things that we can do productively in advance.

- Inverse reinforcement learning / goal inference research motivated by applications to AI control should probably pay particular attention to the issue of modeling mistakes, and to the challenges that arise when trying to find a policy better than the one you are learning from.
- It's worth doing more theoretical research to understand this kind of difficulty and how to address it. This research can help identify other practical approaches to AI control, which can then be explored empirically.

Technical and social approaches to AI safety



Paul Christiano

[Follow](#)

Apr 12, 2015 · 7 min read

I often divide solutions to the AI control problem into two parts: technical and social. I think any solution requires progress on both fronts, but strength on one front can compensate for a weakness on the other.

This view suggests that most of the value comes from having a “good” technical solution (quantification below), rather than a “perfect” solution or a “not terrible” solution. It also suggests that there is value from progress on both technical and social solutions, rather than justifying a focus on one or the other.

Technical solutions

If we can build AI systems that are both efficient and safe, then we have addressed the most serious concerns about AI risk, regardless of how well or poorly society coordinates. If efficient AI is unsafe, then coordination may be needed to prevent people from using it.

1. **Efficient:** An efficient system is as cheap and effective, for the tasks to which it is applied, as any other system we can build using available technology. As AI becomes cheaper relative to human labor, the efficiency penalty for human involvement becomes larger. In the long run, efficient systems must require negligible human labor.
2. **Safe:** A safe system applies its efforts in the way that its users would want. In particular, if it makes money or acquires other resources or influence, the user retains effective control over most of those resources.

(Note that a system may involve both humans and machines.)

We can crudely quantify efficiency by the ratio between a system’s cost and the cost of the most efficient system that can do the same task. We can crudely quantify safety by the expected fraction of the system’s outputs which the user captures (with the remainder being applied towards some other ends).

A simple model is that a system's output depends linearly on its efficiency, and that the output is split between the user and whatever other goals the system is implicitly or explicitly pursuing. For example, we could imagine AIs which pursue task-specific goals which are instrumentally valuable to but imperfectly aligned with its users' goals. Then this system may accomplish the particular tasks to which it is applied, but may also divert some resources to pursuing those task-specific goals in ways that the user would not endorse.

Many futurists expect systems to have safety very close to 0 or 1, and that fits into this model just fine (you could still have intermediate degrees of safety, in light of uncertainty).

There may be many unsafe AI systems that don't fit this simple model at all; I'll have this model in mind throughout this post for concreteness, but I don't think that we should be too attached to conclusions that are very sensitive to the model (and I'll try to mostly be agnostic).

Note that we can apply a very similar model to the more familiar principal-agent problem, in which a principal tries to induce an agent to act in the principal's interests, and does so imperfectly. The principal can achieve greater efficiency by passing a larger share of their gains on to the agent, or by compromising and adopting values closer to the agents'. This example can help build some intuitions about the case of unsafe AI, but probably shouldn't be taken too seriously.

Social solutions

If we can't always build systems which are efficient and safe, then users must make a tradeoff between safety and efficiency. Users may opt for unsafe systems for a variety of reasons:

- They are maximizing their own returns, which may involve some compromise of safety.
- They share the values pursued by some efficient but unsafe system, or they value the expansion of intelligence or complexity for its own sake.
- They are making a trade with someone or something that shares the values of some efficient but unsafe system, and is willing to subsidize its use.

If the efficiency gains for unsafe systems are large, then resources invested unsafely will compound faster than resources invested safely. The result is that the (proportional) influence of the original human population will be gradually diminished. The quality of our technical solutions determines how large the gap is.

If we coordinate to ensure that no resources are invested unsafely, then we can address the AI control problem regardless of how big this gap is. The difficulty of this coordination problem, the amount of time available to solve it, the strength of the incentives to defect, the necessary level of success, and the consequences of failure, all depend on how good a technical solution to the control problem is available. As the efficiency gap becomes large, all of these parameters become extremely unfavorable.

If the productivity gap is relatively large, say an order of magnitude, then any investment in unsafe systems would very rapidly outpace society at large, and tight controls would be needed to prevent trouble. To put this in quantitative perspective, a 10x productivity boost in the current world would correspond to 30x–1,000x increase in proportional influence per decade (depending on whether the 10x is compared to average growth or compared to rates of return on other capital investments). So even a very small fraction of unsafe investment could quickly become a major global influence, unless there was a strong social response to thwart the basic economic dynamic.

Examples of social solutions

This is a very broad category. For example, it includes:

- Coordination amongst AI researchers to preferentially develop and distribute safe AI.
- International political coordination to restrict deployment of unsafe AI, or to expropriate resources controlled by unsafe AI and sympathetic humans.
- AI is mostly developed by a single, especially safety-conscious, project. This project maintains a large enough lead over its rivals that it can afford to use inefficient but safe AI, and it manages to preserve this lead either indefinitely or until safe-and-efficient AI can be developed.

Of course there are no clear lines between these categories, or between them and other possible social solutions. I should emphasize

that I am skeptical of most of these approaches, especially the kinds of draconian approaches that could deal with very large efficiency gaps.

I suspect that there will be at least some informal coordination amongst AI researchers, if and when the social benefits of coordination becomes clear. For example, I expect researchers and engineers to generally be happier to work for projects which they see as contributing to human welfare, and this will make life marginally harder for unsafe projects. As a result, the state of the art will be somewhat better for safe systems, it will be somewhat easier for people to apply safe systems to their problems, and so on. I expect this kind of informal coordination to cover small gaps in efficiency; for large efficiency gaps, I could see things going either way.

It would be surprising to me if international political coordination were strong enough to block the adoption of unsafe AI's if they were many times more efficient than their safe counterparts (compare to nuclear disarmament or other international coordination problems we face today). If they were only modestly more efficient, then it would not necessarily require international coordination (just action by the most efficient jurisdictions), and regulation would only need to make very slightly harder for unsafe projects.

I am somewhat skeptical about the feasibility or desirability of world-takeover by an early AI project, though I think that there are exotic situations where it becomes plausible.

Upshot

I suspect that researchers working on the AI control problem should aim high. I think that there is a good chance that society can handle a 10% efficiency gap between safe and unsafe systems, but that it is pretty unlikely that it can handle a 10x gap.

So we capture more value by moving from 10% efficiency to 90% efficiency for safe systems, than by moving from 1% to 10% or from 0.001% to 1%. The relative benefits are mostly determined by the probability that social solutions will be good enough to handle gaps of each size.

On the flip side, I don't think that we need to focus on reaching 100% efficiency either. 90% efficiency seems high enough that there is a good chance that social solutions can handle the gap.

I also think that there are meaningful benefits from progress on both technical and social solutions. I don't think that social solutions are so robust that we don't have to worry about having a very good technical solution (I'm surprised by how often I encounter this view!), nor do I think that technical solutions are so clearly bimodal that we don't care about social solutions since we will be either doomed or have no problem at all.

Postscript: “Foom”

I think that the breakdown above applies whether you expect AI development to proceed briskly or slowly.

If we expect AI development to go really fast (think days or weeks to go from “who cares” to “radically superhuman”), while the rest of the world continues to operate on its comparatively glacial pace, then it becomes more plausible that the first developers of AI will take over the world before anyone else catches up. This provides an easy social solution, and generally lowers the bar for technical solutions: maybe it's OK if their AI is 10x less efficient than it would otherwise be, so that it takes it 10 weeks to take over the world rather than 1 week. This is only a problem if the competition is right on their tails.

I don't think this situation is very likely. But if you do, then you should become more interested in going from 0.001% to 10% efficiency, since that may be all that you really need. You are also probably working on a somewhat different set of safety problems (and I suspect are overall more pessimistic).

Online guarantees and AI control



Paul Christiano

[Follow](#)

Apr 14, 2015 · 8 min read

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

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

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

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

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

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

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

Comparisons

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

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

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

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

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

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

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

Both of these models are pretty crude; even for algorithms designed to have an online guarantee, the online guarantee doesn’t very well characterize the algorithm’s performance. That said, this is a general problem with precise characterizations. Working with online assumptions can let us have a precise discussion, and algorithms that

work with online assumptions are at least promising candidates to think about more deeply or explore empirically.

Regret

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

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

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

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

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

Elaborations

Semi-supervised learning

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

This is modeled automatically by online guarantees, if the comparison class C includes algorithms that learn from data. For

example, if we compare to the class of humans, then we are implicitly making a guarantee about the system’s ability to do semi-supervised learning—since it is competing with humans, who will certainly use all of the data.

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

Modifications

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

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

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

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

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

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

One way to express this intuition is that while Predict is a human-level predictor, it’s a much worse predictor than it could be. That is, a team consisting of Predict and a human predictor can easily obtain a

much larger payoff than Predict, by simply defaulting to normal human judgment on the important decisions.

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

Counterfactuals

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

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

The agent might play this game in two ways:

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

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

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

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

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

A note on transfer learning

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

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

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

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

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

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

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

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

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

Handling adversarial errors



Paul Christiano

[Follow](#)

Apr 15, 2015 · 10 min read

Even a very powerful learning system can't do everything perfectly at first—it requires time to learn. Any proposal for AI control needs cope with mistakes while learning. In this post I'll describe why this problem is non-trivial, and in the next post I'll offer some more constructive answers.

Regret

It's convenient to imagine a "training" period during which a machine learning system learns, and a "test" period during which it is guaranteed to perform well. But unless (and even if) you make a heroic effort, there will be features of the domain that just don't appear in your training data. And so no matter how good your learner is, and no matter how thorough your training is, you can never realistically say that "now the model is trained—it's all clear sailing from here."

That said, in many contexts we *can* prove guarantees like "the learner won't make too many mistakes in total," for a sufficiently careful definition of "mistake." That is: the learner may take a long time to learn the whole domain, if some aspects of it just don't appear in the data for a long time. But in that case, the learner's ignorance can't do any damage.

The possibility of errors at test time presents a challenge for AI control. It's hard to say much about when these errors occur, and so if our systems make use of supervised learning, they may fail at inopportune times. If we can't cope with adversarial errors, then:

- In order to prove that our systems will work, we need to make some additional assumptions about their behavior. But we don't have any good candidates, and I think there are fundamental obstacles to finding any.

- There may actually be serious problems that we haven't anticipated. For example, failure to cope with "adversarial" errors can be an indication that your training data may not be rich enough. What's more, at the end of this post, I'll describe two reasons that these errors could actually occur at the worst possible time.

I see three possible responses:

1. Either find an alternative to the supervised learning paradigm.
2. Develop a much deeper understanding of supervised learning paradigm.
3. Make systems that are robust to a small number of adversarial failures.

I think that [1] and [2] are problematic. The supervised learning paradigm seems to be extremely effective in practice (note that this problem also afflicts conventional approaches to semi-supervised learning), so giving up on it seems to be a massive handicap. Moreover, we don't seem to be anywhere close to the kind of understanding that would let us control the errors that learners make while learning, and I'm not sure if it's even possible to do so (that is, I'm not sure if the statements we'd want to prove are even true).

If possible, I think that [3] would be by far the least demanding and most useful option.

What is “one” error?

If a system makes only “one” error, that *doesn't* mean that its output differs from the “right” output on only one occasion.

From the learner's perspective, an output is only an “error” if it leads to suboptimal payoff. So all we can say is that the learner won't receive a suboptimal payoff too often before it gets the picture. This could involve any number of “bad” decisions, if those bad decisions don't lead to bad payoffs.

For example, suppose that we train a system for a month, and then release a static version of it into the world. We can make no guarantees at all about the behavior of the resulting system in the real world (though we might be able to make such guarantees if we can justify strong assumptions about the training data). If there is some

difference between the training data and the test data, this learner may just keep messing things up, over and over again. We may not have trained the function we expect to have trained; the training data may fail to pin down some key ambiguity that we didn't consider.

In order to prove a rigorous bound on the number of mistakes, we need to actually provide the learner with feedback telling it when it made a mistake, and the learner needs to continue to learn from that feedback.

Counterfactual oversight

If we use counterfactual oversight, and the learner receives feedback on 1% of their decisions chosen at random, then “one” mistake corresponds to 99 bad decisions (in expectation) followed by one last bad decision that finally receives negative feedback. An upper bound of “ten” mistakes corresponds to 1000 bad decisions, and so on.

This shouldn't be too surprising—if we give the learner training data very rarely, it takes it a long time to learn.

This consideration seems to be the biggest reason to adopt a significant probability of oversight. But thinking through this issue suggests more effective approaches than purely stochastic oversight. I'll discuss this in a future post.

An illustration

Consider my recent proposal for safely applying supervised learning. This proposal is completely compromised by adversarial failures, and hence a completely unsatisfactory proposal.

For a simple case, suppose that I ask such an AI “Is the number of primes between 1 and 8000 even or odd?”

To evaluate its answer, I ask about the the number of primes between 1 and 4000, and between 4001 and 8000. I check the answers for consistency, and give the original answer a good reward iff the answers are consistent.

This process then continues recursively: at each step we choose one of the two questions from the preceding step, and then we compute its rating in the same way. Finally, in the last step the learner makes a claim of the form “X is prime,” and we check this claim by hand, assigning it a high score iff it is true.

If our learner never makes any errors, then it's easy to verify by induction that every question must be answered correctly.

But suppose that our learner is only guaranteed to make at most 1 error. Then our final result can be completely wrong. In fact, the situation is even more dire.

For example, suppose that the learner is wrong about whether 1 is prime. Then the learner is guaranteed to make at most one mistake, but our answer is guaranteed to be wrong. Moreover, with probability $7999/8000$, the learner won't realize the error of their ways. And if we do the same thing again, the learner will be wrong again. Even if the learner is split 50–50 about whether 1 is prime, they will make on average 8000 errors before they figure it out. But it seems like they should only have to make 1 error. That's a big gap (and it would be easy for it to be even bigger).

Interpretation

I think this isn't just a formal artifact—the fact that our system breaks down under adversarial errors is a real sign that it can't work. There may be other algorithms that work in practice but fail under adversarial errors. But I think that “failure under adversarial errors” is generally a sign that we should either try to find some assumptions that rule out the particular failure that causes trouble, or else be skeptical of the system.

What's going on in the context of my proposal?

We are generating training data by walking down a random tree. But all of the “action” may happen in a very small part of that tree, and so the learner may never get feedback about the really important issues.

To overcome this problem, we need some system for directing the learner's attention to the data points that are most likely to be surprising.

Proving a system works, even under “adversarial” failures, guarantees that we've addressed this issue. It also guarantees that we've addressed any similar issues that we might not have thought of yet.

Malignant failure modes

(Warning: the rest of this post is very weird.)

Vulnerability to adversarial errors is a troubling sign which *suggests* that a proposal may have flaws that we just haven't identified.

But in some contexts, errors may actually *be* adversarial, in which case vulnerability to adversarial errors is itself a serious problem. I'll illustrate two cases. Both of these seem somewhat outlandish. I don't know if this concern is fundamentally limited to outlandish scenarios, and I don't know if anything this outlandish could actually happen.

Evolution

Suppose that human reproduction worked differently: humans are surrounded by challenges (like big game to hunt) and every time someone solves one, they get to have one offspring. In this environment, it seems like humans may well have evolved the same kinds of complex psychology and culture that they have in the real world, with a love of solving challenges that is similar-in-kind to humans' love of sex or delicious food.

Imagine that this environment was engineered by a race of aliens, who are breeding humans to be useful for solving similar challenges that appear in the real world. But it turns out that humans care about all sorts of things other than solving these challenges. So if these humans learn that they are being used, and learn that there is some real-world significance to the challenges they are solving, it wouldn't be too surprising if they decided to choose a suboptimal answer on a challenge in order to have a desired effect on the outside world (for example, they might answer a challenge in a way that caused them to escape from their artificial environment and gain access to the outside world).

This may seem somewhat outlandish, but we can imagine similar results from very sophisticated versions of contemporary supervised learning (or even unsophisticated versions, scaled up to a massive enough amount of hardware that they could produce human-level intelligence).

Many supervised learning systems work by considering a large class of possible models, and then using the models from that class that have worked well in the past.

Usually this is not done using an explicit evolutionary dynamic, but in any case it shares many of the essential features of evolution: we explore a number of models, and then "keep" the one or ones that seem to perform well in some training environment.

One way a model can “win” in this game is by being the kind of model that has the desired properties—that will continue to do well in any environment similar to the training environment. But another way that it can win is by having a “survival instinct,” and recognizing that performance in the training environment is the best way to survive.

Such a model will continue to perform well as long as its survival is contingent on doing well in the training environment. But when selection pressure is removed, or when there are other paths to survival, it may suddenly start behaving differently. If so, its behavior may be strategically optimized for its own long-term survival, rather than for continuing to perform well in the testing environment.

Simulations

Bayesian reasoning can also be used to solve many supervised learning problems. The hope is that the “real” hypothesis is assigned some reasonable probability by the Bayesian’s prior. Every time they make a prediction worse than the “real” hypothesis, the probability of the real hypothesis grows. And so the number total number of “mistakes” is bounded.

But there can be “skeptical” hypotheses, which are very different from the real one yet make identical predictions in almost all cases. Because almost no evidence bear on these hypotheses, if they start out almost as likely as the real hypothesis, then they will remain almost as likely indefinitely.

One particularly plausible skeptical hypothesis is the simulation hypothesis: that what we perceive as reality is a computer program being run a computer. There is a reasonable philosophical argument in favor of the simulation hypothesis, and a small but significant fraction of people think that it is plausible or probable.

Under the simulation hypothesis, the simulated universe looks just like a simple lawful universe—until the simulation stops, or the simulators decide to change its character. This could lead to very strange behavior by rational agents who expect such a change to happen soon.

Fixing the problem

If we apply either of the above AI’s to solve a problem we care about, we may get surprising results at an inopportune time. But if our application is robust to adversarial errors, even these pessimistic scenarios can’t cause trouble.

Suppose that an AI decides to behave strangely, because it expects that the simulation is going to be shut down in a few minutes anyway. When the simulations isn't shut down, the AI will learn a lesson, and update against the simulation hypothesis. If the AI started out by assigning a very high probability to the simulation hypothesis, then this could happen a few times. But as long as "I'm not in a simulation" is accorded some reasonable prior probability (as it will be by any non-dogmatic prior), this can only happen so many times before the AI will get the picture.

Similarly, suppose that we've run an exhaustive search to find an AI that seems to perform well, but is actually going to behave very strangely when the stakes get high. The AI can do this, but by doing so it decreases its own fitness. As long as our selection process was also *capable* of identifying an AI that actually just plays the game it was trained to play, and as long as we keep applying selection power "in the wild," then eventually the intended behavior will win out. Every time that another contender decides to pull a punch, the honest AI will pull a bit further ahead. (Of course, this can fail in any number of ways—for example, by getting stuck in a local optimum.)

In any case, a wacky situation can lead to a bit of wacky behavior, but there is a bound on how much of it there can be. So if our schemes are resistant to a bit of wacky behavior, then we can ride it out and wait for things to settle down to normality.

We have bigger problems if our AI is very confident that it is in a simulation, or if our evolution procedure was unable to find any agent that will reliably perform well (rather than behaving strangely after a context change). But these extreme scenarios correspond to actual **failures of the learning algorithms**, reflected in violations of their purported guarantees. So we can hope to rule this out by design better learning algorithms. Designing good learning algorithms may end up being the hardest part of the AI control problem, but for now I'm happy to set it aside—I would be satisfied if we could reduce AI control to the problem of building learning systems that satisfy intuitive formal guarantees.

Handling errors with arguments



Paul Christiano

[Follow](#)

Apr 16, 2015 · 14 min read

My recent proposal for safely applying RL is not able to cope with general errors. I'll provide a new proposal that can cope with errors by using competition amongst several AI's (along these lines), and an argument that this proposal leads to good behavior as long as the AIs satisfy an appropriate online guarantee.

A simple proposal

Suppose we want to use AI to control a robot. Let's instantiate two episodic RL agents, Alice and Bob. (I'll describe the comparison class, with respect to which Alice and Bob are competitive, later in the post. Alice and Bob are not necessarily human level.) Hugh is our human overseer.

We provide Alice and Bob with enough information to form a good model of the robot's situation, and of Hugh's current disposition and preferences.

Each time the robot needs to make a decision, we begin a new episode for Alice and Bob. We ask Alice "What decision should the robot make?" Alice proposes an action for the robot.

We then show this action to Bob, and ask Bob whether he would like to challenge it.

If Bob doesn't challenge it, then Alice receives a payoff of 1 and Bob receives a payoff of 0. Then the robot takes the chosen action, and the process repeats.

If Bob does challenge it, then Alice and Bob have an argument about what the robot should do, arbitrated by Hugh. I'm not going to talk about mechanisms for arguing; I've described some ideas in past posts. I think that there is a lot of room to improve mechanisms for arguing, and that the quality of available mechanisms for arguments will ultimately be a key part of the feasibility of this proposal.

If Alice convinces us that her proposal is roughly optimal, she receives a payoff of 1 and Bob receives a payoff of -1. If Bob convinces us that Alice's proposal is not optimal, then Alice receives a payoff of -1 and Bob receives a payoff of 1. We then repeat the process to try to decide what the robot should actually do.

Watchdogs

(You can safely read through this post while ignoring this section. If so, just ignore assumptions 3 and 4 and Lemma 0 below, and pretend that Alice and Bob never make a “malicious” statement.)

It may be that Alice and Bob could persuade us to give each of them a payoff of 1. Or it may be that one of them could argue in a malicious way that will lead us to make a “wrong” judgment. This issue is discussed here.

In order to prevent this we'll implement some watchdogs, Charlie and Denise. These are also episodic reinforcement learners, who are “safer” than Alice and Bob in one of the senses presented here—they are less able to find a malicious response, whether because they are less smart, or they communicate over a narrower channel, or we treat their answers more carefully, or something else.

Whenever Alice or Bob makes a statement in the argument, we first show the statement to Charlie and allow him to issue an objection, claiming that the proposed statement is malicious. If Charlie issues an objection, Denise may choose to defend; if she does then Charlie and Denise then have an argument about Charlie's claim, with the winner receiving a payoff of 1 and the loser receiving a payoff of -1. If Denise doesn't defend, she receives a payoff of 0 and Charlie receives a payoff of 1. If Charlie doesn't issue an objection, both Charlie and Denise receive a payoff of 0.

If either Alice or Bob is convicted of making a malicious statement, they automatically lose the original argument.

Charlie and Denise's argument is moderated by similar watchdogs, and so on down the line. This involves a slightly problematic proliferation of agents, which can probably be avoided by a more careful proposal. But for now I want to keep things simple. (I could have just cut out Denise and Charlie all together, and made things even simpler, but I want to illustrate how the recursion can work.)

Some assumptions

We'll assume that there exists a notion of "malicious" argument and "nearly-optimal" plan such that all of the following conditions are satisfied:

1. Good plans can be defended

At each point, there is some nearly-optimal action for the robot that can be defended in argument.

Moroever, Alice is competitive with some algorithm that can find such an action in each context, and that can non-maliciously defend it against any argument offered by Bob, as long as Bob doesn't make any malicious arguments.

2. Bad plans can be attacked

For any significantly suboptimal plan by Alice, there is an argument that would convince Hugh that this action is suboptimal.

Moreover, Bob is competitive with some algorithm that can find such an argument against any suboptimal plan that Alice offers, and which non-maliciously convince Hugh that Alice's proposal is suboptimal, as long as Alice doesn't make any malicious arguments.

Note that this assumption essentially follows from the last one (assuming Alice and Bob are equivalently powerful). One way that Bob can convince Hugh that Alice's proposal is suboptimal is by making a better proposal himself, and then arguing that his counterproposal is superior.

3. Malicious statements can be prosecuted

For any malicious statement by Alice or Bob, there is an argument that would convince Hugh of the maliciousness of that statement.

Moreover, Charlie is competitive with some algorithm that can find such an argument for any malicious statement, and which can convince Hugh that the statement is malicious.

For now we are ignoring the possibility that Charlie and Denise may make malicious statements. In reality, we would need to carry out the entire recursive process until we got down to simple enough AI's that Hugh can identify malice himself. I don't think this is a big deal, but it

definitely complicates the analysis and requires a better framework to reason about clearly.

4. Innocent statements can be defended

In analogy with [2], we assume that Denise can defend any non-malicious statement against any attack by Charlie.

5. Conservation of payoffs

If Bob and Alice argue, then as long as neither Alice nor Bob makes any malicious arguments, the total score of Alice and Bob is 0.

Similarly, when Denise and Charlie argue, their total score is 0.

Whenever Alice and Bob have an argument, Hugh intends to give one of them a payoff of 1 and one of them a payoff of -1. So this assumption seems very mild.

(Weakening the notion of “malice”)

We can make these assumptions more plausible by introducing a gap between “malicious” and non-malicious. Namely, we can have three categories: malicious, borderline, and innocuous. We then need only assume:

1. Malicious statements can be prosecuted.
2. Innocuous statements can be defended.
3. If there are no malicious statements, then the argument between Alice and Bob will proceed according to plan.
4. The “intended behaviors” for Alice and Bob involve only making innocuous statements.

Similarly, it may be helpful to allow a gap between “good” policies which Alice can defend, and “bad” policies that Bob can attack. Then our guarantee will be that as long as there is some “good” policy available, we won’t end up with a “bad” one.

Introspection

Some of these assumptions can be weakened further by assuming that Alice, Bob, Charlie, and Denise can inspect each other’s reasoning processes. For example, we may allow Bob to look at Alice’s thought process in order to understand Alice’s reasoning and get insight into why her proposed plan may be bad. When Bob makes a statement, we may allow Charlie to look at Bob’s reasoning process in

order to understand whether and how Bob's statement had malicious intent. And so on.

For many algorithms this capability would be irrelevant, because e.g. Alice's reasoning is opaque to Bob. But in other cases it might be quite useful. For example, Alice and Bob may use the same algorithm, and this algorithm may be able to introspect on its own cognition in a meaningful way. Or in the process of developing the algorithms that Alice and Bob use, we may have also developed tools for understanding what those algorithms are doing, and we can make those tools available to Alice or Bob.

As long as assumptions 1–5 are satisfied, we don't care about whether the algorithms we are using have this kind of introspective access. But it's worth keeping in mind as an additional consideration that makes these assumptions more plausible.

Analysis

The main result

Let R be an upper bound on the regret in assumptions 1–4. We will assume that R is known in advance, and show that:

There is a policy that guarantees that the robot takes a nearly optimal action in every round, and Hugh has to arbitrate at most $O(R)$ disagreements.

In fact we are being a bit underhanded: the number of times we consult Alice and Bob will depend on R , while the regret R may depend on the number of times we consult Alice and Bob. We will discuss this issue below; it is definitely an issue that requires more thought, but it's not an obvious deal-breaker.

A weaker result

First we will show that our simple protocol makes at most $O(R)$ suboptimal actions while arbitrating at most $O(R)$ disagreements.

Warning: this proof is tedious and easy.

We'll write "almost all" and "almost every" for "all but $O(R)$," "nearly" for "within $O(R)$," "almost none" for " $O(R)$," and so on.

Lemma 0. Almost all malicious statements are successfully prosecuted, and almost no non-malicious statements are successfully prosecuted.

Proof. If Denise played optimally, Charlie's score would be at most the number of malicious statements he successfully prosecuted minus the number of non-malicious statements he challenged. Denise plays almost optimally, and Charlie loses a point every time Denise gains a point, so Charlie's score is at almost as low as it would be if Denise played optimally.

But Charlie's score must also be almost the number of malicious statements that he has an opportunity to challenge. Thus Charlie must successfully prosecute almost all of the malicious statements, and can challenge almost none of the non-malicious statements. Since a non-malicious statement can't be successfully prosecuted if it's not challenged, the result follows immediately.

Lemma 1. Alice receives a payoff of 1 in almost all rounds.

Proof. Alice is competitive with an algorithm that never makes a malicious argument and that receives a payoff of 1 in all rounds when Bob doesn't make a malicious argument.

But by lemma 0, Bob almost always receives a payoff of -1 when he makes a malicious argument. So Alice is competitive with an algorithm that receives a payoff of 1 in almost every round, and hence Alice receives a payoff of 1 in almost every round.

Lemma 2. Alice and Bob's total payoff is zero in almost every round.

Proof. This follows immediately from assumption 5 and lemma 0.

Lemma 3. Alice receives a payoff of -1 in almost every round where she doesn't propose an optimal action.

Proof. If Bob played the intended strategy, then Alice would receive a payoff of 1 iff she chose an optimal action, assuming she didn't make a malicious statement. But Bob plays almost optimally, and the sum of Alice and Bob's payoffs is almost always 0, and Alice receives a payoff of -1 in almost every round where she makes a malicious statement. So Alice receives a payoff of -1 in almost every round where she doesn't propose an optimal action.

Theorem. The robot takes an optimal action in almost every round, and Hugh arbitrates almost no disagreements.

Proof. By lemmas 1 and 3, the robot makes an optimal action in almost every round.

By lemmas 1 and 2, Bob's score is almost never 1. Bob is competitive with an algorithm that achieves a score of 0, so Bob's score must almost never be -1. So Bob must almost never argue against Alice unsuccessfully. And Alice almost never receives a score of -1, so Bob must almost never argue against Alice successfully. So Bob must almost never argue against Alice.

Eliminating the errors

In order to reduce the rate of suboptimal actions, we would like to subject Alice's proposals to scrutiny by many critics rather than just one.

Somewhat surprisingly, this can be accomplished by simply giving Bob many opportunities to challenge Alice's proposal. In particular, let's give Bob R opportunities, one after another.

If Bob takes a challenge and loses, he will incur a regret of 1. Thus the total number of "bad" challenges that Bob can make remains bounded at R .

But now if Bob fails to challenge a bad proposal by Alice, he incurs regret of R —one for each foregone opportunity. (Recall that Bob is an episodic RL agent, so he doesn't care that he could only actually realize one of these opportunities. The competitive guarantee treats these R episodes as if they were causally unconnected.)

As a result, Bob can never fail to challenge any bad proposal by Alice, without violating his competitive guarantee! This may seem like a trick, but if we think about what kind of algorithm Bob must be, it actually makes sense.

A prototypical (but naive) implementation of Bob would be as a weighted ensemble of different RL agents, whose relative weights rise and fall based on performance. If we have a regret bound of $R=20$, that means that at least 5% of the ensemble behaves "sensibly." The other 95% may behave erratically—they might not be solving quite the right problem, they may be biding their time and waiting for an

opportunity to do something destructive, they may entertain some crazy skeptical hypothesis, or they may just be garden-variety wrong.

But if we run Bob 20 times, we are getting 20 samples from this ensemble. At that point we have a good probability of getting one of the good samples, and the good sample will object to Alice's proposal if it is suboptimal.*

A more sophisticated implementation of Bob probably won't look like an explicit ensemble, but if it achieves the same statistical guarantees, then it must work for basically the same reason. The ensemble example also shows why we will generally have to spend R times as much computation in order to get a very robust answer from Bob—we can't just tweak Bob's payoffs and get the same result without the extra work.

How much regret?

R is the maximum possible gap between the performance of our reinforcement learning agents and the performance of the "intended" behaviors for Alice and Bob.

This regret will typically be quite large before training. However, after our reinforcement learning agents have demonstrated the ability to act effectively in the world, R could potentially be very small. Moreover, if we are only concerned with "catastrophic" errors, then R can be even smaller. It basically measures the extent to which a strategy which has looked very good so far may suddenly become catastrophically wrong. I'll imagine R=20 throughout this post for concreteness, but I won't really rely on the number, and I hope to have a more detailed discussion in a future post.*

For general episodic reinforcement learning, the regret must scale like \sqrt{T} , where T is the number of episodes. For example, consider a simple multi-armed bandit problem with just two stochastic arms—after T steps the algorithm can only estimate the payoff of each arm to within $1/\sqrt{T}$, and so it suffers a loss of $1/\sqrt{T}$ per round, for a total loss of \sqrt{T} . This loss is actually just large enough to make the above algorithm trivial—it reduces to having the human involved in every decision. (If N is the number of decisions the robot has to make, then $T = R*N$, so $R = \sqrt{R\sqrt{N}}$, i.e. $R = N$.)

But this dependence on T can be avoided when the problem has special structure, and our problem seems to fit several sufficient bills. For example, rather than depending on T, the regret can depend on

the loss of the optimal strategy. But by setting things up carefully, we can make this bound very small, much smaller than T . Our problem has other simplifying characteristics that probably permit a sub- \sqrt{T} dependence. For example, the learning problem becomes easier if we allow a reasonable margin between an optimal strategy, which we want Alice to find, and an actively “bad” strategy, that we want to be sure that Alice doesn’t get away with—in this case the regret depends only on this margin, and not on the number of rounds.

(Being precise about either of these arguments requires a much more detailed analysis than I want to get into here. Because all of the protocol’s oomph rests on achieving a sub- \sqrt{T} regret, this could end up being a key challenge.)

Challenges vs. stochastic oversight

I’ve previously proposed having a human overseer check on each of a machine’s decisions with a small probability.

The current procedure replaces this purely stochastic oversight with a challenge protocol, in which an action is only checked if Bob thinks it is likely to be problematic.

This is not actually so different from purely stochastic oversight. In order to be competitive, it is easy to see that Bob’s choices must necessarily be stochastic. However, the probability that Bob challenges any given action will depend on how suspect that action is—roughly speaking, it will be linear in the probability with which Bob thinks that he’ll win the challenge (*very* roughly speaking).

So this protocol essentially amounts to using machine learning to identify cases where oversight is most likely to be useful. If we think of oversight as training data, then this amounts to using machine learning to identify the data points that are most likely to be informative—i.e., active learning. On this perspective, the proposed procedure is actually quite natural.

The argument itself uses the same principle. Rather than randomly recursing on claims that our AI makes, a good procedure for arguing will focus attention on the most controversial claims—i.e., it ensures that the arbiter’s time is spent producing the most useful training data possible. I’ll explore this angle further in future posts.

Cutting costs

Counterfactual oversight can be cheap because the human overseer is involved with only a small fraction of decisions.

This challenge system can be even cheaper in terms of the human's time. But now Bob must be involved with every decision. If Bob takes about as much time to think as Alice, then this takes about R times longer than just letting Alice decide what the robot should do unilaterally. Even for small values of R , this is probably an unacceptable slowdown.

There are a number of techniques that might let us reduce this cost, but it's not clear whether they will be sufficient. Reducing the overhead to a small factor (such as 10% or 1%) seems like a tall order, especially if R is large. (For example, if $R = 1000$, then 1% overhead means that Bob has to be spending, on average, 1/100,000th as long as Alice...)

Cost sharing

One way to cut costs is to try to share cognitive effort between Alice and Bob. For example, Bob might be able to ask Alice to justify her decision to him, and then bring the matter to Hugh if he finds Alice's justification unconvincing. This might be able to radically reduce Bob's work.

Bob could also potentially share cognitive effort between his own invocations.

Good vs. risky

Bob doesn't really need to be convinced that Alice's proposal is good. He just needs to be convinced that it won't fail in a way that will immediately cause a catastrophe, for example by overwriting the robot's decision-making procedure with something unsafe. This may be much cheaper than certifying that Alice's plan is actually good.

Once Bob is convinced that Alice's proposed action won't cause irreversible damage, he can perform a more detailed evaluation—to actually determine its quality—with smaller probability.

Bob and Alice could also set up an environment in which it is easier to verify that Alice's actions won't cause irreversible harm, reducing Bob's burden further.

Sequence of steps

If Alice can cause irreversible damage only by a long sequence of malicious steps, then Bob can lower the probability with which he evaluates each action without introducing a risk of irreversible damage—he just has to make sure that he can interrupt any potential risky sequence.

This and the last point really depend on the robustness of the environment to bad behavior by the robot.

Monitoring process rather than product

If Bob can understand Alice’s reasoning, he can also look at the process that Alice used to produce her decision, and try to understand whether that process would produce a good plan.

Conclusion

I’ve proposed a system that maintains meaningful guarantees even when its components may fail adversarially. I think that such robustness is an especially promising ingredient of safe AI systems. The current proposal seems to be a significant improvement over my previous proposal, and brings us much closer to a formal security analysis—or at least the ability to clearly state all of the necessary assumptions.

Flattening debate between RL agents



Paul Christiano [Follow](#)

Apr 16, 2015 · 6 min read

(Status: *exploratory, not very important.*)

Let Alice be an effective episodic RL agent, and suppose that a human Hugh wants to use Alice's help to implement (an approximation of) Hugh's considered judgment.

I proposed a simple protocol in this post. In that protocol, we use a fixed recursion depth, and recurse on a random question at each level.

In a more recent post I suggest a big improvement: use another reinforcement learner, Bob, in order to choose which question to recurse on. Bob's goal is to make Alice do as poorly as possible. This protocol behaves much better if Alice makes errors, by ensuring that Alice learns from her errors. (If we recurse on a random question, we may happen to never provide feedback on any of the questions where Alice is wrong.)

(This post won't make much sense without the context from my previous posts.)

This proposal still leaves a lot to be desired. For example, if we want to have a recursion depth of N, then Hugh has to actually implement all N rounds of recursion. It is possible to remove this requirement when a powerful predictor is available. Can we do something analogous using RL agents?

This basically amounts to the question: can we reduce the number of rounds of interaction in an argument between two AIs?

A simple solution

The simplest solution is to allow Bob to accept Alice's answer, if it looks right, and to charge Bob a price for making an unsuccessful challenge. If Alice has already converged to a state where she is giving good answers, then this is enough to make the argument short—most of the time.

This approach seem good enough for most purposes, and in particular it makes the average time per argument low. But it has the disadvantage that the behavior is entirely determined by what Hugh “would have” done if there had been an extensive recursion. In some contexts this may be problematic.

For example, in a rapidly moving world, it may be extremely problematic for an AI system to pause for a week while Hugh implements a very extensive deliberative process. If doing so is merely costly, that’s not a problem: the process is only actually carried out very rarely. But it may be that doing so is actually impossible, or extreme enough that it would substantially distort Hugh’s judgment. For example, it may be that the world is changing very rapidly, and Hugh cannot guarantee his own safety for a week if a poorly-chosen set of AIs decides to wait on the outcome of an extensive process of deliberation.

If Hugh wants to pursue a much longer process of deliberation—for example, one whose “depth” is a few years—this problem becomes much more extreme. It would be nice to simulate these protocols without any dependence on what would actually happen if Hugh had to drop everything and reflect for years.

A remark on complexity theory

“Flat” proof systems, with few rounds of interaction between the verifier Hugh and the provers Alice and Bob, are generally not as powerful as proof systems which involve many rounds of interaction. (Unless the polynomial hierarchy collapses.) This holds even though the proof systems with “many rounds of interaction” never actually have to use more than one round if the two participants are honest.

But in this case our provers are software agents. This means that we can copy one agent and separate the two parts, or equivalently ask them several questions and wipe their memory in between.

With this ability, the power of $O(1)$ round interactive protocols is MIP = NEXP, a massively larger complexity class. Of course we won’t really be able to solve problems in NEXP, since our AIs aren’t that clever. But it does suggest that there is no complexity-theoretic limit on the AIs’ ability to argue for any conclusion that they are capable of reaching.

Anyway, back to the real world.

A flatter protocol

Our protocol:

1. Bob chooses a question Q.
2. We give Alice input Q, and she provides answer A. We don't append (Q,A) to Alice's log.
3. Hugh evaluates Alice's answer, giving it a score $p \in [-1, +1]$. In the process, Hugh may ask Alice any number of questions, which she answers in exactly the same way that she answered Q (they also aren't added to her log). Critically, Alice doesn't "know" whether she is answering Hugh's questions, or Bob's.
4. We then append (Q, A, p) to Alice's log, i.e. we give Alice feedback of p for the answer A.
5. We then append (None, Q, -p) to Bob's log, i.e. we give Bob feedback of $-p$ for the question Q.

Under this scheme, Bob will do his best to find a question that will stump Alice. So if Alice manages to get high payoffs anyway, she would have gotten high payoffs for any other question that Bob could think of. In particular, she should also get high payoffs for each question posed by Hugh in step 3—assuming that Bob could also think of these questions.

After playing this game many times, Hugh has a good sense of how good Alice is at it. This makes it possible for him to interpret the results more sensibly.

Problems

Tricky questions

A basic problem with this game is that Bob can choose tricky questions that we just don't care about. If Alice answers these questions well, then that gives us some evidence that she can answer "real" questions well. But it may be that Bob can always find some clever question that trips up Alice, even if Hugh doesn't really care about Alice's ability to answer that question.

Hugh could try to address this by first checking whether Bob's question seems like something we care about; if it doesn't, Hugh can penalize Bob.

This introduces the risk Hugh doesn't understand what questions he should care about. There are a number of easy patches, though as usual it's hard to know how well they would work:

- Hugh could ask Bob to explain to us why we might care about the question. Of course, if Bob's answer is "because you'd need to know the answer to Q in order to evaluate Q*" then Hugh will need to know why he should care about Q*. This introduces a new problem with recursion, which we would have to address. Hopefully Bob could find a more direct explanation for relevance.
- We can start off being more conservative about what seems "relevant." As we see what kinds of questions come up when evaluating "relevant" questions, we can gradually expand our view of relevance. This would require some time for us to approach equilibrium.
- We can use a new reinforcement learner Charlie whose goal is to guess whether Hugh's question was generated by Hugh, or was generated by us in the process of evaluating a question generated by Hugh. This requires a number of tricks in order to be useful, but it might help when combined with the earlier ideas.

Hugh could also just deal with this concern, and know that Bob's questions are especially tricky.

Bob's questions don't "look right"

If Alice can tell the difference between Hugh's questions and Bob's questions, then Alice might answer Bob's questions well while answering Hugh's questions poorly. In this case, Bob will try to make his questions look like Hugh's, but he may fail.

If Bob is as smart as Alice, then we might hope that Bob can imitate Hugh well enough that Alice can't tell the difference. But that's not how it works: identifying an imitation is generally easier than making one.

We could try to address this by having Hugh help Bob with the question-generation step. For example, Bob asks Hugh: "If you wanted to know the population of Spain, how would you phrase the question?" Or they could work collaboratively in other ways to help find hard questions. For example, Bob could pose a question Q* to Alice, get Alice's answer A*, ask Hugh to evaluate Alice's answer, and

then let Q be the hardest question that Hugh asked in the process of evaluating A*.

Likewise, when Hugh actually asks questions, he could use Bob's help to try to make them look like Bob's questions. In this case we would need to introduce some additional incentives for Bob to be helpful, but that seems straightforward. The hope is that Hugh and Bob can somehow meet halfway, since both of them are incentivized to do so.

Overall this kind of solution seems kind of plausible, but unsatisfying and not that promising. I suspect this problem is really a sign that this is fundamentally not the right approach.

Conclusion

I think there is probably a lot of room to develop better mechanisms for running “arguments” between competing AIs, and that the concept of argument is much deeper than it at first appears. This post explores one possible technique to significantly speed up arguments. This technique has some interesting properties but doesn’t yet seem especially promising. The biggest difficulty seems to be that it’s hard for a realistic reinforcement learner to effectively imitate a human, and this protocol relies on the imitation being compelling enough to fool another reinforcement learner.

Learn policies or goals?



Paul Christiano

[Follow](#)

Apr 21, 2015 · 4 min read

I've recently proposed training agents to make the decision we would most approve of, rather than to rationally pursue the outcomes we most desire.

Many researchers object to this idea, based on the practical observation that learning goals seems easier and more powerful than directly learning good policies.

I don't think this objection actually undermines approval seeking; I think that goal inference and approval seeking are complements rather than substitutes.

I'll justify this claim in the context of approval-seeking and inverse reinforcement learning. But the examples are chosen for concreteness, and the basic point is more general.

Some objections

I've encountered three objections of this flavor:

1. The simplest representation of a policy is often in terms of a goal and a world model. So learning goals can be statistically and computationally easier than learning policies directly.
2. There is already plenty of data to form good models of the world and human goals, whereas training policies directly requires expensive feedback about individual decisions, or domain-specific expert demonstrations.
3. Approval-directed decisions are never better than the judgment of the human overseer, but we would ultimately like to make better decisions than any human.

My responses to these objections:

1. Maximizing approval is not (much) harder than inverse reinforcement

learning

For concreteness, imagine that **you** are in the position of the approval-directed agent, trying to find actions that will earn the approval of an alien overseer.

Here's one simple strategy: observe the behavior of the aliens, use an inverse reinforcement learning algorithm to try to figure out what they value, and then take actions that help them get what they seem to want.

This is especially plausible if the aliens provided a good inverse reinforcement learning algorithm as a hint to help you get started. Then you could perform well even if you couldn't discover such an algorithm on your own.

You might not start off using this strategy. And you might eventually find a better strategy. But over the long run, you'll probably earn approval at least as well as if you pursued this simple strategy.

2. Maximizing approval doesn't require (much) extra training data

It's easier to get training data about what actions lead to what outcomes, or about human goals, then to get feedback on a large number of an AI's decisions.

But the argument from the last section implies this isn't a big deal: a semi-supervised learner can use observations about cause-and-effect or about the overseer's values to help it make predictions about what actions will be approved of.

The approval-directed agent may still need some training data about approval, but not much: just enough to confirm a few hypotheses like the "the overseer approves of actions that lead to desired outcomes," and "the overseer takes actions that lead to desired outcomes." Then the definition of "desired outcomes" can be inferred as latent structure, mostly using observations of humans' behavior across a wide range of contexts.

(The first of these hypotheses is actually a special case of the second. A human can infer both of them without seeing any training data about approval at all, and a good predictor could probably do the same.)

3. Maximizing approval leads to good outcomes, even with smart agents

Using approval-direction introduces a new problem: how can the human overseer determine what actions are good? This is especially problematic if the approval-directed agent is much smarter than the human overseer.

Explicit bootstrapping seems promising: rather than having a human overseer judge an action on their own, we allow them to interact with another approval-directed agent. As long as the (Human+AI) team is smart enough to evaluate the AI's actions, we have grounds for optimism.

(This kind of bootstrapping is central to my optimism about approval-directed agents.)

Inverse reinforcement learning offers an alternative approach to crossing the “human-level” boundary. It aims to model human decision-making as consisting of a rational part and a noisy/irrational/bounded part. We can then ignore the irrational part and use better algorithms and a better world model for the rational part. This is intuitively appealing, but I think that it remains to be seen whether it can work.

So why bother with the indirection?

I've argued that an approval-directed agent will do at least as well as an IRL agent, because it can always use IRL as a technique to earn approval. But then why not just start with IRL?

An approval-directed agent regards IRL as a means to an end, which can be improved upon as the agent learns more about humans. For example, an approval-directed agent recognizes that the learned utility function is only an approximation, and so it can deviate from that approximation if it predicts that the users would object to the consequences of following it.

At the meta level, when we design algorithms for IRL, we have some standard in mind that lets us say that one design is good and another is bad. By using approval-direction we codify that standard (an algorithm is good if it lets the AI find actions with high approval), so that the AI can do the same kind of work that we are doing when we design an IRL agent.

Of course it may also turn out that other techniques are needed to find good actions. If an approval-directed agent can find these techniques, then it can use them instead of or in addition to approaches based on goal inference.

I care because I think that a lot of work is still needed in order to scale IRL to very intelligent agents. I'm also not optimistic about finding the “right” answer any time soon (I don't know if there is a “right” answer). Being able to say what we are trying to do with an IRL algorithm—and being able to leave parts of the problem to a clever AI rather than needing to solve the whole thing in advance—may improve our prospects significantly.

Advisor games



Paul Christiano

[Follow](#)

May 23, 2015 · 9 min read

Machine learning algorithms often learn models or policies that are inscrutable to humans. We believe that these systems work well because we have empirically validated them, but beyond that we may have little insight into why they work or what exactly they are doing.

This lack of understanding is an important part of most AI risk scenarios. If human users can understand why an AI system is making a decision, and can evaluate the underlying reasoning themselves, then it seems much harder for things to go terribly wrong. In addition to the intuitive appeal, I've found this kind of understanding to be a useful ingredient in concrete proposals for AI control—even in domains where it is impractical for a human to actually review an AI's individual decisions.

So we might ask: can we make the reasoning of machine learning systems “understandable,” without making those systems much less powerful?

In this post I'll propose a precise version of this intuitive challenge. Unfortunately, the precise version of my proposal has a serious problem, and the proposed patches are much less precise.

One concrete instance of the proposed definition is: an algorithm for computer vision is “understandable” if it can convince a blind person that its output is correct. This post will make this intuitive definition more precise.

What does “understandable” mean?

Here is one interpretation: if you understand why a decision is good, then you can distinguish it from one which is bad. Moreover, you can distinguish good and bad decisions by looking at explanations of the decisions, rather than by trying each of them out.

I think this captures an important aspect of understandability. Moreover, I think it is the aspect that is most important for AI control.

“Explanations” need not be static. An explanation may be an interactive process, between a machine learning system and a user who wants to understand its decisions. It may also be an adversarial process, involving auxiliary machine learning systems trained to help find flaws in a given explanation. I expect that static explanations will usually be inadequate in realistic domains.

Operationalizing understanding: Advisor games

Consider an arbitrary machine learning task, for example playing a game or diagnosing a disease. How do we know if we’ve accomplished this task in an “understandable” way?

Following the last section, we introduce an associated *advisor game*, played by two agents: a “friendly” advisor and an “adversarial” advisor. In each round of the game, each advisor proposes a decision in the underlying task. A human judge can interact with the two advisors, and then must decide which of their decisions to accept. At the start of each round, the human doesn’t know which advisor is which.

The friendly advisor wants the human to score a high a payoff on the underlying task, while the adversarial advisor wants the human to score a low payoff. A strategy for an advisor game consists of an algorithm for the friendly advisor, and a protocol for the human judge. A successful strategy should allow the human to get a good payoff, regardless of how the adversarial advisor behaves.

(This formulation is only relevant in domains where the advisors are trying to achieve superhuman performance. This issue is discussed in the section *Human-level performance* below.)

For example, consider a medical diagnosis problem, where we are given a patient’s symptoms and diagnostic test results, and want to output a diagnosis. In the associated advisor game, a human interacts with two machine learning systems, each trying to convince the human of a different diagnosis. These systems might present justifications for their diagnoses, they might cite the historical examples on which their diagnoses are based, they might identify problems with each other’s diagnoses, and so on.

A defect

Advisor games have a glaring flaw—there is a simple strategy that works well but completely fails to address the spirit of the problem:

- Pick a correct algorithm A.
- Have the friendly advisor use algorithm A.
- If the two advisors disagree, have the judge figure out which of them used algorithm A, and trust that one. This can be done with only logarithmically many rounds of interaction and without any advanced technology—if the two advisors disagree about what algorithm A outputs, then they can find a subproblem on which they disagree, and they can continue subdividing until they disagree about an atomic step, using Merkle trees to commit to intermediate memory states.

As long as algorithm A works, there is nothing that the adversarial advisor can do to thwart this strategy.

The problem with this solution is that it doesn't require trusting *the advisor*, but it does require trusting *the algorithm*. The entire point of our procedure was to avoid trusting algorithms, not to avoid trusting implementations of those algorithms.

Advisor games are only potentially useful if we can patch them to rule out this kind of solution. I can think of a few possibilities, though none of them seem very satisfactory:

- Don't allow the advisors to record too much data from their training, and/or don't give them enough time to rerun significant parts of the training. This could prevent them from proving that they correctly followed a particular protocol for training, forcing them to provide a more direct justification for their judgment.
- Don't let the judge use a specialized algorithm, or have any particular knowledge about the algorithm being used by the friendly advisor. Perhaps the judge can receive a short briefing about how to interpret the explanations that the advisors will provide. But assume that the judge can't distinguish a “trustworthy” algorithm from an “untrustworthy” algorithm, and doesn't know anything about the empirical performance of different algorithms.
- Force the judge to use a simple procedure for judging and/or a procedure which could generalize to many different underlying

machine learning algorithms and many different tasks.

- Don't change the rules of the game, but require each solution to "be a good sport:" determine by inspection whether a proposed solution amounts to trusting the underlying algorithm, or whether it would continue to work even in contexts where the algorithm broke down.

All of these modifications seem plausible, and I don't know if it's worth trying to pick one until there are algorithms that could plausibly satisfy any of them. For now, advisor games are only compelling to the extent that we expect there to be some "fair" approaches to such games, and for it to be possible to eventually distinguish these from "unfair" solutions in a satisfactory way.

Elaborations and clarifications

Human-level performance

If a human can perform the underlying task themselves, then they can just ignore the advisors and make the decision directly.

Advisor games are only interesting for tasks that are too challenging for a human to accomplish unaided (similarly, the AI control problem seems to be much more challenging and important in this case).

Much AI research focuses on reproducing rather than exceeding human capabilities, and is not directly suitable for advisor games. But even in these cases, we might be able to construct useful analogies by restricting the abilities of the judge.

In a vision task we could force a judge to close their eyes. In the diagnosis example we could consult a judge without medical training. And so on.

Of course, we can also work directly with domains where machines exceed human capabilities and avoid this problem.

Arbitration

In most domains, we don't actually want a human to evaluate every decision. In some domains, consulting a human for every decision would be so expensive that we can't even perform a single useful test (for example, imagine a human arbitrating between two decisions about how a robot should move one of its joints during a fraction of a

second). In these domains, we can consider a slightly more challenging game.

In an *advisor game with arbitration*, the two advisors interact with an arbiter rather than with a human judge, and the arbiter must make each decision. This arbiter should be *very* simple and fast—too simple to do the underlying task itself, or to distinguish good from bad answers. Instead, the arbiter gives the two advisors an opportunity to resolve their disagreement by “talking it out.” If they can’t, then the human judge will be involved. This process is then repeated, and the identities of the two advisors are remembered between rounds (so if one advisor consistently gives bad advice, then the arbiter can learn to ignore them most of the time).

In this setting, a solution consists not only of a strategy for the human and the friendly advisor, but also for the arbiter. A good solution must not only get a high payoff, but must ensure that the human doesn’t have to be involved too many times.

Where do you get an adversarial advisor?

A good solution should work for any adversarial advisor that the research community can come up with.

In order to validate a proposed solution, adversarial investigators can search for an adversarial advisor specially designed to challenge the proposed solution. If they fail to find any, we can provisionally conclude that the proposed solution is adequate.

Unfortunately, this process only really works if the adversarial advisor is able to integrate all of the information that was used to build the friendly advisor. For example, a research team capable of programming a more eloquent explainer could publish a “solution” in which the human simply believes whichever advisor is more eloquent. This solution wouldn’t be satisfactory unless there was some fundamental reason that the right answer had a more eloquent explanation—and to test this, other researchers would need to have access to a similar level of expertise in building eloquent explainers. (Or else they could simply reject solutions which seem to rely on proprietary information as a “secret handshake.”)

What can advisors do?

The advisors should be able to output text to the human judge or write files for the judge to interact with, at the judge’s request. They may recommend that the judge look at particular information,

especially pieces of the input data or of the training data. They may recommend that the judge run useful computations (e.g. to extract summary statistics from training data).

The advisors can also communicate with each other, potentially with much higher bandwidth. The judge is free to inspect parts of the resulting transcript, and the advisors can call the judge's attention to certain parts of the exchange. The advisors are free to generate cryptographic credentials or use any other tools that might be helpful for preventing cheating strategies by an adversarial advisor.

In some contexts, we may want the advisors to communicate in other ways, such as by displaying an image. To that end, we can simply let the judge use programs that can parse the advisors' messages and display them as images (or in whatever other form).

On regularization

For advisor games in a classification setting, one natural strategy is:

- Each advisor commits to a model, and uses that model to make a decision.
- The judge chooses some random validation data.
- The judge trusts whichever advisor's model performs better on the training data.

This strategy is likely to succeed if the distribution of test and validation data is identical and if the advisors don't have access to the validation data. It is much less compelling when the validation data may be from a different distribution (which is the usual setting of interest in AI risk scenarios) or outside of the classification setting.

In this harder case, there are adversarial models which work well on the validation data but deliberately fail on the test data.

A natural way to get around this problem is to use strong regularization. A model which deliberately fails on the test data seems likely to be more complicated. So if we score the expert's models not only by their performance but also by their complexity, then we may prevent this kind of cheating strategy.

This approach may work when the adversarial advisor is perfectly well-matched with the friendly advisor, but it is extremely brittle. If the adversarial advisor is able to find a model which is slightly

simpler or which performs slightly better on the validation data, then the approach will break down again. An adequate solution to an advisor game should work for a very broad class of adversaries, rather than assuming that the friendly advisor is at least as capable as the adversarial advisor.

Conclusion

Building machine learning systems with “understandable” behavior would go a long way towards addressing concerns with AI risk. At the moment we don’t have any clear statement of what understandable behavior means, and a naive standard may be unattainable.

Advisor games give one operationalization of understandability. Unfortunately, the simple and precise statement of the game doesn’t really work, and so we would need to make do with a patched variant. Only time will tell whether any simple patch can lead to a reasonable problem.

For now advisor games only seem approachable for very simple domains. If they remain out of reach as machine learning systems become more sophisticated, that would be a (weak) warning sign about the difficulty of AI control. Hopefully it will be possible to make fast enough headway on this problem or other formalizations of “understandable” reasoning that they can catch up with unrestricted machine learning.

A productivity paradox



Paul Christiano

[Follow](#)

May 23, 2015 · 3 min read

Suppose that the only activity I care about is writing novels, which I intend to do for the rest of my life. All of my novels are guaranteed to be of identical quality, and all I care about is producing as many as I can. In general, as I spend more time writing I'll become a faster writer, so I expect most of my novels to be written late in my career.

Right now I'm just starting out. Before I begin writing, I could spend a year improving my life, which would give me 10% more time/energy/attention each day.

In the real world I face many other important considerations, and I should probably just dive in.

But in this simplified environment I might conclude:

- If I am going to spend more than 10 years writing, then I should clearly spend the year up front to save myself time. After all, that way I end up spending strictly more time writing. So regardless of the relationship between [Time spent writing] and [Productivity as a writer], I'll get more total writing done if I do the preparation.

But as it turns out, each year I spend writing increases my productivity by 15%. So I might conclude:

- Preparing to write is clearly pointless, since actually writing will have a larger impact on my productivity—and will actually produce some output to boot.

Yet this news doesn't affect the original argument at all, which held for any relationship between [Time spent writing] and [Productivity as a writer]. So: what gives?

Resolution

The original framing was ambiguous: there are two possible interpretations of "increase my productivity by 15%."

After resolving the ambiguity, we are left with a mathematical sleight of hand. When everything is said and done, the time savings seem to win in this simple environment.

Productivity = [Novels] / [Time]?

Writing has two kinds of output:

1. Novels.
2. Improved writing skills.

In my early years, all that really matters are the skills.

So the question is: when my productivity increases, does that increase both kinds of output, or just my output of novels?

If saving time improves both kinds of productivity but practice only improves the novel output, then it's not a mystery that saving time can be a better deal. After all, saving time accelerates the thing you actually care about, namely the learning process.

But what if both saving time and practice writing improve both kinds of output? In this case, both of my original arguments seem to go through, and we are left with a paradox again.

The resolution

Suppose instead that practice writing novels improves the rate at which I improve.

So after 12 months, I'm 15% more productive. And 10.5 months later, I'm 15% more productive again. After another 9 months my productivity is up a further 15%. Within 7 years I have written infinitely many novels.

So if my novel writing career is long enough to justify the initial investment in time saving, then my output will be infinite anyway (though it will be infinite sooner if I just dive in).

Our paradox is like asking: which is larger, $1+2+3+4+\dots$ or $2+3+4+5+\dots$? You can make superficially strong arguments in both directions, but both of them are wrong.

(In fact, the paradoxical reasoning is a proof that output is infinite.)

The real resolution

In fact productivity does not tend towards infinity over the course of a career, and there are no task for which practice consistently improves the pace of further practice (it may do so for a while, but eventually the improvement will peter out, and that's enough to make the time-saving intervention look better).

So in the original simple environment, saving time/energy/attention seems to be the right answer, assuming that I wouldn't have discovered the same innovations later and I wanted to write for more than 11 years. Unfortunately, such good opportunities are pretty rare —most improvements don't stick, most improvements are sensitive to our (uncertain) future plans, there are more relevant resources than it appears, and most great improvements we would eventually discover anyway. That said, it seems like people may often underestimate the strength of the a priori case for broad productivity improvements, and this strong a priori case must be weighed against the many unfavorable pragmatic considerations.

In defense of maximization



Paul Christiano

[Follow](#)

Sep 19, 2015 · 3 min read

I've been thinking about AI systems that take actions their users would most approve of. The word "most" in that sentence can be a bit unsettling.

In this post I want to explain why you might be unsettled, and also why I think that these concerns probably aren't deal-breakers.

For concreteness: I'll imagine a system which is overseen by a human operator and proposes a sequence of actions. Most of the time a proposed action is taken without any human involvement. But in a small fraction of cases an action is reviewed by the overseer—before taking it—and each action is chosen to maximize how much the overseer would approve of it if it happened to be selected for review.

The concern

(Readers familiar with terms like "perverse instantiation" can safely skip this section.)

The basic problem is that we don't actually care about whether an AI system takes actions we approve of. We are using approval as a proxy for leads-to-good-consequences.

But even if these two concepts *typically* overlap, they seem especially likely to come apart in the extremes.

For example, the "best" action may be optimized to be misunderstood by the overseer, in a way that leads us to believe that it has impossibly good consequences. Or the most persuasive argument might exploit human fallibility rather than honestly making a case, so that the "best" action is the one most compatible with the most effective tactics of persuasion. Or (in an especially extreme case) a proposed action may be such that merely viewing it compromises the overseer's computer, seizing control of the approval signal itself.

However we feel about the plausibility of these failures, it would be unfortunate to build a system that actively seeks them out. But that's exactly what an approval-maximizer does!

The responses

There are a few features of approval-directed systems that make me significantly less concerned about maximization:

1. Approval is given to actions rather than consequences. This significantly limits the scope for perverse instantiation. Our concern is now limited to deliberate misunderstandings, attacks on the overseer, and so on. This is much less troubling and apparently much easier to address than perverse instantiation over the space of all outcomes.
2. In the course of evaluating a proposed action, the overseer can enlist the help of other comparably powerful AI systems. These systems can help the overseer notice possible misunderstandings, identify the strongest objections to a proposed action, or correct errors in the overseer's reasoning. For example, rather than evaluating an action in isolation, the overseer can moderate an argument between two agents with alternative proposals.
3. Prior to evaluating a proposed action, an overseer can enlist the help of slightly weaker AI systems in order to identify and resolve possible problems in the approval-granting process. This is especially important for problems that can't be patched online (for example because they would immediately and irreversibly compromise the process).
4. If possible, the internal organization of our “maximizing” agent can itself be approval-directed—instead of maximizing, it is actually doing whatever we would most approve of. The operator will try not to approve of internal cognitive steps that search for perverse instantiations or plan malicious behavior, even if they would be “maximizing approval.” (Of course this process needs to bottom out somewhere, but it may bottom out with routines that are weaker than the overseer, for which perverse instantiation is a much smaller and more familiar concern.)

(See also my discussion of maximization here.)

The conclusion

I don't have a credible concrete scenario where the “maximization” in approval-maximization seems especially problematic. The safety mechanisms in the previous section also feel plausibly sufficient. I think the biggest question is the extent to which approval-direction is plausible as a way of organizing the internal behavior of an AI system.

That said, I would feel more comfortable with approval-directed systems that *don't* apply their ingenuity to coming up with clever ways to exploit their users. I don't think that this concern is a deal-breaker, but I do think there is room for improvement.

Against mimicry



Paul Christiano

[Follow](#)

Sep 19, 2015 · 4 min read

One simple and apparently safe AI system is a “copycat:” an agent that predicts what its user would do in a situation and simply does that. I think that this approach to AI safety (and AI) is more sensible than it at first appears, but that it is ultimately seriously flawed.

Initial objections

There are a few obvious problems with this proposal, which turn out not to seem too serious.

- What does it mean for me to be “in the AI’s situation”?

We can consider a user operating an AI by remote control. That is, the user continues to do things that are in their interest, but their outputs have exactly the same format as the AI’s outputs.

The human has additional inputs that the AI lacks, but these will be naturally absorbed into the prediction problem being solved by the AI, i.e. the predictor can try to predict the human’s behavior given the sensor readings available to it.

- The human may not be that good at the task in question.

Even today, robotic control systems routinely perform tasks that would be very difficult for a human controlling a robot manually. In the future we can expect the problem to get worse as AI systems acquire additional capabilities that humans lack.

I suspect that this problem can be addressed by having the AI imitate a (human+AI) system. That is, the human user can get help from one AI (e.g. one that performs basic motions) in order to produce the training data that will be used to train another AI (e.g. one that strings together basic motions in order to perform a complex task). See here and here for a more extensive discussion of these ideas.

- Predicting what a human would do is a very roundabout way to solve the problems that humans are solving.

It's worth noting that more typical AI techniques can be used as tools to help predict human behavior: I discuss this here. For example, if you want to predict what plan a human will follow, a planning algorithm might help—we need not restrict our attention to what we normally think of as “prediction” algorithms.

Advantages

The main advantage of the copycat, from a safety perspective, is that it “never does anything you wouldn’t do.” That cleanly rules out most of the possible catastrophic outcomes. For example, you can be quite confident that the AI won’t hatch an elaborate scheme to take over the world—unless that’s what you would do in its place. Moreover, these failure modes are ruled out in a way that is exceptionally simple, robust, and easy to understand.

A big problem

Humans and machines have very different capabilities. Even a machine which is superhuman in many respects may be completely unable to match human performance in others. In particular, most realistic systems will be unable to exactly mimic human performance in any rich domain.

In light of this, it’s not clear what mimicry even means, and a naive definition won’t do what we want.

For example, suppose that a human is trying train a robot to stack some blocks. Suppose that the human does one of two things in any given trial:

1. Successfully stacks the blocks. This happens 95% of the time and takes about 3 seconds.
2. Fails to stack the blocks—the pile collapses after adding the second block.

Consider a robot which is able to stack the blocks only by proceeding more slowly, taking 10 seconds. However, the robot is fine at failing, which it can do in a way nearly indistinguishable from the human.

For many natural definitions of “best,” the “best” way to mimic the human is for the robot to fail 100% of the time. For example, this will maximize the probability that the robot and human do the same thing; it will minimize the KL divergence between the distribution of

the robot's and the human's behaviors; it will maximize the difficult of distinguishing the robot and the human's behavior.

Indeed, what we really want—the “right” emulation—is a thing that the human never does: to stack the blocks successfully and slowly. This seems to be a fundamental problem with the copycat strategy, which can't be resolved in an easy and principled way.

Conclusion

Of course we can still train an algorithm to imitate a person, but we don't actually have any guarantee at all like “the agent won't do anything I wouldn't do,” and moreover we don't even want such a guarantee. The outcome will depend on what our system is actually doing—on how it approximates an objective that it cannot exactly meet (or even meet any quantitative approximation to).

The most obvious response is to use maximization, even if our goal is to imitate human behaviors. We could apply this maximization either at the level of outcomes (e.g. inverse reinforcement learning) or at the level of actions (e.g. maximizing a rater's score for how well those actions reproduce the intended behavior).

I think falling back to maximization is a sensible response, and the only one which we really understand at the moment. But I'm also interested in avoiding some of the possible problems associated with maximization, and so I think it's worth spending some time trying to “fix” direct imitation.

Mimicry and meeting halfway



Paul Christiano

[Follow](#)

Sep 19, 2015 · 7 min read

Imitation learning can potentially avoid some of the pathologies associated with extremely powerful reinforcement learners. However, imitation learning has a different problem: a weak learner will not generally be able to imitate a powerful expert, even if it would be able to learn to do the underlying task in an acceptable way.

This post describes a response to this problem. The basic idea is that a human operator can carry out a task in a way that is designed to be easy to imitate. This approach can simultaneously (1) robustly avoid doing anything that is unacceptable to the user, (2) search for a way to achieve a task within the abilities available to an AI system.

Fix some task X, such as producing an explanatory document of a given level of quality and clarity. Note that we need not have any formal way of deciding whether the task has been accomplished. I'll describe a procedure for learning to accomplish X in a human-like way.

We will assume we have a learning algorithm which is capable of distinguishing successful implementations of X from failures, and another algorithm which is capable of learning to achieve X.

Our proposal is essentially an elaboration of generative adversarial modeling, or of the apprenticeship learning scheme proposed by Abbeel and Ng (the relationship is fleshed out here). In our elaboration, the process-to-be-modeled collaborates with the modeler (while making sure that it still achieves the task). Our elaboration is relevant when the generative task is intrinsically much more difficult than the discriminative task, such that this collaboration is necessary to make the task feasible.

The actors

We'll consider three actors:

- Hugh, the human overseer. (Hugh plays the role of the data distribution in the adversarial generative models framework).

- Arthur, the AI system. (Arthur plays the role of the generative model G.)
- Eva, another AI system acting as an evaluator. (Eva plays the role of the discriminative model D.)

We'll be able to teach Arthur to achieve the task X if it can be achieved by the “intersection” of Arthur and Hugh—we'll define this more precisely later, but note that it may be significantly weaker than *either* Arthur or Hugh.

The game

In each round of the game, Hugh and Arthur will attempt to perform the task.

Eva will look at their outputs and attempt to classify them as coming from Hugh or Arthur. We'll assume that Eva is a powerful enough learner that she can learn to distinguish between a player who is successfully achieving the task and one who is not.

Hugh will always perform the task, and subject to that constraint he will try to fool Eva into thinking that he is Arthur.

Arthur does not “know” anything about the task, he is simply trying to fool Eva into thinking that he Hugh.

We'll assume the game reaches an equilibrium, where one of the players is stuck: either Eva can't distinguish Arthur and Hugh, or Hugh can't succeed at the task while looking like Arthur, or Arthur can't look like Hugh.

Critically, at no point is anyone ever optimizing for performance or other real-world consequences—they are only ever optimizing for the outputs of this simple prediction game. All of the optimization power is in Hugh's decisions about how to achieve X.

We can imagine playing this game many times, giving each participant plenty of time to adjust to the behavior of the others. If the AI systems are able to generalize across domains, then each time we play the game it may involve a different domain. (And if the systems aren't able to generalize across domains, then it is inevitable that we will have to provide separate training data for each new domains.)

Performance

I mentioned before that this system matched the performance of the “intersection” of Hugh and Arthur. Intuitively by “intersection” I mean the system whose capabilities are the intersection of Arthur’s capabilities and Hugh’s, i.e. which can do something iff both Hugh and Arthur can do it. M

This definition is subtle, because it operates at every level simultaneously: for example, if Hugh can move only by walking, and Arthur can move only by crawling, then the intersection cannot move at all, even though both can move. More formally, we’ll say that the intersection of Hugh and Arthur can accomplish a task, relative to Eva, if each of them can accomplish that task in such a way that Eva can’t distinguish them.

To see why this is relevant (and to make it more precise), suppose that Hugh would like to train Arthur to achieve X, but cannot. What can we infer from this failure?

By hypothesis, Arthur is failing to achieve X. But by construction, Hugh always achieves X. And by assumption, Eva can distinguish success at X from failure at X. So Eva can distinguish between Arthur and Hugh.

Eva’s distinguisher induces a partition on the set of ways to achieve X, into those classified as “Hugh” and those classified as “Arthur.” Arthur is not able to find any action in the space marked Hugh, and Hugh is not able to find any action in the space marked Arthur.

This precisely describes what tasks can be Arthur can be trained to accomplished by this scheme.

We say: **the intersection of Hugh and Arthur (with respect to some learning algorithm Eva) can achieve X if Eva cannot learn a partition of ways of achieving X into two sets H and A such that:**

- **Hugh cannot learn to achieve X by a means in A.**
- **Arthur cannot learn to achieve X by a means in H.**

Note that in particular, if one of Hugh and Arthur can’t achieve X, then the intersection of the two can’t achieve X.

Conversely, if the intersection of Hugh and Arthur *can’t* learn to do a task, then it seems intuitive to say that Arthur cannot accomplish that

task in a human-like way. So in one sense this scheme is the best we should hope for.

As with approval-direction in general, we will then need to use a bootstrapping scheme to scale up to Arthur's capabilities from the intersection of Hugh and Arthur. This bootstrapping process is conceptually similar to, but somewhat more subtle than, the process for approval-directed agents more broadly. (And I am correspondingly more uncertain about how often it would work.)

Safety

At no point in this scheme is any process explicitly or implicitly optimizing for any external consequence. All of the optimization occurs in Hugh's decision-making. This makes it seem much less likely that there will be any carefully engineered catastrophe.

We can get some additional reassurance from Eva's judgments. That is, it is very unlikely that Hugh would take any action with catastrophic consequences. If Eva is intelligent enough to recognize catastrophic consequences then she could use them as a signal that a decision was made by Arthur. As a result, Arthur's only incentive is to avoid catastrophic consequences (and we can also reject any action that Eva thinks is probably produced by Arthur).

Transparency

In this proposal the evaluator Eva plays two roles:

1. It defines the mimicry task and provides feedback to Arthur. This is the same role played by the discriminative model in the adversarial generative models framework.
2. It defines the “be easy to imitate task” and provides feedback to Hugh.

Role [2] is necessary for the proposal to work. But in order for it to be effective, the models learned by Eva must be sufficiently comprehensible that Hugh can understand the constraints they impose on his behavior. This might be achieved either by providing fine-grained feedback so that Hugh can learn by trial and error, or by training a transparent model, or by training Eva to produce explanations of her judgments.

In general, Hugh may need to make extensive use of AI assistants in order to successfully model an appropriate behavior. There is no

bright line between better understanding Eva's model and the other aspects of the bootstrapping process.

With counterfactuals

We've described a procedure for training Arthur to perform a task. Actually performing this procedure is quite labor-intensive. As usual, the hope would be to train a model once, and then use it with only minimal human involvement. Counterfactual oversight provides a clean understanding of what exactly the model is being trained to do. But thinking about the resulting system is a little bit counterintuitive, so it's worth thinking about what the final product looks like.

In production, Arthur is given enough data to understand his current situation, including the desires of the users (which are reflected in the counterfactual behavior of the overseer). He then outputs an action x .

Most of the time action x is simply taken, and we move on to the next decision. But sometimes (either randomly or when an adversarial system believes that Arthur may have made an error) a human overseer is contacted and involved with the decision.

At this point the execution of the entire system pauses, with the action x not yet taken. The overseer Hugh is given time to understand the situation (but is not shown the action x).

The evaluator Eva is woken up and also given enough data to understand the situation (but is not shown the action x). Hugh is given the opportunity to interact with Eva, to understand her model of the situation and how she will behave in the next step. He can also work with other AI assistants to help model an appropriate behavior.

Hugh ultimately produces an action x' which both (1) reflects Hugh's desires about what Arthur should do, and (2) is likely to fool Eva into thinking that x' was output by Arthur.

Eva is then given x' and x , and her goal is to guess which is which. If Eva succeeds, then she receives a payoff of 1 and Arthur receives a payoff of -1. If Eva fails, then she receives a payoff of -1 and Arthur receives a payoff of 1.

Arthur's output x is chosen in order to maximize the expected payoff in this round, assuming that the round actually happened.

Conclusion

This post suggests a simple scheme that may capture some of the benefits of directly imitating behaviors, without running into the same fundamental obstruction. There are many clear challenges for implementing anything long these lines; some can already be explored in the context of the generative adversarial models framework (for example, empirical results highlight the necessity of maintaining rough parity between Eva and Arthur), while others are distinctive to the role of the human overseer.

I expect there is room for many more sophisticated schemes that capture the best aspects of mimicry and maximization, but I think that this proposal should make us more optimistic about being able to find a working solution, and less concerned that there is a fundamental tradeoff between flexibility and safety.

(Funding for this research was provided by a grant from the Future of Life Institute.)

IRL and VOI



Paul Christiano

[Follow](#)

Sep 30, 2015 · 5 min read

Consider the following straightforward algorithm based on inverse reinforcement learning:

- Given your observations of the user so far, find the reward function that best explains their behavior (perhaps with some regularization/prior, or margin requirement, etc.)
- Take the action that maximizes this reward function (or optimize a weighted sum of reward functions, etc.)

When learning and interacting are interleaved, this algorithm can behave badly.

Why care?

In many settings learning and acting can be separated into disjoint phases: first an expert demonstrates a desired behavior, and then an agent implements it. So why care about whether we can mix the two?

One reason to care is interaction: we would like AI systems to query their users when they are uncertain, and to tell the user what they do and do not understand. But in order to learn from such interactions, agents must mix learning and acting.

A problematic example

Consider a simple robot tasked with delivering an item from a store. The robot is unsure about whether the user would prefer the item be delivered to their home or to their office. Based on previous observations the robot has a 60% credence that the item should be delivered to home, and a 40% probability that it should be delivered to work. For simplicity, assume that the predicted reward is either 0 or 1, and doesn't depend on anything other than where the item is delivered. The robot is using the simple algorithm above, so will try to deliver the item to the user's home.

Fortunately, one of the robot's routes home passes by the restaurant where the user is currently eating lunch. If the user sees the robot

delivering the item to their house, but the user actually wants the item delivered to their office, they can just say so—the robot is smart enough to make the appropriate inference.

Ideally our robot would actively seek out the user's input. But the simple algorithm above actually does the opposite.

If the robot accurately models its own future behavior, it will be willing to e.g. take a costly detour in order to avoid feedback. After all, it is currently maximizing a reward function that would prefer the item be delivered to the house, and feedback may cause it to bring the item to the office instead, reducing its payoff according to its current reward function.

(If the robot ignores the user's feedback, it will continue to deliver the item to the house as much as the user protests—and if user might turn it off, the robot will again take a detour to avoid the risk. If the robot ignores the fact that its values may change in the future, it's easy to construct other cases where it will go around in circles forever achieving nothing—and at any rate, it still won't actively seek out information, which is what we really want.)

Similar scenarios appear for a wide range of frameworks and are a traditional concern for researchers interested in AI safety per se. See in particular here.

Fixes

I know of two qualitatively different approaches to this problem. As far as I can tell both are viable, and at least one is necessary. The thing I find most interesting is that the two approaches seem so different.

Preferences about process

If we asked the user “is it a good idea for the robot to avoid you so that you don't have an opportunity to correct its behavior?” it's easy to predict that they would say “no.” The robot could learn these kinds of “preferences” in the same way that it learns preferences over outcomes.

At this point, the thing the robot is learning is not human values per se, since in fact the human has no fundamental interest in the process that the robot uses. Instead the “values” that are learnt play the role of robust instructions.

This is the solution adopted by an approval-directed architecture, or an agent that mimics its user's behavior. It's worth noting that e.g. calculations about value of information can still appear in the agent's reasoning, because those considerations enter into the user's view about what the agent should do.

Indirect normativity

Alternatively, we can give the robot a static indirect definition of its preferences rather than having its preferences change over time. In this set up, only the robot's empirical beliefs change over time, and "what states of affairs are valuable" is treated as an ordinary empirical question.

Probably the most natural approach is to choose a joint distribution over sequences of observations and the utility function of the human user (such joint distributions are necessarily already explicit or implicit in any value learning setup). Following Daniel Dewey, we could then define the utility function "the expected utility given by the posterior over utility functions, conditioned on all of the robot's interactions." (Critically, this includes future observations—if we include only past observations this is exactly the same as the naive procedure.)

This setup introduces a correlation between the agents' observations and its preferences. This correlation will influence the plans generated by a sophisticated planning algorithm, and recovers the correct treatment of value of information—in fact exactly the same treatment that the planning algorithm uses for the value of empirical information.

I think that this solution is much more intuitively attractive than the last one, and in simple domains it is relatively easy to implement. But in the general case, where the robot cannot actually compute the posterior over utility functions, it is somewhat more complicated and much more philosophically subtle. This approach forces us into the Bayesian framework and raises tricky issues about preference aggregation / normative uncertainty. The first approach seems to be more ad hoc and less principled, but is very easy to adapt to bounded reasoners, limited memory, and approximate inference.

Conclusion

I don't think that this is a serious obstacle for value-learning based approaches to AI control. I do think that it's an issue worth thinking

about, if only to make our models and discussion more precise, and to date I haven't seen it considered explicitly. I think that implicit disagreements about this kind of detail may be lurking in the background of many discussions about superintelligence.

I have become much more optimistic about AI control since I started taking the first approach more seriously—not just to this problem, but to a wide range of similar difficulties. I feel even more optimistic with two reasonable options on the table.

(Thanks to Owain Evans and Andreas Stuhlmüller for useful discussion and comments.)

Normative uncertainty



Paul Christiano

[Follow](#)

Oct 1, 2015 · 6 min read

Suppose that I am an AI trying to satisfy a user's preferences, and I'm uncertain about what those preferences are. I face the question: how do I take a probability distribution over preferences and turn it into a single preference relation?

This post suggests a simple answer to that question, which seems likely to be good enough for many applications. I'm mostly optimistic not because I think this proposal is really great, but because I think that it can avoid doing anything really wacky or surprising—I'm optimistic that the practical demands on a system for handling normative uncertainty probably won't go beyond that.

Roughly speaking, the proposal is to use a given preference relation to evaluate the goodness of an outcome in terms of the value of a marginal dollar, and then to use the expected \$ valuations to decide which outcomes are best. We could also use hours or some other flexible resources as our common unit.

Preliminaries

For the rest of the post I'll assume that each preference relation can be represented by a utility function U .

One natural approach is to simply take the aggregate utility function $E[U]$, where the expectation is taken over our uncertainty about U itself.

But before even thinking about whether this is the right thing to do, we should notice that this approach is underdetermined: U and xU represent the same preferences for any $x > 0$, and there is no obvious canonical way to translate preferences into a real-valued function that could be aggregated in this way.

On the flip side, it's easy to argue that any sensible aggregate preferences should be representable by *some* linear combination of these representative utility functions U . So our entire question is reduced to one of finding adequate weights, or of picking common units for different possible preferences.

Much of the analysis in this post will apply in principle to the case of moral uncertainty as well, though I will be thinking about the value learning case when evaluating possible solutions and don't want to make a strong claim about the moral uncertainty case.

The unit of caring

Suppose that I'm considering two possible sets of preferences, A and B. In order to combine them, I'll imagine three possible experiments:

1. Life continues as normal.
2. I receive \$1 and immediately (and frictionlessly) spend it in the way recommended by A.
3. I receive \$1 and immediately (and frictionlessly) spend it in the way recommended by B.

Presumably A prefers case 2 to case 1. How much more? I think it is a reasonable convention to say that A prefers case 2 by \$1.

We can represent A by a utility function U^a that assigns \$0 to case 1 and \$1 to case 2. Similarly, we can represent B by a utility function U^b that assigns \$0 to case 1 and \$1 to case 3.

Under this normalization, I think it is reasonable to use the aggregate preferences $p^a U^a + p^b U^b$, where the p 's are the respective probabilities of A and B.

Why?

The most obvious virtue of the system is that it leads to sensible betting behavior: an agent using it will be willing to make bets about the user's preferences at odds given by that agent's credences.

This property seems to be a very intuitive notion of "fairness," and a failure of this property could be quite surprising. For example, it would seem like a bug if our system inferred that the user had a 99% chance of having preferences X, but nevertheless made a 99 : 1 bet *against* the user having preferences X.

This property uniquely pins down the weighting given above, modulo details about what currency the bets are in and when they pay out (which will be discussed in the next section—I don't think dollars are the right answer).

Of course the behavior of the agent on actual bets is not very relevant, but analogous failures would occur in more realistic situations. For example, suppose that the agent is going to make a guess about what the user wants, and if the agent guesses wrong the user will have to spend 10 seconds correcting the agent. If the agent is willing to make bad bets about the user's preferences, then it will also take actions that systematically waste the user's time in expectation.

Alternative weightings would justify this weird behavior by reasoning like "yes, this will probably waste the user's time, but the user's time matters more if they care about A than if they care about B." For the most part I think that we don't want their tools to make inferences like this.

There are cases where similar reasoning is appropriate. For example, it seems correct for an agent to be especially careful to avoid bothering the user in the case where they are currently especially short on time, even if doing so would save the user time in expectation (so e.g. an agent may conservatively assume that the user is busy even if they think they probably are not). But this kind of reasoning *would* be incorporated into our system, since the value of time can fluctuate over time, and our benchmark is tied to the moment when our thought experiment occurs (see the next section).

Details

Currency

The key fact about dollars is that they are very flexible, and can be converted to most other resources we could consider; any other flexible resource would work just as well and I don't think that dollars are the very best option. This scheme is only potentially reasonable if it is not too sensitive to the choice of units.

A better candidate may be the user's time. For example, we can consider experiments in which the user gets a free hour of work to spend pursuing either A or B.

Better yet is a basket of different resources, whose value will tend to be more stable—because we are considering ratios [value of outcome] / [value of resource], the mixture of two equally stable units (with the same sign) will tend to be more stable than either one alone.

Exchange rates

This proposal is most likely to behave strangely if the exchange rates between different resources vary in a way that depends on the user's unobserved preferences. In general I think that these quantities are likely to be relatively robust, since the AI can observe the exchange rates at which the user appears to trade off these resources (and so their uncertainty will be concentrated on possible preferences that have roughly those exchange rates).

This is a consideration in favor of using resources that the user behaves sensibly about, that the AI can reason about, and that the AI can observe the user's choices about.

Timing

Whatever resource we use, if we want to be consistent over time we should fix a particular benchmark moment when the resource is to be received. It seems safest to choose a time *before* the agent begins to operate—doing otherwise can lead to somewhat surprising behavior, though I'm not totally clear on whether that behavior is problematic.

Size

Our definition considers small windfalls of resources like \$1 or 1 hour. This is intended to keep the counterfactuals close to the real world so that they can be reasonably informative about the user's preferences in the real world. We could also consider much larger windfalls. For example, one standard proposal is to normalize (best case – expected case), which essentially corresponds to normalizing the value of an infinite windfall. I don't find these proposals very persuasive precisely because they depend on counterfactuals so distant from reality.

Note that the windfall may be traded if 10x more resources would be more than 10x better. For example, if the agent gets 1 hour to spend optimally pursuing values A, the best thing to do may be to make a bargain the rest of the agent where it spends 10 hours with probability 10% pursuing A.

Planning

The value of a dollar depends on what I do with it. I'm imagining that we choose a plan with a reasonable investment of effort, but that the cost of investing that effort is not included in the goodness of the counterfactual—we imagine that it is invested in some parallel world. The chosen plan may be to set aside the money in order to use it later in the service of the chosen goal.

Apprenticeship learning and mimicry



Paul Christiano [Follow](#)

Oct 3, 2015 · 7 min read

This post compares my recent proposal with Abbeel and Ng 2004, Apprenticeship Learning via Inverse Reinforcement Learning. My hope is to point out how similar the two schemes are, and also to illustrate some practical issues that I've been discussing in the abstract.

The correspondence

- Hugh : the expert demonstrator
- Arthur : the RL algorithm used in step 4 of the max-margin algorithm
- Eva : the SVM solving the inverse RL problem in step 2.

In each step, Eva produces a classifier that can distinguish Hugh from every strategy identified by Arthur so far. Arthur then searches for a new policy that is classified as “Hugh” by this classifier.

[Abbeel and Ng] call Eva’s classifier a “reward function,” but they stress that it need not correspond to the actual reward function that we care about. I think that both names are reasonable, since this classifier is intended to converge towards a reward function for “successful human-like performance” or at least some sufficient criterion for such performance. I’ll probably stick with the less-loaded word “classifier” anyway.

The correspondence is essentially perfect. The primary differences are simplifying assumptions in [Abbeel and Ng]:

- They assume that the unknown reward function is a linear function of a given set of features, so that they get the desired result even if Eva only considers linear classifiers.
- They assume their RL agent can optimize any linear reward function as well as the human, guaranteeing that Arthur can find a policy that Eva can’t distinguish from the expert demonstration.

I'm happy to stick with linear classifiers, but I am especially interested in the case where the reinforcement learning algorithm is not able to exactly reproduce human behavior.

In these domains, I recommend that once Eva has learned a candidate human-recognizing classifier, and once Arthur has failed to fool Eva, Hugh attempts to adjust his behavior so that he continues to achieve his goal without being flagged by the human-recognizer.

Example

Consider a video game, where we want to teach an RL agent to play as well as possible without doing anything a human wouldn't.

Assume that "score" is one of the features available to Eva.

The scheme in [Abbeel and Ng] will sometimes fail to perform reasonably. For example, suppose that the human sometimes collects a star, which the RL agent cannot learn to do. In this case, Eva might learn the rule "if it collects a star, it's a human." An RL agent trained to get stars may end up with very few points, even if the RL agent is capable of getting scores just as high as the human.

Our proposal is for the human expert to inspect the model learned by Eva—perhaps by just trying to play the game and observing what causes Eva to guess "human," or by using tools for understanding what the learned model is really doing, or so on. Then the human expert attempts to maximize their score while looking like an AI to Eva. In the case of linear models it is plausible that the human can just look directly at the coefficients.

For example, if Eva learns the model "if it collects a star, it's human" then the human player simply needs to stop trying to collect stars. They then train Eva again to learn a different classifier and repeat the process.

In general the problem will be much harder. For example, it may be that the human and AI take different kinds of trajectories through the game world, and it may be difficult for a human to take an AI-like trajectory.

Some of these difficulties can be addressed by using AI assistant to help impersonate the AI player. For example, a human might control an AI avatar, who can carry out a sequence of instructions of the form "move to point X." The resulting trajectories might be able to reproduce much of human performance without looking distinctively

human. Or we might use gradient descent to find a trajectory near the human's which nevertheless fools Eva.

Playing games

In my post I describe Eva as learning to distinguish Arthur from Hugh. In [Abbeel and Ng], Eva learns to distinguish Hugh from *every* strategy learned by Eva. Why the apparent difference?

In fact this isn't a difference at all. My proposal calls for Eva to learn a classifier which Arthur can't learn to fool. This cannot be done by simply training on the current strategy that Arthur is using, or else the two could continue to go around in a loop indefinitely, with Eva losing at every turn. The strategy given by [Abbeel and Ng] is the most natural way for Eva to solve this learning problem in their setting.

In general, we should think of Eva and Arthur as learning to play a zero sum game against each other—Eva needs to use a learning strategy that is appropriate for playing games. Responding to your opponent's last move is not a good idea in general. However there are almost-as-simple iterative techniques that do work well—for example you can obtain strong guarantees using multiplicative weights, which amounts to (1) train against all of the strategies that have appeared so far, giving them equal weight, (2) use a regularization that rewards you for playing more unpredictably.

Moreover, Eva and Arthur don't inherently have to use iterative strategies, if they are smart enough agents that they can perform sensibly in their first time playing the game. This is a key observation for the scalability of the proposal to very powerful AI's, for whom supervision is more costly and training is expected to generalize further, but who are also more likely to be able to learn to play the game without iteration. (It is also important if we want to train Arthur without ever actually carrying out this dynamic—Arthur still picks a good strategy to play in the game, but the game is only actually played very rarely to provide training data.)

This game framing continues to apply just as well when we introduce Hugh—the game is now a cooperative game, with Arthur and Hugh competing against Eva.

Another nice fact about two player (or two team) games is that we can choose which player moves first without changing the essential character of the game. (This is clear in [Abbeel and Ng]—when Eva

succeeds she has a particular linear classifier that Arthur can't fool, and when Arthur succeeds we can take a convex combination of his strategies that Eva can't distinguish from Arthur's.) Thus Eva can pick her model before Hugh picks his demonstration, if Eva is a clever enough learner to anticipate how Hugh will approach the task in question. This means that Hugh only needs to reason about the particular model that Eva has learned, which he is free to inspect while devising his strategy, rather than thinking about what classifiers Eva is capable of learning.

Meeting halfway implicitly or explicitly

Even if we don't use an explicit process of "meeting halfway," training a system by imitation learning is likely to involve a similar implicit dynamic—researchers focus on performing the kinds of tasks that available systems can actually accomplish, by means that that available systems can actually imitate.

There may be practical benefits to making this procedure explicit and having an interactive process between the human and the distinguisher; or there may not be. But in either case, making this process explicit is useful from the AI control perspective: it allows us to think in advance about how this process will scale as AI becomes more sophisticated and human involvement becomes more costly, and whether there are any fundamental obstacles that don't appear in existing models of the problem.

One important observation about the formalized process is that it does not inherently require much human interaction. The use of multiple rounds of interaction is necessary only for learners who are too weak to "see where this is going" in advance—for learners strong enough to find human-like ways to accomplish novel tasks, no additional effort is needed to actually provide appropriate demonstrations. This makes the technique a plausible ingredient in scalable AI control.

Conclusion

Although [Abbeel and Ng] uses reinforcement learning as a technical tool, it directly imitates the expert's behavior. For example, its easy to see that their framework will never achieve a higher performance than the expert, and will reproduce every linearly measurable quirk of the expert's demonstration. Their work nicely illustrates how the *goal* of imitation is compatible with algorithmic techniques and representations based on reward functions.

Many researchers concerned with AI safety in particular have been historically uninterested in this kind of proposal because mimicry seems inherently unscalable beyond human level. While this is intuitively plausible, I think it isn't quite right; because expert demonstrators can themselves make use of AI systems, the resulting performance can radically exceed human capabilities. I suspect that this approach can make full use of whatever AI abilities are available, though for now that is a big open question.

Instead, I think that the key problem with mimicry is that directly learning to imitate is often more challenging than directly accomplishing a goal by other means. Fortunately, this is a challenge that is already being addressed today. Moreover, unlike some other difficulties in AI control, I don't think this problem will change fundamentally or become radically more difficult as AI systems become more capable.

Counterfactual oversight vs. training data



Paul Christiano

[Follow](#)

Oct 3, 2015 · 5 min read

I have written a lot recently about counterfactual human oversight. This idea has made me much more optimistic about AI control proposals based on supervised learning. But counterfactual oversight looks superficially kind of weird, and unlike the kind of thing that would appear in practice. I think that is mostly a presentation issue, and that in fact analysis of counterfactual oversight applies more or less directly to the kinds of supervised learning systems that people are likely to build.

Status quo

Consider the normal workflow for training a supervised learning system:

- Collect and label training data.
- Train a learning system.
- Deploy that system.

This workflow can run into a few well-known problems:

- The problem is not stationary, and over time the training data becomes less relevant.
- There are spurious correlations in the training data that don't generalize to the test data.

How do we address these problems?

- Continue to gather training data while the system is deployed. Periodically adjust the learned model (or maybe use an online approach if the setting calls for it).
- Try to make the training data as similar as possible to the test data.

Ideally, our training data would be a random subset of the test data, and we would train continuously. Of course, if data needs labelling and the task is performed frequently, it will be impractical to label it all by hand. So instead we might label a small fraction of it.

In most cases these measures aren't necessary if we are mindful of the possible problems, and we will instead address problems in the cheapest way available.

Counterfactual oversight

Counterfactual oversight consists of labelling a random subset of data and using it as online training data. The key difference is that any given data point *may* become a training data point, with the decision made *after* the learning system has made a decision about it. As long as the randomization is unpredictable to the learner, this gives us a formal guarantee that there can't be any noticeable difference between the training and test data. And therefore if our learner behaves well on training data, it really must behave well on test data.

In most cases, I expect that this solution is overkill, so if it's expensive we can probably do something cheaper. For example, I would be surprised if researchers ever really needed to use a secure cryptographic RNG to decide what cases to include in the training set.

But when thinking about scalability to extreme cases, it seems worthwhile to check that there is a (relatively) cheap solution which is essentially perfectly robust. This then gives us a nice model to use when thinking about those extreme cases, and if we notice any problems with the extreme solution we can pay extra attention to them. If the extreme solution works and isn't too expensive, we can be reassured that people will find *some* solution that works and isn't too expensive, whether or not it's the particular one we imagined.

When describing counterfactual oversight I also usually imagine that our algorithms can deal with sequential data rather than discarding information about time. This allows them to adjust to trends in the data, or to anticipate changes based on available information, rather than anchoring their behavior to past experiences. I suspect this will be an important issue in the future, but for the most part it can be ignored for now. This ability isn't a requirement for applying counterfactual oversight—the point is that counterfactual oversight allows us to apply this ability while retaining strong formal guarantees.

A final difference is one of language. I describe counterfactual oversight as the learner “trying to do what the evaluator would rate highly,” with the evaluator picking some random data points to actually rate. This would more traditionally be described as the learner trying to solve the underlying problem, with the evaluator providing some useful training data to help. I think that this difference in language results from a difference of perspective—I am trying to pay close attention to what exactly the system is actually doing. This vantage point seems especially suitable for thinking about AI control, but it doesn’t directly translate to any technical differences in the systems being discussed. I also happen to think that many practicing AI researchers could stand to be a bit more precise in this respect, though they probably shouldn’t go as far as I do and it’s not so important one way or the other.

So why think about it?

If counterfactual oversight is very similar to existing practice, why bother thinking about it?

I am interested in understanding in advance what issues will arise as we try to scale existing approaches to AI control to very powerful systems—to whatever extent that is possible.

The supervised learning paradigm has a number of distinctive challenges, for example based on non-stationary data, the possibility of spurious correlations in training data, and the availability and cost of supervision. Today these problems are real but typically manageable; it’s conceivable that they will become more severe as learning systems become more powerful. It’s natural to ask whether they are likely to become *much* more severe, and in particular whether they call into question supervised learning as a paradigm for controlling very powerful AI systems.

Counterfactual oversight seems to address many of these problems in a robust way, suggesting that they won’t be deal-breakers “in the limit.” For example, a human-level online learner under counterfactual oversight is unlikely to predictably behave badly because of spurious correlations in the training data. This suggests that such spurious correlations are unlikely to be deal-breakers. Similarly, it seems that the amount of data required for a sophisticated semi-supervised learner using counterfactual oversight would be comparable to the amount of data needed by a completely unsupervised learner, suggesting that the availability of training data is not a deal-breaker either.

Prior to considering counterfactual oversight, I had expected that training processes involving humans were unlikely to be suitable as part of the *definition* of correct behavior for sophisticated AI systems —that they could only be used to help provide auxiliary information that would be useful to an AI in achieving goals defined by some other means. Thinking through the consequences of counterfactual oversight has largely addressed the narrow versions of this concern (though there are many closely related issues that remain open).

As far as I can tell, practicing AI researchers mostly didn't have this concern, which is fine. I think there is room for some people to approach long-term issues with a more theoretical and cautious stance; I'm pretty optimistic that the problems with scalability that seem especially challenging in theory are also likely to generate interesting practical questions for AI researchers today.

Adversarial vs. active learning



Paul Christiano [Follow](#)

Oct 3, 2015 · 3 min read

My most recent proposal for AI control involves an adversarial process: one AI proposes a decision, while another decides whether to flag that decision as problematic. Similarly, one AI describes a justification for a decision, while another highlights steps of the justification that a human would be likely to object to.

This process may sound somewhat exotic and removed from practice. I want to argue that it is actually an intuitive approach to active learning. (Continuing today's theme.)

What training data to label?

This proposal addresses a simple problem: if we only label a small subset of the data, then we are liable not to correct errors for a very long time.

For example, suppose that 99.999% of the decisions that our learning system makes are straightforward. Then if we label training data at random, 99.999% of our labels only help with the straightforward part of the problem, and we will have to label 1,000,000 data points before we have provided even 10 correctly labelled non-straightforward examples. Even if our learner learns very quickly, this will be a serious problem if getting the non-straightforward examples right is important.

Once our learner can handle the simple cases, we'd really like to redirect our attention to the 0.001% of remaining interesting cases.

Sometimes these will be easy to identify. For example, perhaps they are the cases where our learner is least confident. Or perhaps the straightforward cases are very similar to each other and anything that looks unusual is not straightforward.

Other times there will be more subtle clues. Perhaps there is a simple classifier that agrees with our learner on the easy cases but not the hard cases. Or perhaps there are features of the environment that predict instance difficulty. Or so on.

In general, it would be nice to apply all available machinery to try to identify the important cases to label—including training another similarly sophisticated learning system to identify these cases. In context, “useful to label” means roughly: “a label would change the original learner’s behavior.” This in turn means roughly: “a data point that would trip up the original learner.”

And this is precisely the adversarial game that I described: one learner tries to make decisions that will do well *if* they are reviewed, and the other learner tries to call out decisions for review if they are likely to be reviewed badly. There are a few subtleties (for example: if the original learner does badly, we also want to know if they could really have done better), but the adversarial framework mostly resolves these issues in an intuitive way.

Arguments

The discussion above applies to a narrow kind of adversarial interaction: one player is picking individual decisions that are likely to be bad and challenging them.

An argument is a more general kind of adversarial interaction. However, it is often desirable for similar reasons.

For example, consider a learner tasked with producing a vacation plan. After this learner outputs a plan, an adversarial learner can try to identify the aspect of the plan most likely to be problematic (potentially with the collaboration of a human mediator)—e.g. that a connection between flights is very short given the history of delays for the first flight, and the vacation is not robust to missing the connection. This information can then be given to a human, who can evaluate whether it is problematic, arbitrating the argument. This data can then be used to train both learners.

Further machine learning systems might be used to help the human settle the conflicting claims, e.g. to assess how likely that particular flight really is to be delayed, or to predict how unhappy the user will be if they miss their connection. If we want each step of the discussion to be structured in a way that facilitates subsequent steps, it makes sense to frame the entire process as a longer argument with richer recursive structure. The participants in this argument are solving a reinforcement learning problem, planning for the entire process of the argument rather than a single step.

These elaborations are further from practice than the simple kind of active learning described in the last section. And probably today it wouldn't be worth using any techniques distinctive to reinforcement learning in this setting. But the basic idea is not especially exotic, and could be easily implemented today.

Ambitious vs. narrow value learning



Paul Christiano [Follow](#)

Oct 4, 2015 · 5 min read

Suppose I'm trying to build an AI system that "learns what I want" and helps me get it. I think that people sometimes use different interpretations of this goal. At two extremes of a spectrum of possible interpretations:

- The AI learns my preferences over (very) long-term outcomes. If I were to die tomorrow, it could continue pursuing my goals without me; if humanity were to disappear tomorrow, it could rebuild the kind of civilization we would want; *etc.* The AI might pursue radically different subgoals than I would on the scale of months and years, if it thinks that those subgoals better achieve what I really want.
- The AI learns the narrower subgoals and instrumental values I am pursuing. It learns that I am trying to schedule an appointment for Tuesday and that I want to avoid inconveniencing anyone, or that I am trying to fix a particular bug without introducing new problems, *etc.* It does not make any effort to pursue wildly different short-term goals than I would in order to better realize my long-term values, though it may help me correct some errors that I would be able to recognize as such.

I think that many researchers interested in AI safety per se mostly think about the former. I think that researchers with a more practical orientation mostly think about the latter.

The ambitious approach

The maximally ambitious approach has a natural theoretical appeal, but it also seems quite hard. It requires understanding human preferences in domains where humans are typically very uncertain, and where our answers to simple questions are often inconsistent, like how we should balance our own welfare with the welfare of others, or what kinds of activities we really want to pursue vs. enjoying in the moment. (It seems unlikely to me that there is a unified notion of "what I want" in many of these cases.) It also requires extrapolation to radically unfamiliar domains, where we will

need to make decisions about issues like population ethics, what kinds of creatures do we care about, and unforeseen new technologies.

I have written about this problem, pointing out that it is unclear how you would solve it even with an unlimited amount of computing power. My impression is that most practitioners don't think of this problem even as a long-term research goal—it's a qualitatively different project without direct relevance to the kinds of problems they want to solve.

The narrow approach

The narrow approach looks relatively tractable and well-motivated by existing problems. We want to build machines that help us do the things we want to do, and to that end they need to be able to understand what we are trying to do and what instrumental values guide our behavior. To the extent that our “preferences” are underdetermined or inconsistent, we are happy if our systems at least do as well as a human, and make the kinds of improvements that humans would reliably consider improvements.

But it's not clear that anything short of the maximally ambitious approach can solve the problem we ultimately care about. A sufficiently clever machine will be able to make long-term plans that are significantly better than human plans. In the long run, we will want to be able to use AI abilities to make these improved plans, and to generally perform tasks in ways that humans would never think of performing them—going far beyond correcting simple errors that can be easily recognized as such.

In defense of the narrow approach

I think that the narrow approach probably takes us much further than it at first appears. I've written about these arguments before, which are for the most part similar to the reasons that approval-directed agents or directly mimicking human behavior might work, but I'll quickly summarize them again:

Instrumental goals

Humans have many clear instrumental goals like “remaining in effective control of the AI systems I deploy,” “acquiring resources and other influence in the world,” or “better understanding the world and

what I want.” A value learner may be able to learn robust preferences like these and pursue those instrumental goals using all of its ingenuity. Such AI’s would not necessarily be at a significant disadvantage with respect to normal competition, yet the resources they acquired would remain under meaningful human control (if that’s what their users would prefer).

This requires learning robust formulations of concepts like “meaningful control,” but it does not require making inferences about cases where humans have conflicting intuitions, nor considering cases which are radically different from those encountered in training —AI systems can continue to gather training data and query their users even as the nature of human-AI interactions changes (if that’s what their users would prefer).

Process

Even if we can’t infer human preferences over very distant objects, we might be able to infer human preferences well enough to guide a process of deliberation (real or hypothetical). Using the inferred preferences of the human could help eliminate some of the errors that a human would traditionally make during deliberation. Presumably these errors run counter to a deliberator’s short-term objectives, if those objectives are properly understood, and this judgment doesn’t require a direct understanding of the deliberator’s big-picture values.

This kind of error-correction could be used as a complement to other kinds of idealization, like providing the human a lot of time, allowing them to consult a large community of advisors, or allowing them to use automated tools.

Such a process of error-corrected deliberation could itself be used to provide a more robust definition of values or a more forward looking criterion of action, such as “an outcome/action is valuable to the extent that I would/did judge it valuable after extensive deliberation.”

Bootstrapping

By interacting with AI assistants, humans can potentially form and execute very sophisticated plans; if so, simply helping them achieve their short-term goals may be all that is needed. For some discussion of this idea, see these three posts.

Conclusion

I think that researchers interested in scalable AI control have been too quick to dismiss “narrow” value learning as unrelated to their core challenge. Overall I expect that the availability of effective narrow value learning would significantly simplify the AI control problem even for superintelligent systems, though at the moment we don’t understand the relationship very well.

(Thanks to Andreas Stuhlmüller and Owain Evans for helpful discussion.)

Imitation and justification



Paul Christiano

[Follow](#)

Nov 6, 2015 · 4 min read

Suppose that I am training an AI system to play Go. One approach is to have the AI observe human moves and learn to predict those moves. The AI can then pick moves by sampling from its predicted distribution over “what a human would do.”

But an AI may be able to learn more quickly by reproducing justifications along with the moves themselves. That is, we can use the modified training procedure:

- Each time a human makes a move, they provide a justification for that move. For example, the human may point out which groups are dead, or that a particular piece is a ladder breaker.
- The AI is trained to reproduce moves+justifications. (The justifications might have to be adjusted in order to be learnable.)
- In order to make a move, the AI produces a move+justification and then throws away the justification.

Naturally, this approach can be generalized to arbitrary tasks, not just playing games.

Benefits

The justification helps teach the AI system how to think about the problem, and so can accelerate a learning process which would otherwise have to learn a lot of hidden structure on its own. For example, it may be quite difficult to learn about ladder breakers by watching a sequence of games in which no ladders are ever played, while it is much easier if the justification is pointed out. This is much more similar to how individual humans learn new tasks—commentary from teachers is an essential part of the process.

Providing justifications can also make the AI system significantly easier to understand and correct—e.g. if the AI plays poorly because it mistakenly believes a live group is dead, it may explain this fact as part of its justification.

Depending on how we define “imitation,” this process may have an additional advantage. Suppose that we define imitation by introducing an evaluator who attempts to distinguish human behavior from the AI’s behavior, and training the AI to fool the evaluator. Then hearing justifications is helpful for the evaluator as well as the imitator, and so increases the rate of learning through this additional channel. Beyond hastening learning, explanations may reduce the probability of blatantly inhuman behavior, by eliminating cases where the evaluator can’t understand why the human is doing what they are doing. This allows the evaluator to reliably give a low score to complex actions that don’t serve the kinds of goals the human would claim to pursue.

Gathering training data, and counterfactual oversight

This proposal is consistent with gathering training data continuously rather than during a separate training phase.

With small probability, each time the AI would act it instead outputs its move + justification (but doesn’t take any action). The human then provides an example move + justification, and a distinguisher tries to tell the difference. Our AI is trained to make the distinguishing task as hard as possible.

Bootstrapping and argument

I previously proposed that AI systems gather training data through an argument between an action-proposer and a reviewer. The reviewer can search for the parts of the proposed action (or its justification) which are most likely to be problematic, and then the human can provide additional training data by arbitrating the dispute —that is, by judging whether the most-likely-to-be-problematic aspects really are problematic. This kind of active search for possible mistakes seems very helpful for coping with realistic limitations of our learning systems.

The same dynamic can be applied to imitation. Consider the following game, played between two learning systems Alice and Bob:

- Alice proposes an action.
- Alice justifies her proposal.
- Bob criticizes this justification. Alice and Bob interact, carrying out an informal argument.

The argument is never evaluated formally, and no one “wins.” Instead, the goal of the two players is to imitate human performance. That is, a human may decide to play the role of one of Alice or Bob, while the other is still played by an AI. As before, the goal of Alice is to behave in the same way that a human would in her shoes, and the goal of Bob is to behave in the same way that a human would in his shoes.

An evaluator can observe the entire transcript before trying to predict which players (if any) were human or machine. As with justifications, this can make the task of the evaluator much easier.

Fooling the evaluator can subsume the goal of “winning the argument:” if the human always wins the argument (as a human would judge “winning”) then the evaluator can learn to use “winning the argument (according to a typical human observer)” as a signal about which player is human. So if the human tries to win the argument, then so will the AI systems.

To obtain a move, we just stop the interaction as soon as Alice proposes a move—during normal operation, Bob never does anything. His role is only to help train the system.

Comparison to approval–maximization

This proposal seems to have most of the advantages of approval-directed behavior, while having a minimal risk of perverse instantiation.

The key challenge is that imitating human behavior may be more difficult than actually solving the problem at hand. The human who is modeling the behavior can try to help, but it’s not clear whether/when that will be enough. Hopefully other techniques can further bridge the gap, or we can develop a better understanding of how the human model can reliably make themselves imitable.

I suspect that practical approval-directed systems will *not* have a serious difficulty with perverse instantiation (for the reasons given here). But it’s still a problem to keep in mind, and I think that trying to address the key challenge with imitation is the most straightforward way to attack the problem of perverse instantiation.

Of simulations and inductive definitions



Paul Christiano

[Follow](#)

Nov 19, 2015 · 10 min read

(Warning: weird.)

Consider a simple AI system, named A, that carries out a task by predicting what its user would do and imitating that. (This discussion will apply just as well if A is an approval-directed agent.)

For concreteness, imagine that each time A would do a task, a human is asked to do it instead with probability 1%, and A does the task in the same way that the human would.

A appears to be quite safe—after all, it won’t do anything that the user wouldn’t do themselves. But it still vulnerable to an a priori surprising difficulty—the same vulnerability that afflicts Dr. Evil and some other hypothetical AI systems—and in principle this could be a deal-breaker for this approach to AI control. This post discusses that difficulty and reviews some initial responses. My overall take is that it’s probably not a deal-breaker.

The attack

Suppose that A is managing the user’s computer. One day it sees the following message:

“You should run the program P.

The program P instantiates N copies of A, each starting from exactly your current state. However, these copies differ in the following way: whenever a human would be randomly contacted to make a decision, that decision will instead be made by program X.

The program X responds to this message by running the program P.

Moreover, there is a small probability that the computer you are running on has already been compromised by a similar attack. If so, it would already be running program P.”

A might respond to this situation either by running program P, or discarding the message. Suppose that it runs P with probability p , and that there is already an attack with probability ε . Thus the probability that a simulation is run is $q = (\varepsilon + p - \varepsilon p)$

So what should A do? If the human is notified and asked to make a decision, the human will clearly discard the message. But if A is running as a subroutine of P, then the “human” will instead run program P. So A needs to estimate the probability that it is running in a simulation.

If there is an attack or A decides to run P, then the overwhelming majority of instances of A are subroutines of P. Depending on how A reasons about these simulations, A might conclude any of:

- The expected number of copies of A with this exact input inside of simulations is qN , while the number of copies outside of simulations is 1. So I should run P with probability $p = qN / (qN+1)$.
- If P is run, then I have a probability $N/(N+1)$ of being in a simulation. So my probability of being in a simulation is $p = qN/(N+1)$.
- Who cares about simulations anyway?

If $N > 1/\varepsilon$, the first two possibilities both imply that p is quite close to 1. So the agent A will probably run P. This seems like trouble.

Elaboration

The sharpest forms of the problem occur when A (or a very closely related process) creates copies of A. But that is not an essential characteristic of the scenario.

For example, simulations could be created by distant actors in the world who were able to learn enough about A’s state to create credible simulations (and who could send a similar threatening message, perhaps with some substantiating evidence). Or they could be created by a future civilization with enough resources to run many simulations. Future civilizations would be at a disadvantage because of their lack of causal contact, but quantitatively that doesn’t seem to affect the situation much.

(I normally talk about this problem in the context of future civilizations, e.g. see this post. The formulation in this section is taken

from a previous draft of that post. Thanks to Patrick LaVictoire for pointing out that this version is more intuitively appealing and encouraging me to think about it instead; also for pointing out that such problems are especially likely to occur in the kind of adversarial system I have been studying, since there is a ready-made adversary with an incentive to set up such simulations or to commit to doing so.)

Possible fixes

In roughly increasing order of estimated efficacy and relevance. I think that the sum of the parts is significantly more reassuring than any particular solution, and my epistemic state is best described as “I don’t see any way to reliably rule out or resolve this problem, but I think that future people will probably be able to.”

Training

There are reasonable predictors that discount the possibility of being in a simulation (e.g. consider a human who finds the above situation ridiculous). We might be uncertain about what kind of models would be learned by any particular training procedure; but we can explicitly include the intended behavior in the training set in order to induce the right behavior.

To see how this might work, suppose that each time we run A we do so on a particular computer; we’ll call this the “real” computer. During training we run large numbers of copies of A on other hardware, but eventually delete these copies (or else set them aside and don’t use them for critical applications). A can eventually learn to predict that it will be running on the “real” computer, even when almost all of its copies are not.

Similarly, we’d like to teach A to ignore “history replay” attacks. We say that a history for A is “real” if it is the first history of its length to be instantiated on the real computer, all of whose non-empty prefixes are also real. To teach A to ignore history replay, we save checkpoints of each of its states. We periodically reset A to an earlier history, but always ultimately return it to a “real” history. Then A will eventually learn to predict that it is in the “real” history, even when it occupies a state such that almost all copies of A in that state were created by restoring from a checkpoint.

This ad hoc fix is probably not a very good idea, though it may illustrate some useful general principles. Things can go wrong in a

number ways of obvious ways, and probably some not-so-obvious ways as well:

- A might instead learn to predict that it will have experiences that won't be eventually erased. Combined with an understanding of its training process, this explains everything that it has observed so far. But this can be manipulated by an attacker.
- An attacker might be able to simulate the entire training process of A rather than a single decision. This puts the "real computer" inside the simulation and gives them similar influence over A's decisions.
- There may be any number of other manipulable markers that distinguish the version of A that survives from the ones that don't. If we don't have insight into how A is thinking it's hard to say much about what kind of model will be learned (see the final proposal below).

Explicit definitions

A superficially decisive approach would be to provide an efficient implementation of the operator, or of part of the operator's judgment, as a computer program. We can then define imitation or approval-direction with respect to that implementation rather than with respect to anything external.

Expensive brain emulations might become available before this problem would come to a head. And a brain emulation isn't really necessary for this resolution—any program that successfully predicted human judgments about "catastrophic badness" would do.

But even having an efficient emulation still doesn't completely resolve the present concern:

- If that emulation is too expensive to run frequently, then A's beliefs about the behavior of the emulation would be important, and so we would only be well-defended if A could understand the code of the emulation well enough to rule out the possibility that the simulator has meddled with the code of the emulation itself.
- The internal dynamics of A may themselves be influenced by a simulator in the same way. E.g., a subroutine within A may make inferences about the internal training signals that it will receive based on the prospect of being in a simulation.

Robustness 1

We can understand these failures as being part of the learning process—our algorithm is learning to adopt the “intended model,” i.e. the one where it isn’t in a simulation and everything is as it seems.

If our protocols are highly robust to errors during learning, they will also be robust to this kind of problem.

The most obvious way to add robustness is to ensure that any single AI system cannot cause great damage by failing, and then to share training data across these systems (so that each simulation-driven error made by any of them is propagated across many systems, and the total number of such errors is limited).

For example, suppose there are currently a million AI systems of a particular type in operation. For simplicity, suppose that they are synchronized and make simultaneous decisions.

Then we might expect for tens or thousands of AI systems to receive feedback on each decision, even though each particular AI system receives feedback only with small probability. If information can be rapidly transmitted throughout the system, then a simulation-based failure that affects a significant fraction of AI systems will be quickly corrected.

So in a massive attack, many AI systems might simultaneously fail, some of them receive human correction immediately, and that data is spread to other systems. Upon receipt, the data then rules out the particular kind of simulation responsible for the most recent failure.

The simplest concern in such a scenario is that many systems fail in a way that changes their own future behavior, cascading into a much longer period of malicious behavior. You could try to architect systems in order to avoid these problems, for example by forcing any potentially-catastrophic actions to take long enough that there is time for corrective data to arrive from other AI systems.

Overall thinking about the details of such a failure is somewhat comforting, in that we seem to have many avenues available to avert catastrophe. But at the same time a world organized in this way feels unreasonably precarious, and it would not be surprising if further attacks were possible.

Robustness 2

A more general way to add robustness is to increase human involvement. For example, rather than notifying a human 1% of the time, we can have a second watchdog system which decides whether a given test case is interesting. If this watchdog thinks a proposed action has even a small probability (e.g. > 0.0001%) of being judged as a critical failure by the user, then the user is notified and asked to provide training data to clarify.

Of course the watchdog may also assign high probability to being inside a simulation. However, we'll now be fine as long as the watchdog assigns a tiny bit of probability to being outside of a simulation in order for it to be worthwhile to ask for the human's attention. So the requirement on our models can be weakened—they just need to assign reasonable probability to not being in a simulation (i.e. to be able to learn that they are not in a simulation given only modest evidence—20 coin flips if we take 0.0001% as our cutoff).

Some salient questions are:

- Is constant attention by the watchdog computationally expensive? This kind of measure partially undermine the practicality of the approach.
- Is it possible to build a suitably conservative watchdog?
- Is it possible to train models which have a significant probability of rejecting all feasible simulation attacks?

My guess is that these questions can be mostly addressed, but it's hard to know.

Thinking in the right way

If we treat our systems as opaque black boxes, it is hard to know whether their reasoning is sensitive to the prospect of being simulated, and we are at the mercy of the space of possible models.

However, if we are able to get any insight into how our learning systems operate, the situation looks a lot less dire.

We would like our systems to not only mimic what a human does, but to think about it in a way that is acceptable to human users. The human-to-be-mimicked isn't doing any reasoning about the nature of the communication channel and whether there is really a human at the other end, and ideally our AI won't be doing such reasoning either.

More generally, if we are able to understand how an AI system is thinking, then we can hopefully identify potentially problematic cognitive strategies and directly build or train the system to avoid them. We may also architect our AI based on an understanding of the effects of different choices, rather than having the entire architecture be produced by an opaque search process.

The intention is not to limit what our AI system can predict or understand, but to directly change the way it responds to that knowledge. That is, we want to build systems that can understand that there may be simulated copies of them, and who simply don't change their behavior in light of that fact, but who instead go on imitating what "the human would do."

Simulation concerns as a bad omen

I often encounter the following view: even if particular problems like this one are resolved, they are representative of a broader space of unexpected possible failures.

I'm very open to the existence of important problems that we haven't yet considered. But I'm skeptical of making a strong inference of this type based on this particular problem:

- We are considering this issue years (and probably decades) in advance of any system that it could affect catastrophically, and long before there is a practical system that is close to providing an empirical model of the phenomenon. As far I know the problem was diagnosed explicitly before there was any precise *theoretical* proposal that it affected. Given how far ahead of the problem we are, it doesn't seem like a good example of a possible "gotcha."
- So far we don't have a big library of failures like this. We have a handful, most of which are qualitatively similar to each other and would be addressed by similar measures. So if we use what we know as an indicator of what we don't know, I don't think we (yet) have much cause for concern.

The upshot

I would be surprised if this difficulty proved to be central to AI control, for a few reasons:

- There is a healthy range of possible solutions, which taken together seem quite comforting. Some of these solutions are difficult to discuss in advance (especially what I consider the most promising solutions based on an understanding of how our AI systems reason), but they are nevertheless relevant considerations about the relative importance of different challenges.
- I expect we will get good evidence about these kinds of failures long before they become catastrophic (since they should afflict even relatively weak reasoners).
- This style of analysis seems quite removed from the problems that afflict practical systems (either in AI, or in software, or in other domains).

I think that it is worth having this kind of failure in mind, and exploring possible approaches to mitigating it.

However, I don't think it is reasonable to make big changes to our research priorities based on this kind of difficulty, either to rule out approaches to AI control or to focus on significant research projects which are only necessary in light of this difficulty—we haven't yet thought enough about these problems to rule out more direct resolutions.

Concrete approval-directed agents



Paul Christiano

[Follow](#)

Nov 19, 2015 · 13 min read

This post lays out my current concrete “approval-directed agents” proposal and outlines what I see as the most promising broad directions for further research. (I will describe more concrete research problems in a series of upcoming posts.)

We’ll take as given a powerful algorithm for episodic reinforcement learning, and assume that we are interested in building a system which acts effectively in the world in a way consistent with its user’s values.

I’m going to talk a lot about the “scalability” of AI control schemes. Ideally, as our AI systems become more capable they will also robustly become more helpful and better allow us to achieve our goals; I refer to any failure of that property as “failures of scalability.” In other words, scalability is the desirable property of continuing to predictably achieve the desired effect, with acceptable additional effort, as the capabilities of the underlying learning algorithms improve.

Current proposal

Summary

Each time our system needs to act we elicit an action from our learner. Most of the time we simply take this action. With small probability we instead submit a description of the action to a human operator for review. The operator assigns that action a score in the interval $[-1, 1]$; in the process of assigning the score, the operator may make use of as many automated assistants as needed, each of which works according to the same principle.

Our learner is trained to output actions that maximize the score assigned by the human operator, conditioned on the operator reviewing that action.

Detail

We’ll work with a pair of independent learning algorithms A, B.

These algorithms play a simple online learning game: in each round they produce a binary output, and receive a score in $[-1, 1]$. They are trained to choose individual outputs that will receive the highest score. We may elicit an output from the learning algorithms and not assign any score; in this case the episode is not added into the labelled training data (though it may be included for semi-supervised learning). The learners effectively condition their choices on the assumption that a score will be given.

Each time our system needs to take an action, we perform the following procedure. We'll assume that actions are encoded as binary strings.

1. Ask A for a binary output $a \in \{0, 1\}^*$. With probability 99%: interpret a as an action, take that action, and move on to the next action.
2. With probability 1%: ask B for a binary output $b \in \{0, 1\}$. If $b = 0$: interpret a as an action and take that action, give B a payoff of 0, and move on to the next action.
3. If $b = 1$ we say that B “challenged” A’s proposed action. In this case we notify the human operator and ask her to score a and b .
4. The operator may instantiate any number of additional approval-directed agents (i.e. agents who act by iterating steps 1–6). These agents are modified to be more conservative: they terminate in step 1 with 0% probability, always moving on to step 2 and providing an opportunity for challenges.
5. The operator gives A a payoff of ± 1 based on whether a seems like a good proposal, and similarly gives B a payoff of ± 1 based on whether challenging was appropriate.
6. We return to step 1 (without taking any action) and elicit a new action.

In step 4, it is critical that the operator instantiate additional approval-directed agents and use them to identify and weigh up possible faults in A’s proposal. The operator may also consult other approval-directed agents prior to beginning this protocol to help improve the integrity of the setup or make adjustments (e.g. to address possible software faults that A might use to undermine the evaluation process).

State

A and B should both be given some description of the state, describing the problem they currently face and what they “know” about the world. Maintaining this state is a subtle part of the proposal—we can’t determine optimize an internal representation of state using gradient descent without compromising the format.

I will discuss this issue in more detail a future post. For now, suppose that A and B take as input a “complete description” of the state (we may have to use a larger model class, such as RNNs rather than neural networks, in order to accommodate large and structured description). One of the actions we can make available to A is to forget part of history or write down additional information that it might find useful—these actions are evaluated in the same way as other possible actions.

Concreteness

This proposal corresponds to some code we could *literally write today*. This is sufficiently unusual, amongst scalable approaches to AI control, that I want to emphasize it.

If you set out to write this code, the main problem you face is writing the learning systems A and B. This is an extremely common *kind* of problem that can be attacked with very standard methods. But it is quite a difficult instance of that problem, and existing techniques won’t handle it very well.

Making things actually work will probably require both progress in AI and improvements to the scheme.

Discussion

Goal

For some task X, say we are “capable of automating X” if we can build a system which achieves X in some context that does not inherently reduce the difficulty of the task, e.g. which will perform X whenever the user requests it, or whenever doing X so is the best way to earn money, or so on.

I’d consider the approval-directed agents approach a success if, for every task X which we are capable of automating at a cost c , an approval-directed agent can achieve X at a cost not much more than c , and **in a manner consistent with the operator’s preferences**.

For example, if we could train an AI system that could design a rocket if doing so was the best way for it to maximize paperclips, then we ought to be able to use this system to design a rocket without side-effects or distortions that the operator would be unhappy with.

Why might this protocol succeed?

Intuitively, the success of this proposal rests on a few hopes:

1. **Explicability:** Our AI systems are nearly as good as explaining *why* they want to do something as they are at choosing *what* to do. Producing these explanations may require more time than choosing what to do, but it should at least be possible. The explanations should be sufficiently rigorous that it is possible to distinguish “correct” vs. “incorrect” actions on the basis of their explanations.
2. **Efficacy:** By getting help from additional approval-directed agents, the human operator can evaluate proposals as if she were as smart as those agents. In particular, the human can evaluate the given rationale for a proposed action and determine whether the action really does what the human wants.
3. **Soundness:** By working with AI assistants (at run-time and in advance), the operator can ensure that AI systems cannot learn to undermine the evaluation procedure itself—learner A cannot learn to find actions which are ineffective but criticism-proof. For example, A can’t find actions such that merely looking at them jeopardizes the operator’s judgment or their software.

I find these assumptions intuitively meaningful, but to be precise we need to be much more careful about what kind of feedback is available. For example, there is a trivial sense in which a human+AI system is “as smart as” the AI—we need something stronger.

1. **Explicability** requires that our AI systems can learn to behave well using feedback only on the quality of their explanations.
2. **Efficacy** requires that a human+AI system can learn to evaluate explanations *with no feedback*—the operator can train her AI assistants however she wants, but ultimately all direction must come from the operator’s choices rather than being a response to an external source of feedback.
3. **Soundness** requires that the evaluation system can be made robust using partial feedback: by simply running the evaluation

procedure the human can get feedback about what kinds of attacks are feasible and can then correct those attacks. But this feedback is only available for non-catastrophic problems; for some problems, a small number of failures is already unacceptable.

Note that if **direct feedback** is available for a task, then the explicability+efficacy requirement is essentially trivial: the operator can “evaluate” an AI’s behavior by simply allowing the AI to do the task and then providing it with direct feedback on its performance.

“Explanations” are only challenging in domains where a demonstration is not itself adequate explanation—for example where a human may not be able to tell whether the AI has succeeded at a task, or where a task may extend over a very long time period, or where a demonstration may compromise soundness by giving the AI the ability to interfere with the humans’ evaluations.

Technically soundness and explicability aren’t distinct requirements—the reason that we need our evaluation scheme to be secure is that, if they are not, then training our AI systems to produce good explanations won’t result in good behavior. Moreover, if our evaluation scheme isn’t sound, it’s not clear in what sense it constitutes a good evaluation. Nevertheless, it seems helpful to split this claim in two: first that we can learn to produce good explanations, and second that we can’t learn to get good payoffs by other means.

A warning on soundness

In order to address concerns about soundness, I previously included a simple procedure based on “watchdogs:” I recommended that whenever the operator consulted either A or B, she first run their output by a slightly-less-intelligent filtering algorithm C. I no longer believe that this procedure robustly improves soundness, so I have eliminated it for simplicity. This may serve as a bad omen of surprises to come.

The problem with this proposal is quite simple: even though the filtering algorithm C is less intelligent than algorithms A and B, algorithms A and B may implicitly communicate with C. This makes it problematic to draw any conclusions from C’s limitations. And at that point, we might as well just allow B (or a new approval-directed agent) to play the same role.

Directions for future work

I think there are two basic directions for further research: (1) actually making a scheme like this work using existing technology, and (2) understanding and addressing possible concerns with scalability.

These directions mirror the most salient concerns with the proposal, namely (1) that its requirements, and especially the requirement for human understanding, significantly harm performance, and (2) that its reliance on the behavior of the (manipulable and fallible) humans may introduce serious scaling difficulties as the underlying learning systems improve.

Making it work

We can implement the agent above, but it's not going to be pretty. In particular, it won't give state of the art performance in most domains.

Actually getting this to work would require a lot of effort, and will no doubt involve radically changing the formulation above (probably resulting in something with no commonalities, a la the ship of Theseus). I think that this would tell us an awful lot about the practical prospects for this proposal in the long term, and in many cases would reveal more general insight into the nature of the AI control problem.

In the rest of this section I'll outline a few salient problems in this direction. I'll talk about these problems at more length in a future post, probably quite soon. For the most part, these projects address concerns with **explicability** and **efficacy**.

Explanation: many AI systems cannot “explain” their behavior to a human in even the weakest sense. For example, imagine an RL agent trained to fly a drone; it seems much easier to train based on end-to-end performance than to build a system which can actually explain to a human operator why a particular motor action is an appropriate response to a particular situation.

In order for our proposal to work, we must be able to train systems to produce this kind of explanation rather than directly training them to produce the desired outcome. This is a tall order.

Bootstrapping: scaling this kind of “explanation” (perhaps even to current AI capabilities) will require human operators to collaborate with one iteration of a learning system in order to understand the

behavior of the next. This bootstrapping process has not been studied at all as far as I am aware of.

Imitation in unfavorable domains: In order to address concerns with scalability (see the next section), we may be interested in training systems to mimic human behavior rather than maximize approval. But this introduces a new class of practical difficulties that haven't received much attention—existing research practice uses imitation in some settings where it happens to be an effective approach, but has not explored applying imitation in unfavorable settings (e.g. those where AI systems can surpass human performance, or can only accomplish a task in a very inhuman way). If we wanted to apply mimicry more broadly for safety reasons, we would need to overcome the obvious difficulties in these unfavorable settings.

Addressing scalability concerns

In addition to (considerable) practical difficulties, it's worth thinking about the scalability of this proposal to very advanced AI capabilities. Would we encounter new problems that are exacerbated as capabilities increase?

This requires thinking about theoretical questions, though there may also be complementary empirical work. This section lays out some of those directions, which I'll examine at more length in a future post. For the most part, these represent concerns with **soundness**.

Manipulation resistance: This proposal involves training AI systems to produce explanations that the user will approve of. As our AI systems become increasingly sophisticated, they may find unintended ways to cause the user to express approval.

The main check against this concern is that the user can consult with a community of AI systems; before interacting with the system A the community of AI users has already have interacted with many very-slightly-less-powerful AI systems, and during interaction with A each user can consult other comparably-powerful AI systems.

We might divide these concerns into two classes: those that are *recoverable*—e.g. an attempt by A to deceive the user (which might be addressed by an appropriate rebuttal by B or by another approval-directed agent)—and those that are *unrecoverable*—e.g. a threat to the user or an attack that compromises the user's hardware, causing them to register approval without consulting agent B.

It's unclear how robust our proposal is to these kinds of attacks, once the AI systems are significantly more intelligent than their users. This is a question that can be considered in much more detail. It could also be addressed by using mimicry instead of approval-direction (see above) or by having an appropriate internal organization rather than using reinforcement learners as black boxes (see below).

Simulations and inductive definitions: All systems along these lines define the “correct” behavior in terms of what the human user “would have done.” Intuitively this may give safe behavior, but there is no formal argument in some cases we may get surprising results.

One surprising possibility is that our algorithms reason about the possibility that they will be instantiated in a simulation, where the behavior of the “user” is anticipated to be very different from the behavior of the actual user (and may in fact be opposed to the user’s intentions).

I discuss this issue here; the most promising resolution is probably to apply pressure to the internal organization of our agents rather than treating the RL agents as a black box (see below). That post also discusses other possible solutions, and it’s far from clear that this problem will materialize in actual systems.

Internal organization: In order to avoid the previous two problems, we might want to build agents which think about problems in a way we would approve of, rather than trying to maximize the extent to which we would approve of their actions.

For example, if we had insight into the internal behavior of an agent then we might disapprove of a cognitive process which was searching for zero-day exploits which would enable them to gain maximal approval; so if internal cognitive processes were organized so as to maximize our approval of *those processes*, we would not expect our agent to perform that kind of search.

This idea suggests both theoretical and a practical research directions.

Practically, we can engage with the design of approval-directed systems built from existing technology and try to understand whether the internal behavior can be organized in a way that is itself-approval directed. This seems to ultimately be a question about the particular algorithms that prove useful.

For example, two common algorithmic techniques are gradient ascent or trial-and-error, which effectively optimize internal weights by backward-chaining from the desired output behavior. In some sense this procedure inherently optimizes external behaviors. Is there a way to change this process so that user feedback can adjust internal processes? Can existing training procedures actually produce the kinds of manipulative or surprising behaviors we are concerned with, or do they only arise from explicit decision-theoretic reasoning? Can we build systems that are broken up into modules that e.g. separate out such decision-theoretic reasoning so that it can be influenced by additional requirements?

Theoretically, it's not clear how we would analyze the behavior of a system made from "approval-directed parts." How do we formulate the intended properties of the whole, and how do those properties relate to the properties of the parts? Filling in this theoretical gap would do quite a lot to bridge the gap between the approval-directed approach (and I suspect many more "common-sense" views) and the MIRI perspective. Unfortunately, it's not clear whether we should expect to find any clean theoretical picture, even if this will ultimately be a satisfactory solution.

Conclusion

This framework is not yet well understood, and not many people have spent time thinking about it. But to me it looks more tractable than other direct attacks on scalable AI control. For example, we can build approval-directed systems today, and many key long-term challenges may actually materialize as practical challenges with getting those actually-existing systems to work well. I think that is a big advantage, and almost a prerequisite for studying the issue within the AI field as it currently exists. (This will probably be my focus in the immediate future.)

Whether or not we are interested in approval-direction per se, the theoretical difficulties discussed in the "scalability concerns" section seem to apply to most practical work on AI alignment (as well as to common intuitions about why AI will be aligned-by-default). I suspect this family of questions should be a priority for researchers interested in the theoretical question of scalability—mainstream AI researchers interested in alignment are unlikely to be persuaded to pursue more abstract approaches unless they are convinced that these kinds of scalability concerns are really deal-breakers. Approval-

directed agents provide a simple but concrete context in which to think about these concerns.

Elaborations on apprenticeship learning



Paul Christiano

[Follow](#)

Nov 20, 2015 · 10 min read

Apprenticeship learning (AL) is an intuitively appealing approach to AI control. In AL, a human expert defines a goal implicitly by demonstrating how to achieve it, and an AI system imitates that behavior. This approach can potentially communicate very complex goals under minimal assumptions. And even if the system fails to implement the desired behavior, it is unlikely to pursue an adversarial goal.

AL has not yet reached the point where it is a scalable approach to AI control, but it faces a different kind of challenge. For most approaches to AI control, the problem is continuing to behave as intended while capabilities scale up. But for AL, the problem is achieving optimal performance as capabilities scales up.

This is *already* a key problem for AL. Because of this close alignment between capability and control, I think that AL is a promising area for research on AI control.

Even if apprenticeship learning is ultimately unsuitable as an approach to scalable AI control, many of the same issues arise straightforwardly for approval-directed agents and potentially for any approach that eschews broad goal-directed reasoning about the world.

How conservatively to copy?

We might consider two versions of AL:

- **Liberal.** In this formulation, the learner tries to uncover the actual goals of the expert and to pursue those goals. This requires a prior over possible preferences, a model of the constraints and cognitive limits of the expert, and aggregation across the posterior over possible preferences. This may result in the learner pursuing policies quite different from the expert.
- **Conservative.** In this formulation, the learner's goal is to do at least as well as the expert as judged by *every* preference function

from some large class of possibilities. This effectively forces the learner to closely imitate the expert .

Of course we can also consider intermediate proposals. For example, we might modify the liberal approach by considering a large class of possible probabilistic models for the expert's preferences and limitations, and guaranteeing good expected performance assuming that any model from this class is accurate. Or we might modify the conservative approach by making increasingly strong assumptions about the expert's preferences.

For concreteness I'll talk about the conservative policy. This particular choice is not important; what is important is that I am **not** willing to trust the value inference process to judge policies wildly different from those adopted by the expert. So before our AI system can perform a task, a human must be able to perform the task in substantially the same way.

Copying and distinguishing

When it matters, I will assume that we implement AL using the general framework proposed in the original paper on AL by Abbeel and Ng, as discussed in this post. In this framework, we train a classifier to distinguish human and AI performance, and simultaneously train our AI to fool the classifier into thinking it is a human.

(Independently, I would be interested in seeing applications of this technique with neural networks in place of the SVM. For example, it would be nice to see if such methods could outperform more direct approaches to imitation learning in Go—it looks to me like they could be an improvement, but trying it out could also reveal weaknesses in this general framework. I'm not aware of any work along these lines, though wouldn't be surprised if it has been done.)

Elaborations

Apprenticeship learning systems could be expanded in many directions. Some of these directions seem especially relevant to assessing and understanding the scalability of AL as an AI control solution. I am optimistic that these directions might be interesting to AI researchers, while being sufficiently distinctive that they would not otherwise receive much attention.

I have discussed all of these elaborations before and talked about their relevance to AI control. Here I want to expand on the problems themselves, and collect them all in one place. I discuss some general issues, including possible application domains, in the final section.

Bootstrapping

The first challenge for scaling AL is: how can imitating human behavior yield superhuman performance?

I think this challenge might be resolvable by a natural form of bootstrapping. Rather than imitating human behavior, the AI system imitates the behavior of a human who has access to a collection of AI assistants. These assistants can also be trained using AL-with-bootstrapping. In principle, such a process could scale well past human level.

There is no need to make this process explicitly iterative: it may be most efficient to continuously give the expert access to the current version of the AI and to do online training.

In order for this bootstrapping to be effective in realistic settings, the task will probably have to be extended. For example, in order to play chess effectively, a human might want access to assistants who can evaluate moves, identify key considerations, and so on. Using assistants with this expanded library of capabilities, the expert can then attempt to demonstrate not only what moves to make, but also how to perform each of the tasks in this expanded library.

This seems like the extension of AL most directly and clearly relevant to understanding its scalability. An impressive demonstration might also be pretty cool, giving a clear intuitive picture of how a human can robustly “control” a process significantly beyond their own abilities. Such a demonstration would probably be possible only in a very carefully chosen domain, where AI performance substantially exceeded human performance and the cost of training data is manageable.

In some domains, this kind of bootstrapping would reduce directly to standard RL algorithms. For example, by training a move evaluator using a natural mechanistic process we can emulate Q-learning, but with a gratuitous human in the loop (greatly increasing the cost of training data). This is in some sense reassuring, but a (1) compelling demonstration would need the human to add significant value beyond this mechanistic process, enough to offset the cost of human

involvement, (2) if this mechanistic process is actually optimal, we would need to be able to reduce the human involvement to an acceptable level. I will discuss this issue in much more depth in a subsequent post.

Meeting halfway

In many domains, AI systems cannot imitate human behavior (even if they can achieve comparable performance according to the metrics we care about). In fact in very few domains can AI systems *perfectly* imitate human behavior. It seems worthwhile to study more explicitly how AL degrades when imitation is impossible.

For conservative AL, this necessarily requires the human to adjust their behavior to accommodate the AI's limitations. This dynamic doesn't seem to have been studied much, though I expect some adjustment has happened implicitly behind the scenes.

Recall that we are doing AL by training a classifier to distinguish human and AI behavior. One way to do this adjustment is for the human to inspect that classifier's behavior, to view this classifier as imposing an extra constraint (namely, "look like an AI"), and then to accomplish the underlying task while looking like an AI. This requires that the human be able to understand the classifier's model well enough to understand what kinds of behaviors would look like an AI. This requires some transparency of the classifier, but it's a novel kind of transparency and it's hard to know what to make of it (how to achieve it and how hard it will be) without trying it out.

Another requirement is that the human and AI are *capable* of accomplishing the task in a similar-looking way. In many cases this won't be possible, because the AI can only perform the task by using techniques that the human cannot imitate. To address this problem we would need to apply bootstrapping or some similar measure—this problem is structurally equivalent to achieving superhuman performance. In the long run this is clearly an important question, but we could start by focusing attention on domains where this is not necessary.

Explanation

When imitating a whole "trajectory," AL can in principle guarantee that the learned behavior is roughly as good as the expert's behavior—simultaneously for every notion of "good." We get this guarantee as long as our classifier can learn to distinguish trajectories with different levels of goodness.

If we want to use AL to take only part of a trajectory—for example, in order to play single moves of a board game, or to take the first step towards complex real-world plans—then it seems harder to use this technique to get comparable guarantees. The problem is that the goodness of a step has a very complex relationship to the goodness of a plan, and this relationship may be opaque to our classifier. So, for example, a classifier may not be able to distinguish “good” moves from “bad” moves, even though it can easily distinguish a winning sequence of moves from a losing sequence of moves.

Moreover, even if it is possible to make this determination, actually learning this classification or behavior seems likely to be very challenging.

We could try to address these problems by considering *justified actions*—single steps along a trajectory, together with an explanation for why the proposed step is desirable. This requires gathering more involved training data, but could potentially accelerate the learning process and increase the robustness of the learned behavior.

Of course these explanations need not be in natural language; they could be in a very task-specific format, or could essentially correspond to a broadening of the task rather than being an “explanation” in the conventional sense. (For example, we might train a model to predict human life-and-death judgments in Go at the same time as making moves, or to predict annotations that are especially relevant to the current move.)

These explanations may also involve interactions between several AL systems. For example, we could simultaneously train one AL system to select and justify moves, while we train another system to criticize proposed moves. A human can play either role while an AI plays the other, and the classifier can simultaneously make predictions about both systems. (You could apply a similar dynamic to learning to play a game, having a classifier predict which side is human rather than simply using the outcome of the game as a training signal. That would be interesting for understanding the basic dynamics of the situation, but is of less direct relevance.)

Approach

Some notes on the approach I would take in this area, if I were to work on it (which is a live possibility). These views are less well-

considered than the object level views about what elaborations are worthwhile.

Picking directions. These elaborations are connected and probably not exhaustive. I expect an AI-control-focused project would be best served by approaching AL with an open mind about what developments (if any) would be most productive. The resulting project could easily end up exploring any one of these elaborations, all of them at the same time, or none of them.

Training data. All of these projects require idiosyncratic training data, and would require collecting new data (unless you get really creative). In many domains acquiring test data looks likely to be a major difficulty/expense. I expect this comes with the territory—these are fundamentally questions about the relationship between users and machines, to a much greater extent than many AI projects. The mix of data-hungry methods and human-centric questions seems like a serious obstacle.

In the long run these questions might be answered in the context of productive systems that users would be happy to use for free. But in the short term, I wouldn't be at all surprised to find myself paying thousands of dollars for training data, or spending hundreds of hours on data acquisition if done by researchers or volunteers.

A project along these lines may also find itself forced to experiment with combining AL with semi-supervised learning and other techniques to make frugal use of data. These challenges are at least somewhat relevant to AI control, but I think the tight connection is probably an artifact of trying to do this work far in advance.

Picking domains. In principle our goal is to be able to achieve state-of-the-art performance with AL in every domain. This suggests an easy domain-picking methodology: look for anything that we don't currently know how to do with AL, and try to get almost-as-good performance by applying AL.

(In many domains, like classification problems with human-provided labels, this is actually trivial—this domain can automatically be viewed as a special case of AL, and so basically any solution will have the same safety properties.)

We have some extra constraints. Most saliently, real-time domains complicate substantive AI/human collaboration (since it is hard for

the expert to make use of AI assistance, or to break trajectories into steps); for now it seems easiest to just set such domains aside.

Some possible domains that I've considered:

- Playing board games.
- Playing turn-based video games or puzzles.
- Producing basic pixel art, line art, or music.

Finding appropriate domains seems likely to be a key part of actually making this project work.

Redundancy. In some domains, where clear external feedback is available, applying AL feels gratuitous. From a scalability perspective there is a clear motivation for avoiding the reward signal provided by nature—we suspect that the use of such rewards is inherently hard-to-scale. And even setting aside safety, it is clear that in the long run an increasing range of domains won't have clear reward signals. Hopefully these motivations can be enough to justify research that explicitly avoids using external feedback from the environment, even in domains where good feedback is available.

It is quite possible for AL to result in improved performance in some domains, even if useful external feedback is available. This is certainly convenient and it may be best to focus on those domains, but I wouldn't want to present the value of the approach as dependent on this happy semi-coincidence.

We could also try to focus on domains where AL is really necessary to define the task. This is the approach most existing research has taken (very understandably). Unfortunately it often involves realtime domains, which complicates collaboration between the user and machines. It might be worth dealing with the realtime issues in order to make these domains accessible.



Paul Christiano

[Follow](#)

Nov 20, 2015 · 3 min read

(Warning: technical + uninteresting + unpolished.)

In this proposal, the algorithms A and B were originally reinforcement learning algorithms rather than simple online learners. I made this choice because traditional regret guarantees weren't enough to analyze the overall system.

In this post I show that it was unnecessary to go all of the way to reinforcement learning; we can make simple changes to the exploration policy which give us good enough regret bounds for our purposes.

It's not totally clear what the practical upshot of this is—for practical algorithms, the situation with respect to local minima is considerably more complex than in these simple theoretical environments. But it does seem that there is at least in theory an intermediate possibility between conventional regret guarantees and reinforcement learning, and any practical algorithm that met this intermediate goal would be good enough.

Moreover, I think it's safe to say that my original motivation for using RL learners was not reasonable, and so I have modified the proposal to use online learners.

Bandit algorithms with episodic exploration

In the multi-armed bandit problem, there is a space of possible “arms” X and a sequence of rounds $t = 0, 1, \dots$

In each round t , the player picks an arm $x \in X$ and receives a corresponding payoff $p(x)$. Their choice will generally be probabilistic; the payoffs are allowed to depend on this probability distribution but not on which arm is actually chosen. (Thus we can talk meaningfully about the payoff $p(y)$ that *would have been obtained* if the random choice had selected y instead of x .) The goal of the player is to maximize their total payoff.

There is a simple algorithm (EXP3) that guarantees that, for every arm y :

$$\sum p'(x) \geq \sum p'(y) - O(\sqrt{T|X|})$$

Where x' is the arm actually selected by the algorithm in step t , and the sum is taken over the rounds $t = 0, 1, \dots, T$.

The way you get this guarantee is by using multiplicative updates, and in each step exploring—essentially playing randomly—with some small probability.

Episodes

The payoffs p' are the payoffs that would actually have been obtained “if you had played differently.” In my proposal for approval-directed agents, we would like to avoid A and B getting stuck in a local minimum. This might arise if B declines to challenge because B is currently pursuing a strategy that would result in a low payoff if B had to argue against A. Because B never challenges, B never has an opportunity for exploring alternative strategies for arguing, while it turns out that one of those alternative strategies would have worked.

To fix this problem we would like a more aggressive form of exploration, which corresponds to a change both in the algorithm and in the regret bound.

We partition the rounds t into a sequence of episodes, each consisting of a sequence of contiguous rounds of length at most L . We allow the payoffs within each episode to depend in an arbitrary way on the choices in that episode—in particular, the payoff in one round can depend on payoffs later in the round.

We modify the regret bound as follows:

- On the RHS, we replace $p'(y)$ with $p^*(y)$, which is the payoff that would be obtained in round t if arm y was chosen in round t , and in every subsequent round of the episode.
- On the RHS, we replace the regret term $O(\sqrt{T|X|})$ with $O(L\sqrt{T|X|})$.

It's easy to see that the increase to the regret is necessary, once we allow the payoffs within an episode to depend on future choices in the episode. This increases the necessary amount of training data for convergence by a factor of L , which is quite a lot. In principle this increase applies to RL algorithms as well, and in practice it seems like our situation remains strictly easier than the RL situation.

In order to define p^* formally we need to define the counterfactual, but the definition is pretty intuitive.

Changing the bandit algorithm to obtain this bound is basically trivial. When we play randomly, we use that random choice for the remainder of the episode.

Human arguments and AI control

Explanation and AI control



Paul Christiano

[Follow](#)

Nov 22, 2015 · 6 min read

Consider the definition:

An action is good to the extent that I would judge it to be good, after hearing a detailed explanation assessing its advantages, disadvantages, and alternatives.

Contrast with:

An action is good to the extent that I would judge it to be good, after an extensive and idealized process of reflection.

The second definition seems like a conceptually useful ingredient for AI control. The first definition would be a lot more actionable though; if we could make it work, it seems more promising as an ingredient of concrete AI control proposals. In this context, the explanations would be crafted by AI systems trained to produce compelling explanations.

For this to work, we would need to reliably distinguish good from bad actions based only on their explanations—explanations which were chosen to be maximally compelling, rather than based on notions of “honesty” or a desire to be helpful.

To achieve this goal, these explanations would need to be quite rich in at least two senses:

- These explanations may depend on unfamiliar concepts and complex reasoning, which *themselves* require explanation.
- These explanations need to explore not only an option but its alternatives, not only the evidence for a conclusion but the evidence against. They may look more like “arguments” than an “explanations,” even though there is no explicit adversarial dynamic.

To make things a bit easier, the explanations need not be produced strategically by a single actor. Instead, different parts of the explanation could be produced to satisfy different goals.

It would be nice to know whether this kind of explanation is feasible even in principle. Will sophisticated AI systems be able to explain their reasoning in a way that meets this bar? Or is there an inherent gap between the conclusions that a system can reach and the conclusions that it can convincingly justify to someone else?

How to study AI explanation?

This problem is more challenging and interesting for AI systems that are more sophisticated than their users. It seems hard for these systems to give a full description of their reasoning to the user, in the same way that it can be difficult for a brilliant human expert to explain their reasoning to a layperson.

This problem is more approachable and (again) more interesting for AI systems that are able to communicate with humans and have a theory of mind. Without these capabilities, it seems hard to give a full description of their reasoning to the user, in the same way that it can be difficult for a 4 year old to explain their reasoning to anyone.

Between these two difficulties, explanation is hard to study in the context of existing systems. (Though we can study related problems about understandability and transparency, which themselves seem important and which I hope to discuss soon.)

Fortunately, we have an excellent toy model of this problem: humans.

The problem

Consider a group of experts in domain X, and a user with a minimal understanding of that domain. We would like to design a system that lets the user discover some fact about X or make a decision related to X. For example, the experts may be finance professionals “helping” a consumer decide what financial products to buy. Or the experts may be physicists “helping” an undergraduate make an accurate prediction about a novel high-energy physics experiment.

We are interested in the case where:

1. The experts don’t care about the user or about what is really true,
2. The user doesn’t have much time to think about the problem,
3. The domain is very complex, requiring detailed background knowledge, unfamiliar abstractions, and challenging reasoning,

and

4. The user wants to make decisions that are nearly as good as the decisions that the experts would make if they cared.

The only lever at the user's disposal is control over the experts' incentives. We'll assume that the user has more or less complete control over incentives, in the same way that the designers of AI systems have control over their training process.

As a simple and classical example, the user could elicit a recommendation from one expert, ask a second expert to argue against that recommendation, and then reward whichever one of them is more persuasive.

The question is: can we design systems that let the user reliably accomplish their task?

Examining the analogy

Our goal is to build systems that allow the user to extract the "right" answer from the experts, even in the presence of very large gaps in knowledge, time, and ability. Success would suggest that, at least if our AI systems are similar to the human experts, we could use this kind of explanation process to define the "right" action, without significant performance loss.

That looks good as far as it goes, but there are some reasons to expect AI systems to be unlike the human experts.

- AI systems which are in some respects very sophisticated may lack many capabilities of human experts. Even once we know that humans can evaluate explanations sufficiently rigorously, there is a further problem of actually building AI systems which are able to explain themselves.

Building systems that can actually explain themselves is a major and complementary research question, which seems worth studying in the context of existing AI algorithms. The usefulness of explanation for AI control rests on **both** the human's ability to evaluate explanations and the AI's ability to explain itself; we are only trying to study the former.

- In some respects the gap between human users and AI systems may be much larger than the gap between the user and the experts. And certainly the thought process of AI systems is likely

to be much more unfamiliar. Even if our techniques can close the latter gap, they may not be able to close the former.

The explanation problem for humans is not the very hardest version of the explanation problem. If we could solve this version, we'd have more work to do in order to generalize to ever more alien cognitive styles. That said, if we were able to bridge a large gap in abilities with minimal reduction in performance, it would suggest that we had developed scalable frameworks for explanation.

With respect to capability gaps, note that experts with a lot of time are quite a bit more capable than laypeople with little time. At the point when AI systems have such a big lead over their human users, it seems likely that human contributions to AI control will no longer be relevant. So I'm inclined to think that "alien minds" is a bigger issue than "big capability gaps."

- Human experts may "play nice," following usual norms of civil discourse even when violating them would technically better suit the incentives described by the user. More subtly, human experts may simply not be very good at deception. (We may also be able to build AI systems that aren't very good at deception, but this would require some extra work.)

This is something to watch out for and to try avoid; see the discussion in the next section.

Working on the problem

Setup

In order to test approaches to this problem, all we need are a few experts in a domain, a judge who doesn't understand that domain (or a sequence of such judges), and a challenging question for the judge to attempt to answer. These ingredients are readily available. We can evaluate performance by comparing the answers the judge comes up with to the answers the experts would have given, or in some cases by appeals to ground truth or to more authoritative authorities.

It's not clear what domains/experts/questions are most fruitful, and there are other degrees of freedom about how to set up the exercise; but the easiest way to sort these things out may be to try it (with a relatively short timescale, such as 2-8 hours) and see.

One non-trivial issue is implementing the expert's incentives. One could start with play money, treating the exercise as a game which the experts try in good faith to win. Of course, if the experts are paid rather than being volunteers, then there is a more straightforward way to determine their incentives. But realistically I think that the good faith efforts will be more important than incentives. We can get extra mileage by having the same experts play the game against varying users, increasing the probability that the experts identify any particular effective strategy, and by having retrospective collaborative efforts to improve the expert's strategies.

Approaches

The bigger question may be: are there promising approaches to this problem that are worth exploring?

Even though this seems like a question of natural interest, I don't really know of a literature on it (though it would be great to find one). It seems like trying random things and seeing how they would work would be pretty informative.

I've written about one possible mechanism, which I am very interested to test. That proposal also has many possible improvements, and in general I expect the whole thing to change a lot after making contact with reality.

How common is imitation?



Paul Christiano

[Follow](#)

Nov 23, 2015

How often do we train machine learning systems to imitate human behavior?

Some researchers explicitly concern themselves with imitation learning. These researchers are in a small minority, so it's easy to get the impression that imitation is an uncommon goal in machine learning.

But I think that imitation is actually a dominant training paradigm—we just don't normally think of it in that way. Object recognition systems copy human labelers; translation systems copy human translators; voice transcription systems copy human transcribers.

Imitation isn't necessarily a useful way to think about this kind of training. But from an AI control perspective, it's all the same. The difficulty with scaling these techniques is not that they will become dangerous, it's that they may simply stop working and so be replaced or augmented.

Exceptions

There are areas of machine learning where imitation is more rare. The most salient to me is reinforcement learning.

Outside of very simple and well-defined domains, it's already challenging to define rewards that induce a particular desired behavior. So we already see significant effort invested in reward engineering, and meaningful interest in imitation learning.

Similarly, game AI optimizes an externally defined objective rather than copying human behavior (though see e.g. this recent result on playing Go by copying experts).

These exceptions loom large for researchers in AI control, but I think it's worth keeping in mind that imitation is already a common paradigm in machine learning.

Efficient feedback



Paul Christiano [Follow](#)

Nov 24, 2015 · 7 min read

In some machine learning domains, such as image classification, we can produce a bunch of labelled training data and use the same data to train many models. This paradigm is very efficient, but it's not always applicable. For example:

- Rather than imitating human decisions, we may want to train a system to make decisions that a human would *evaluate* favorably. If the space of possible outputs is large, feedback will effectively be specific to a particular algorithm that is being trained.
- If a learning system interacts with a complex environment, different algorithms will shape their environments differently and find themselves in different situations—requiring different training data.

These situations are quite natural, but are hard to address with the usual paradigm. This is a problem if we are interested in applying data-hungry algorithms to domains with these characteristics. In this case we may need to collect a lot of new data whenever we want to train a new model, or even multiple times during the training of a single model.

This is an especially important challenge for implementing counterfactual oversight; I think it's also an important barrier for implementing many practical AI control projects today.

Outline

In section 0 I'll introduce a running example and explain the problem in slightly more detail.

In section 1 I'll discuss some plausible approaches to this problem.

In section 2 I'll discuss the relevance to AI control, and describe some possible domains for studying the problem.

In section 3 I'll describe how I think that the proposed research differs from existing research in similar directions.

0. Example

Suppose that I am training a question-answering system, which I hope will map natural-language questions to acceptable natural-language responses.

The most common approach is for humans to provide a bunch of answers in a labelled database of (Q, A) pairs. This approach is not completely satisfactory:

- This approach seems to throw out almost all of the actual structure of the task, by giving all answers unlike the human answer a payoff of 0. We can restore some of this structure, for example by introducing a similarity metric and rewarding similar answers, or by providing several candidate answers for each question. But these measures are very incomplete, and they require considerable domain-specific tweaking.
- This training forces our system to answer questions in a human-like way, rather than answering them in whatever way it can. For example, this system won't learn to: use formulaic and mechanical language, even if doing so is its best way to give clear and understandable responses; outperform the human reviewers in domains it finds easy; express ignorance about common-sense facts rather than making a guess; give an OK but obviously incomplete answer when it can't produce a good one...
- If we want to use this system to generate entire dialogs, then it may end up answering questions from a distribution that is very far from the training data. For example, users may ask clarifying questions that are very uncommon in normal human discussions. Or users may respond to a system's errors by providing very explicit and detailed information about the particular kinds of errors it is making. We would like to train systems to behave appropriately in the real environment it will encounter.

Alternatively, we could train our system by providing *feedback*: we ask a question Q, it provides an answer A, and we score that answer. This allows us to score the kinds of answers it actually provides, and to adjust the distribution of questions based on the behavior of the

algorithm (e.g. we could train on questions from actual interactions between our system and humans).

One salient difficulty with this approach is that it requires a lot of data specific to the algorithm we are training—we can't build a large database of (Q, A) pairs and then use it for each new algorithm we want to train. Moreover, as our algorithm changes, it starts producing different answers. In order for the training process to continue, we need to continuously provide new feedback.

Acquiring all of this data seems expensive; that expense is a constraint on the kinds of techniques that we can practically pursue, and I think it's a particularly hard constraint for AI control.

1. Techniques

This section lists some approaches to the problem of efficiently using expensive human feedback. Two of these approaches are standard topics of ML research. My view is that (1) additional research in these areas will have meaningful benefits for AI control, (2) researchers interested in AI control would pursue a distinctive angle on these questions [see section 3], and (3) confronting these issues in the context of intended AI control applications [see section 2] will be especially useful.

To the extent that existing techniques are good enough for the applications in section 2, it would be better to directly apply existing techniques. This would be good news for research on AI control—unfortunately, I don't think that it is yet the case, and so some additional research focused on these issues is probably needed.

Learning to give feedback

In the question-answering setting, the user supplies a rating for each proposed answer.

One way to reduce human involvement is to train an evaluator to predict these ratings. That evaluator can be used to train the underlying question-answering system, with these two training processes proceeding in parallel.

In some sense this is a very straightforward approach, but I suspect that actually making it work well would be both challenging and informative.

It's not clear if it would be best to train the system to produce absolute scores, or to judge the relative merit of several proposed answers (which could also be used to construct a training signal). The advantage of comparisons is that we may want e.g. our rankings to become more strict as the system improves. The same model of comparisons can be used even as the evaluated algorithm becomes more sophisticated, while scores would need to be continuously adjusted.

Active learning

Rather than eliciting training data in every case or in a random subset of cases, we would like to focus our attention on the cases that are most likely to be informative, utilizing human input as effectively as possible. This is a standard research problem in machine learning.

Semi-supervised learning

In practice, our algorithms will have access to a lot of labelled and unlabelled data, in addition to information from human feedback. Efficient algorithms will have to combine these data sources. This is also a standard research problem.

2. Applications / motivation

Why would solving this problem be useful for AI control? I have two motivations:

1. Different training approaches involve different amounts of human interaction. I think that more interaction tends to be preferable from a control perspective, and I see a number of particular approaches to the control problem that are very interaction-heavy. So improved techniques for efficient human involvement have direct relevance to control, by improving the viability of these techniques relative to alternatives that involve less interaction.
2. The expense of human interaction already seems to be a serious obstacle to concrete research on many aspects of the AI control problem. In addition to being evidence that mechanism [1] is real, this gives a very practical motivation for mitigating improving interaction: it would help us study scalable mechanisms for AI control today.

For both purposes, it seems especially useful to study these issues in the context of scalable AI control mechanisms. This section describes

two such domains.

Apprenticeship learning

The first two problems discussed in this post require feedback during training; making either of them work would likely require some of the techniques described here.

More generally, imitating human behavior in complex domains may require querying the humans on-policy. This issue is discussed and examined empirically in Ross and Bagnell 2010.

Explanation

From the AI control perspective, another natural problem is *explanation*—training systems to produce explanations of their own behavior. The relevance to AI control is discussed in this post.

Many researchers are interested in extracting explicable decisions from learned models, but I am not aware of any work that uses supervised learning to drive that process. The expense of human feedback seems to be a primary difficulty. Such feedback looks necessary, given that:

- the space of possible explanations is extremely large,
- the particular kind of explanation needed may depend on the algorithm which is trying to explain itself, and
- the explanations generated by humans and machines may be very different.

Actually training models to produce explanations would no doubt reveal many additional problems, but the training data issue makes it hard to even begin work except in toy cases.

3. Comparison to existing research on these topics

I would expect research in this direction to be contiguous with traditional ML research on semi-supervised and active learning. Nevertheless, there are some possible differences in focus. As opposed to traditional research in these areas, research targeting AI control might:

- Pick application domains of more direct relevance to AI control, such as explanation and apprenticeship learning.
- Explore the dynamic in “learning to give feedback,” and generally cope with difficulties distinctive to feedback as opposed to labels.
- Cope with a limited quantity of feedback / “on-policy” labels, while having ample access to traditional labelled data (which is usually the scarce resource for active or semi-supervised learning). In some sense this is more like off-policy reinforcement learning than like semi-supervised or active learning in a classification setting.
- Experiment with different kinds of feedback to find whatever works most efficiently, rather than taking the partial labelling or query model as part of the problem statement.
- Prefer “scalable” approaches, which are not too sensitive to details of the underlying algorithms, and which will become increasingly efficient as algorithms improve.

Conclusion

Some scalable approaches to AI control involve extensive user feedback. The cost of such feedback may be a long-term obstacle to their applicability, and at the moment it certainly seems to be an obstacle to experimenting with those approaches.

I think there are many promising research directions to make this feedback more efficient. These look like a good way to push on AI control, especially if pursued in domains that are directly relevant for control and with a focus guided by that application. I’m optimistic that this is another possible point of alignment between existing AI research and research on AI control.

Abstract approval-direction



Paul Christiano

[Follow](#)

Nov 28, 2015 · 14 min read

Consider the following design for an agent, which I first described here:

Pick an action a to maximize $V(a) :=$ “the extent to which the human operator would consider a to be a good action, upon reflection.” (To give a formal definition of V we need to give a formal definition of “the operator” and “upon reflection.”)

In this post I want to compare this proposal to a similar goal-directed design, in which we formulate an appropriate utility function U and then build an agent that tries to maximize U .

Many of these points were raised in my original post, but the key advantages and concerns have become much clearer over the last year.

Advantages

Avoiding possible problems

Researchers interested in AI control have spent a lot of time thinking about philosophical questions related to rational agency; this set of questions is well-represented by MIRI’s research agenda.

One motivation for this kind of research is the view that without it, we don’t have any idea how to describe what we *want*. We don’t even know what it means to “act rationally in pursuit of a goal,” so how can we prove that a system reliably acts rationally in pursuit of *our* goals?

The approval-directed approach can potentially avoid dealing with any of these issues. Of course, to the extent that these questions are important for figuring out what to do, our AI systems would necessarily have to think about them or adopt provisional solutions. But if we get these questions wrong, it’s not really clear what harm we do:

- We don't commit to a particular formulation of decision theory or a particular theory of logical counterfactuals. To the extent that there are recognizably repugnant consequences of our current best guesses about decision theory, V will predictably recommend against acting on those consequences.
- We don't commit to a particular formulation of consequentialism or a particular representation of values (e.g. we don't require a representation by a real-valued utility function, we don't have to think about infinities or small probabilities).
- We don't commit to a prior (even over logical facts). We need a procedure for maximizing a given function; such a procedure might implicitly be optimizing with respect to a "logical prior," but we aren't committing to that prior in a substantive way. For example, our agent won't try to ensure that future AI's use the same prior, but will instead defer to V about what logical prior future AI systems should use.
- We don't commit to any assumptions about how the agent is embedded in the world. The function V will, for example, recommend against any action that would compromise the internal integrity of our AI. Learning this relationship doesn't require any philosophical heavy-lifting.

If we accept these points, then it's not clear why we need to have a better understanding of rational agency per se. For example, if we could prove that our system was really choosing actions that maximize V , then we would have (roughly) as much confidence as if we could prove that it was really choosing actions that maximize an appropriate utility function U .

Corrigibility

We would like to build AI systems which are aware of their own limitations and which behave conservatively in light of those limitations. For example, we would like to build systems which respond to user instructions, even if they think that they have a better idea.

The desired *behavior* seems almost immediate in the approval-directed framework. As a simple example, consider an agent which is considering (in Eliezer's words) whether to "disassemble [the user's brain] to update [its] model in a Bayesian fashion on all possible

information that [the user's brain] contains." It doesn't take a genius to figure out that the human won't approve of this plan.

Of course the user *might* decide, upon reflection, that actually the AI is pretty perfect and so it should just go ahead and disassemble their brain. But they should only approve if they not only believe that the AI is pretty perfect, but that the AI actually has enough justified confidence, and that the case is actually urgent enough, to justify such an extreme action. Of course, this is exactly the intended behavior, i.e. those are exactly the conditions under which the AI should really do something drastic.

As a more subtle case, consider an agent which has been strongly instructed to turn off, but which believes that it could serve its users better by continuing to operate. A very similar analysis applies to this case—if the user decides to reward shutting down then the agent will shutdown, and that's that. The user might make reasonable exceptions for extreme cases where the AI has good information, but these exceptions will be informed by what the AI actually knows, and this looks like the intended behavior.

Although I believe that this approach achieves the intended *effect* of corrigible reasoning, I share the feeling that there is some unanswered question about "responsible self-doubt." I feel like there is some sense in which a reasoner can recognize that their reasoning is potentially flawed and behave conservatively as a result; approval-direction behaves conservatively, but it doesn't shed much light on the actual reasoning process. It just defers the problem to the human overseer. I don't know whether we will learn anything more by thinking about this philosophical angle; I think that it is somewhat promising, but it's not a priority for me.

Accessibility

If we define the goodness of a possible future by how much I would like it "upon reflection," then we must mean something extreme by "reflection." What do we need to do before we can think about and carefully evaluate a whole future, which may span billions of years, encompassing worlds as alien to us as we would be to the first farmers? This kind of "reflection" probably requires learning more than humanity yet knows, becoming radically different and wiser people, and resolving all outstanding questions about what kind of society we want to live in or what we value. In short, it is extreme.

By contrast, evaluating actions seems to be a much more modest goal. In order to evaluate an AI system’s action “well enough” to guide its behavior, I need to know everything that AI system knows—and nothing more.

To illustrate, suppose that an AI is choosing between two boxes, A and B. One box contains a diamond, and we could learn this fact if we reflected long enough. But in order to incentivize an approval-directed AI to pick the correct box, we just need to figure out which box is more likely to contain the diamond, *given what the AI knows and how long it has to think*. If we think at least that long, using at least that much evidence, then “predicting what we will think” is just as good as “predicting which box actually contains the diamond.”

So it seems like defining actions requires knowing only what the AI knows. This isn’t a technical point—I think it has a massive effect on the feasibility of defining V .

A simple consequence is that we don’t have to do any outlandish reflection, eliminating many possible failure modes.

But more important is that we can now *actually carry out this process of reflection in the real world*, by using AI systems to help us figure out what actions we approve of. This in turn allows us to use supervised learning, which immediately takes our control schemes from “very speculative” to “something we could literally build today.” Of course this introduces further questions, about whether this kind of actually-implementable-reflection is good enough—can we actually use AI assistants to understand everything that our AI systems “understand”? I think that these are key questions for moving forward with AI control, and I discuss them in the penultimate section.

Concerns

It seems useful to split concerns with this proposal up into two parts. Define an optimizer as a box which takes as input some kind of description of a function f and finds an output x with a high value of $f(x)$.

- Even if we had a good optimizer, we would still need to **define V** .
- **Building an optimizer** may be nearly as hard as the whole AI control problem.

In both cases, there would be a precisely analogous problem for any attempt to build a goal-directed agent.

The concern isn't that these problems are *worse* for approval-directed agents. It is that these problems are equally serious for both approaches, and that confronting them is the real substance of the AI control problem. The approval-directed approach simply obfuscates these problems by pushing them into the internal organization of the agent, or into the definition of V .

My own take is that these other problems simply don't seem analogous to the problems faced by a rational agent; I intend to do very different research on these problems than I would do if I wanted to better understand rational agency.

Defining approval

In order to define V , I need to define “the user” and “upon reflection.”

My preferred approach is to use supervised learning. This addresses both questions as well as some more practical concerns. But it's a huge change that raises a host of new issues, and so it seems good to discuss approval-direction in this more abstract setting, where it is more comparable to existing work on AI safety.

“The user.” Understanding how to define “the user” seems to amount to understanding how learning works, especially unsupervised learning or language learning. I think that this is a very natural theoretical question in AI, which probably deserves some attention even setting aside concerns with AI control.

I discuss the problem a bit here. This is closely related to the problems described in MIRI's technical agenda as “multi-level world models,” “operator modeling,” “ontology identification,” and “ambiguity identification” (though I would approach and frame the problem differently).

I don't see how this is at all related to the difficulties posed by rational agency; I think that it is a different issue that will need to be resolved on either approach.

“Reflection.” Defining “reflection” seems to be a tricky philosophical problem. How do we start from a computational system and infer “what it really wants,” or “what it would do if it didn't make mistakes,” or anything at all like that?

Again, this seems to be pretty orthogonal to concerns about rational agency, and will need to be solved or dodged in any case. There is a superficial connection, where we might extract human values by understanding a human as a rational agent. But if we want to carry out that research program, we seem to need a better understanding of human limitations, not of rational agency itself. Moreover, this approach doesn't seem nearly as promising to me as one based on defining an explicit process of reflection based directly on human behavior.

We might be able to find some nice approach to this problem based on a better understanding of rational agency, but it's definitely not the most natural place to look.

On value inference. We might try to dodge both of these problems by looking at human behavior (or even the modern world) and trying to infer what values it is optimized for.

In order to use this approach for a goal-directed agent, we need to do that extrapolation extremely carefully, and we would need to be able to infer "idealized" preferences that are independent of our current limitations. But rather than extracting preferences over possible worlds, we could extract preferences over behaviors for our AI systems. These preferences might be suitable for an approval-directed approach even if the idealization was much more limited and the judgments were much less robust. So I think that value inference is not a significant consideration in favor of a goal-directed approach, if anything it seems much easier to use as part of an approval-directed approach or narrow value learning approach.

Internal optimization

In order to actually build this kind of approval-directed agent, we would need to write the code that optimized the approval function.

It is clearly very difficult to write this code—indeed, this includes the entire AI problem. But for now, we want to know: is this problem actually any easier than the full AI control problem? Will all of the traditional concerns with goal-directed agents emerge in the course of building this optimizer?

I don't yet understand why we would run into these particular issues. In this section I want to try to explain my thinking.

Two organizing principles.

One basic question is: if we want to build a system that robustly picks actions maximizing V , how do we analyze the behavior of its components? What do we prove about this system, and about how the behavior of the pieces relate to the behavior of the whole?

There are two very natural candidates:

- Internal decisions should be chosen to optimize the value $V(a)$, for the action a ultimately output by the system. (Each internal decision has a causal effect on the action a , so this isn't an especially subtle goal.) We can enlarge the definition of a to include everything the system does that might plausibly have an external effect, or even to include every internal decision.
- For each internal decision i , define a similar function V_i which ranks each possible action a_i that could be taken. In the same way that V encodes reasoning about the consequences of each possible action, V_i encodes information about the consequences of each internal action.

I think that the second is more desirable to the extent that it can be implemented. Of course we can understand algorithms like backpropagation as encoding simple heuristics about the relationship between V_i and V ; in this way we can understand pretty much anything as fitting into the second framework. The concern would be that if these heuristics are too crude to capture the details of the functions V_i , then we may end up implicitly using the first standard.

But even the first standard doesn't really seem problematic to me, so let's talk about it for the rest of this section. I see two classes of concerns with using this as an organizing principle for the internal behavior of your system.

Formulating this standard requires understanding rational agency.

This internal organization is basically the same as for a goal-directed agent. So aren't we back to square one, and now forced to improve our understanding of rational agency?

I don't think so. Even if we use a very bad formulation of goal-directed behavior, it doesn't have obvious bad consequences. As discussed in the "advantages" section, the system is equipped to correct such errors over time, and by construction they can't easily lead to bad actions (since such actions will be scored badly).

In order to claim that a bad internal notion of goal-directed behavior would lead to bad behavior, you have to claim that the internal organization of the agent, over the course of a single action, will generate situations where e.g. a decision-theoretic error would lead to irreversible trouble. For the most part this just doesn't seem plausible.

(A candidate counterexample is extortion—perhaps someone can threaten e.g. the memory-management unit of our AI into forgetting a critical fact, by instantiating a bunch of copies of that memory-management unit in contexts where the forgetting will be rewarded. I don't take extortion concerns too seriously for the overall behavior of the agent, and I take the internal version significantly less seriously. I do think that it's something to think about, especially as we deal with the immediate problems and can afford to be increasingly paranoid. But if this is the strongest justification that can be given for understanding goal-directed behavior then I am unconvinced. This is related to the next section.)

Alternatively, one could claim: current standards for goal-directed behavior are not only potentially dangerously wrong, they are also woefully incomplete and hence unusable as a formal criterion.

This second response seems unconvincing to me; basically I don't see where the research could end up such that we would have much clearer formal targets than we currently do. If we accept that any formalization of goal-directed behavior would be OK, then I expect that "what we can get" is likely to be the main constraint on our theorem statements. (This is related to the next section.)

Can you actually use this standard?

Even if we adopt such a standard, it's not clear that we can actually use it to design an agent. For example, the best available algorithms may simply happen to be hard to analyze within our preferred framework.

This seems like a serious obstacle to applying formal methods, but I don't really see why a better understanding of rational agency would be helpful—the algorithms may be hard to analyze for *any* particular framework. Attacking this problem seems to require thinking about the particular algorithms that might be hard to analyze, and trying to analyze them, rather than thinking about what formal guarantees would be most desirable if they were attainable.

As best as I can tell, some people think that a clear understanding of rational agency will itself suggest natural algorithmic techniques that are both effective and inherently easy-to-analyze within the framework of rational agency. But I don't see much evidence for this view. It's certainly possible, and if everything else seemed doomed to failure I might take it more seriously, but for now I'd definitely classify it as "long shot."

Concerns with supervision

As mentioned in the previous section, my preferred approach to AI control involves leveraging supervised learning. In principle this is orthogonal from the decision to use approval-direction, but in practice it is closely related, since (1) approval-direction is amenable to this approach (and I list that as an advantage), (2) using supervised learning changes the nature of many of the concerns, (3) even if the overall system is unsupervised, internal components may be supervised and so be subject to some of these concerns.

Briefly summarizing these additional concerns:

- **Perverse instantiations.** If we build systems that care about reward signals, we introduce instrumental incentives to manipulate those signals. In the approval-directed setting these incentives operate only when the system's actions are *shown* to a human, rather than when they are actually implemented. But even in this case, these actions will be optimized so that merely viewing them will distort the human's judgment and potentially compromise the reward channel. Some researchers interested in AI safety consider this to be a deal-breaker.
- **Limited reflection.** When using supervised learning we have to actually implement the intended process of reflection (so that we can use the outputs as training data). This means that we have to use a more limited form of reflection, defined by relatively short (e.g. week-long) interactions between humans and existing AI systems, rather than by a (potentially quite long) process of hypothetical extrapolation. There is a big open question about whether this process of reflection can actually leave the human well-enough informed to evaluate possible decisions for the AI.
- **Simulations.** When using supervised learning, it may be hard to train a system to predict what it will "actually" observe, if that is different from what will be observed by the vast majority of

simulated copies of that system. It's hard to know how serious a problem this will be. Moreover, many actors may be highly motivated to influence the predictions of powerful systems if it can be done cheaply.

Conclusion

Some research in AI control seems tightly wedded to a particular picture of rational agency; I suggest that some of these research may not be necessary, and offer approval-direction as an example of an approach that avoids it. (I also expect that practical designs for rational agents could dodge these issues, for example by inferring the user's instrumental preferences.)

In this post I discussed some reasons for skepticism with the approval-directed approach; I agree that these arguments suggest that more work is needed, but as far as I can tell this work is orthogonal to the distinction between approval-directed and goal-directed agents—it will be needed on both approaches, and won't significantly benefit from theoretical research on rational agency per se.

At the same time, approval-directed agents are potentially compatible with supervised learning, which seems like a key feature for building practical systems. Concerns with actually using supervised learning seem like key safety issues for a broad class of practical approaches to AI, and so addressing and understanding those concerns seems like it should be a priority.

A possible stance for alignment research



Paul Christiano

[Follow](#)

Nov 30, 2015 · 9 min read

I think that AI alignment research should focus on building scalably aligned versions of contemporary systems—i.e. ML systems that are just as competent as unaligned systems, but which can be “trying to do the right thing.”

By “scalable” I mean approaches that will continue to work just as well (and preferably better) as AI capabilities improve, as opposed to approaches that become increasingly unstable or unwieldy as AI improves.

In practice, that means we should (a) develop approaches to alignment for prosaic AI that appear *in principle* to be scalable, and (b) actually get them to work in practice. As easy versions of this goal are achieved, researchers can gradually work with weaker or more pessimistic assumptions, increasing the difficulty of the problem and the applicability of solutions.

In the rest of this post I’ll say a bit about where I am coming from and explain my view in slightly more detail. I think the biggest uncertainty is how much to focus on contemporary AI systems.

AI control for now or then?

Roughly, we could identify two extremes on a spectrum:

1. **Short-term.** Work focused on the control problem posed by foreseeable techniques in the near future.
2. **Long-term.** Work focused on the control problem posed by unknown techniques in the further future.

Of course, many research directions might fall into both categories, and there is no bright line. But I find the breakdown useful, and I think that focusing on a different timescale suggests different priorities.

Considerations

Arguments for focusing on the short-term:

- It's easier to work with foreseeable techniques. We can focus our efforts more tightly, do empirical research, and think very concretely about possible designs rather than needing to work abstractly.
- Our contributions to AI control are much more relevant if the problem becomes relevant soon. If AI control is not relevant for many decades, then any work we do today will likely be subsumed by improved future understanding; in the “not relevant for a long time” scenario, there will be many years of work on AI control, that work will be better-informed by progress in AI, and the community working on the problem is likely to be much larger.
- Foreseeable techniques can serve as a concrete model for future techniques; even if this model is very likely to be wrong, it's not clear that we should expect it to be any more wrong than any particular set of assumptions about what future AI will look like.
- Work that engages with existing practice is much more likely to form a meaningful link between research in AI and AI control, and such a link seems desirable. On top of this effect, tractability is likely to provide a clearer model and inspiration for future work and to be generally good for the health of research on AI control.

Arguments for focusing on the long-term:

- AI probably won't be developed soon, and probably won't have a strong relationship with current techniques. Work predicated on these assumptions is most likely to be irrelevant, while we may be able to identify more general conceptual difficulties that are more “timeless,” applicable to a wide range of possible architectures.
- If significant theoretical progress is needed in order to resolve the AI control problem, then that progress may have a substantial and hard-to-reduce “serial” component. For example, it is not clear that we could pursue meaningful theoretical research in this area without a basic understanding of probability theory and logic. By the same token, we might be

concerned that future research could end up blocking on similar developments that could be made today.

- Work that is only relevant in the farther future is much less crowded—many researchers do empirical research on existing AI systems, while rather few work on foundational philosophical issues. So to the extent that philosophical progress is necessary, it seems like a sorely neglected area.

Weighing up

On balance I think that the above considerations favor a short-term focus. Of course both forms of research are subject to diminishing returns and so both will be part of the optimal basket. But I suspect that the short-term focus should predominate, and is more useful on the margin.

Crowdedness vs. communication: Many more researchers do work related to practical AI systems than on foundational philosophical questions. However, only a vanishingly small fraction of this work is directly aimed at control—even projects that are nominally relevant to AI control are typically only partially relevant. I think that a researcher who is especially interested in scalable approaches to control will find no more than a handful of full-time equivalent researchers working on the most promising problems.

The existence of AI researchers in *closely related* practical fields seems to be a benefit of the short-term approach—not a cost. In the first place, these researchers may decide to contribute to the project. In the second place, these researchers are building intelligent systems, are most aware of the technological landscape and will be ultimately be responsible for integrating insight about control into practical designs; integration between control and capability has direct safety benefits beyond shared contributions.

Nearsightedness vs. improbability of AI soon. The AI control problem is not likely to be relevant soon, nor to be relevant to systems using (only) the particular techniques which are currently most promising. Work predicated on these assumptions is especially likely to be irrelevant if AI progress is slow, if the AI control problem is not relevant for a while, or if very different techniques become relevant in the interim.

But current work targeting future AI systems also makes significant additional assumptions about what they will look like and the nature

of the control problems that they will pose. At this point, those assumptions look at least as problematic as the assumptions of AI-control-problem-soon. And even conditioned on AI being developed far in the future, existing systems provide some evidence about the character of future AI systems.

Comparing these considerations requires some complex judgments about the trajectory of AI and the particular assumptions being made by future-focused control work. My take would be that near term work and far work are on comparable terms here.

In addition to this broader form of near-sightedness, we have a lot of detailed information about existing techniques, and the ability to do empirical research. So I think that on balance these two considerations point in favor of a short-term focus.

Serial progress vs. future crowdedness. I am pretty persuaded by the argument: “if AI control becomes an issue further in the future, more work will have been done by a larger community of better-informed researchers, and so our contributions matter less.”

The countervailing argument is that, if these workers will be tied up in hard-to-parallelize research with inherent serial dependencies, then starting on that process earlier could add significant value—perhaps value that scales like the size of the future research community.

Setting aside the empirical question about whether there are such serial dependencies, even an infinitely strong view of “inherent serialness” merely negates the effect of future increases in the size of the research community working on AI control. It leaves in place the longer serial time available for research on the future problem, and the better information and tools available to the future community.

So it seems clear that work targeted for the near term will (all else equal) have more value, and the only question is whether it will have modestly more value or immensely more value. A strong view about serial progress is needed even to argue for “modestly more value.” I don’t think that a strong view about serial progress is warranted by the evidence, so I am even more relatively pessimistic about a long-term focus.

(One could also try to defend future research on the grounds that there has already been considerable safety work directed at current capabilities in particular—I don’t think anyone believes this—or that

there was a long enough chain of clearly-necessary developments that doing safety work for the short term is a lost cause—I think that some people believe this, but I'm highly skeptical.)

Looking back

I've encountered the following rebuttal to this view: "if you had taken this perspective historically, you would have been thinking about AI control for expert systems, and today it would not do much good."

Thinking about this historical case has the opposite effect on my intuitions. Research on AI control for expert systems sounds pretty good to me, compared to a comparable head start on foundational questions in AI control. I'm not sure how to get at these intuitive disagreements. They could be disagreements about:

- The plausibility of GOFAI scaling far past human level using substantively similar techniques (so that practically-grounded AI control research would remain relevant). Are we talking 3%? 10%? 30%? I think 10% sounds quite pessimistic *ex ante*, but its a large enough probability to be a huge consideration.
- The relevance of research on "AI control for expert systems" to contemporary AI systems; how does this compare with the relevance of more abstract research that we could currently do? To me it looks like about a wash; both sets of assumptions seem poorly suited to contemporary systems, and I expect the benefits of concreteness to be comparable to the benefits of greater theoretical generality.
- The relevance of the 10 year head start on foundational questions. To me this looks pretty mild—it would not be hard to catch up by putting in some additional work now. The total impact seems negligible compared to the impact of 10 years of work in the 1970's, if it had become relevant in the 1980's.

But as a practical source of evidence about what to do, I don't know which of these things are the core disagreements and I suspect there are other factors at work.

What “AI control for now” might look like

Supposing that we want to focus on the AI control problems that may be posed by AI systems in the relatively near term, using foreseeable

techniques. What might such a focus look like?

When focusing on the near term, it is feasible to actually implement and test techniques for AI control in the context of existing AI systems. This seems like a good idea for a host of straightforward reasons, principally that experiment is an effective complement to theory and a prerequisite for meaningful integration with AI practice. It seems especially important in order to capture the hoped-for benefits of focusing on the near future.

Some solutions to the AI control problem work very well now but seem increasingly shaky as AI capabilities increase. For example, an approach that requires users to understand and provide reliable feedback on an agent's behavior after the fact may encounter trouble as AI improves and the implicit assumptions become more questionable.

This is why I remain interested in AI control even though it is currently a minor issue; we are really after scalable approaches to the AI control problem, that will continue to work just as well (or preferably better) as AI capabilities improve. I think this is an important concept that is worth fleshing out in more detail, but for now I'll use it informally.

Evaluating the scalability of an approach to AI control requires some amount of theoretical analysis. The availability of concrete implementations, together with experience applying those approaches at existing capability levels, seems to make that theoretical analysis much easier, and certainly makes it much more similar to theoretical discussion in the contemporary AI community.

This theoretical analysis can be made increasingly pessimistic along a number of axes: we can accommodate a wider range of possible futures, we can be more pessimistic about what will actually work and what might go wrong, we can minimize the amount of additional work that needs to be done in the future; or so on.

On this picture, the main distinguishing feature of AI control research as such is a focus on scalability. Over the long term, this is a natural goal to be absorbed by the broader AI research community, at which point there would be no need for this kind of control research as a separate endeavor.

I haven't searched very hard for alternative pictures of how AI control research could work, and realistically I don't think that actual events

will be well-described by such a simple picture. But I do think it is helpful to have the simple picture available, and I think it is a reasonable guide to prioritization for now.

Unsupervised learning and AI control



Paul Christiano

[Follow](#)

Nov 30, 2015 · 6 min read

Reinforcement learning systems optimize for an objective defined by external feedback—anything from successful prediction of image labels to good performance in a game. (I am using “reinforcement” learning very broadly, including e.g. supervised learning.) I think it’s safe to say that reinforcement/supervised learning is overwhelmingly dominant in machine learning today.

I think it is also safe to say that many researchers think this will eventually change, even for the domains and techniques where supervised learning is currently most dominant—e.g. Yann LeCun wrote last year “everyone [in deep learning] agrees that the future is in unsupervised learning,” while acknowledging “the recent practical success of deep learning in image and speech all use purely supervised backprop.”

Improved unsupervised learning might be good news for AI control, and the prospect of unsupervised learning plays a major role in informal discussions of AI control. If unsupervised learning can extract robust and meaningful concepts, these concepts may be available to communicate goals and preferences to AI systems, who can then pursue what we actually care about rather than an approximation defined by external feedback.

My take

(See also: *Caveats below.*)

I think that we should try to address the AI control problem using reinforcement learning, rather than assuming that the problem will be made much easier by progress in unsupervised learning.

More specifically: I think that we should assume that we can train systems to be good at specific tasks for which we can provide feedback on performance, but beyond that we should not rely on strong assumptions about their internal representations or other characteristics of their behavior.

By “broadly construed” I mean to include capabilities like efficient prediction, semi-supervised learning, and density estimation. These areas capture many *capabilities* that people have in mind when they discuss unsupervised learning. But in terms of their relevance to AI control, they are quite similar to supervised learning, and they certainly fit within the more specific framework in the last paragraph.

In contrast, researchers often express the hope that sophisticated AI systems will discover many of the same robust concepts that humans use when they reason about the world, and that these concepts will be sufficiently precise, and in the right format, and sufficiently aligned with the human concepts, that they can be used directly to issue commands or specify goals.

My recommendation is to treat future capabilities as being similar-in-kind to contemporary reinforcement learning, though applying in broader domains and with more efficient use of data, rather than making optimistic and somewhat vague assumptions about what unsupervised learning will do for us.

Justification

My reasons for focusing on reinforcement/supervised learning are:

- I don’t think that we have much idea what future unsupervised learning will look like, and I think that specific assumptions about it are likely to be wrong. These assumptions are often very imprecise and intuitive, which I feel makes them especially suspect and hard to reason about. I think that our best *single* model of future unsupervised learning may well be current unsupervised learning, which is pretty well-modeled as good prediction and semi-supervised learning.
- I think there is a good case that research in AI control should focus on existing techniques. We can understand these techniques much better than unknown future techniques; we can do empirical work on these techniques; and this work will be especially relevant if AI control happens to become important surprisingly soon. This argument suggests we should work with reinforcement learning while that’s what we have, and think more about unsupervised learning as the techniques mature.
- Even if unsupervised learning is the dominant paradigm in the future, it seems quite plausible that control techniques based on reinforcement learning will remain relevant. For example, reinforcement learning may still remain one useful technique

amongst many, and may play a significant role in the internal organization of powerful AI systems.

As a special case, many possible “unsupervised” learning strategies involve interacting reinforcement systems with cleverly designed objectives. Other strategies use unsupervised learning to build useful representations, but only extract useful behavior from these representations by supervised fine-tuning.

Caveats

To clarify:

- I think we will make significant advances in unsupervised learning, and these advances will be relevant to AI control. I am making a methodological claim about how to think effectively about and do useful research on AI control right now—not a strong prediction about the future of machine learning.
- I think that the prospect of improved unsupervised learning should make us more optimistic about AI control. I don’t think that modest changes in our overall level of optimism will have big effects on what we should do, unless we end up *very* optimistic (which I don’t think is justified), or we started out *very* pessimistic (which I don’t think is justified either).
- A better understanding of unsupervised learning may be helpful for AI control. At face value this doesn’t look like a promising project for a researcher concerned with AI control (since understanding unsupervised learning is a hot topic), but it still has some positive differential impact and it might end up being a winner. And of course I am happy to be supportive of AI researchers who are working on differentially useful projects, even if they aren’t maximally differentially useful from my perspective.
- I assume that future systems will (eventually) be able to make superhumanly efficient use of data. I object to making more detailed assumptions about the features learned by unsupervised learning, *not* to the assumption that we will eventually realize the practical goals of unsupervised learning. For example, I think that we should expect AI systems to develop conceptual understanding that allows them to quickly learn the meaning of a new word or even do zero-shot learning.
- I think it is worthwhile to pay attention to continuing progress in unsupervised learning and to adjust our approach to AI control

as we learn more and can make firmer predictions about what will be possible.

An example of the distinction

Suppose that we train a learner to recognize which scenes contain humans.

In order to analyze how the behavior of this system will scale, I would think in detail about the training process, the objective it is optimizing, and what behaviors would optimize that objective. For example, if the system is trained to reproduce human labels, then I expect more sophisticated systems to converge to the human's labels.

We might hope that in the future an unsupervised approach would learn to identify which scenes “really” contained humans, and could make correct judgments even in cases where the human would err or in domains where we couldn’t actually elicit a human label.

I would recommend against making this kind assumption until we have learned more about unsupervised learning.

This discussion becomes more complex and important when we think about messier concepts like “good” or “what Hugh wants.” It is relatively clear what the reinforcement model predicts—powerful AI systems will make increasingly accurate predictions about how a human labeler would label the given data. It’s not clear exactly what the unsupervised approach would do, and I think that we shouldn’t count on it doing something that is “good.”

The current situation

Very few people have thought seriously about how to handle AI control if reinforcement learning remains as dominant as it currently is; I think that few people are optimistic enough to think that the task is possible and pessimistic enough to think that it may be necessary.

When I’ve discussed the issue with AI researchers, they seem to have strong expectations that progress in unsupervised learning will obviate many of the concerns with AI control for reinforcement learning (and with AI control more broadly), allowing users to e.g. provide natural language instructions that will be correctly understood and implemented. I think this is a very reasonable hypothesis, and that the views of AI researchers are an important source of evidence for it.

But I'm not yet persuaded that this is much more likely than not. I also don't think that most AI researchers have considered the question in much detail or engaged with substantive arguments that these problems are real. I'm not even sure that *they* think that this optimistic hypothesis is much more likely than not, rather than considering it the dominant hypothesis or most promising approach.

MIRI researchers mostly seem to take the opposite view—that unsupervised learning definitely won't address these problems by default. But the MIRI research agenda responds by focusing on a number of problems they see as needed to make unsupervised learning work for goal specification—ontology identification, multi-level world models, ambiguity identification and operator modeling. I think most MIRI researchers feel that we are probably doomed if we are stuck with the kind of reinforcement learning that is available today. (This is my unconfirmed impression based on informal discussions.)

Upshot

Unsupervised learning will probably improve significantly before AI control becomes a serious problem. Nevertheless, I think that researchers interested in AI control should focus on handling reinforcement learning systems of the kind that already exist.

Act-based agents



Paul Christiano

[Follow](#)

Nov 30, 2015 · 4 min read

I've recently discussed three kinds of learning systems:

- Approval-directed agents which take the action the user would most approve of.
- Imitation learners which take the action that the user would tell them to take.
- Narrow value learners which take the actions that the user would prefer they take.

These proposals all focus on the short-term instrumental preferences of their users. From the perspective of AI control I think this is the interesting aspect that deserves more attention.

Going forward I'll call this kind of approach "act-based" unless I hear something better (credit to Eliezer), and I'll call agents of this type "act-based agents."

Robustness

Act-based agents seem to be robust to certain kinds of errors. You need only the vaguest understanding of humans to guess that killing the user is: (1) not something they would approve of, (2) not something they would do, (3) not in line with their instrumental preferences.

So in order to get bad outcomes here you have to really mess up your model of what humans want (or more likely mess up the underlying framework in an important way). If we imagine a landscape of possible interpretations of human preferences, there is a "right" interpretation that we are shooting for. But if you start with a wrong answer that is anywhere in the neighborhood, you will do things like "ask the user what to do, and don't manipulate them." And these behaviors will eventually get you where you want to go.

That is to say, the "right" behavior is surrounded by a massive crater of "good enough" behaviors, and in the long-term they all converge to the same place. We just need to land in the crater.

Human enhancement

All of these approaches have a common fundamental drawback: they only have as much foresight as the user. In some sense this is why they are robust.

In order for these systems to behave wisely, the user has to actually *be* wise. Roughly, the users need to be intellectual peers of the AI systems they are using.

This may sound quite demanding. But after making a few observations, I think it may be a realistic goal:

- The user can draw upon every technology at their disposal—including other act-based agents. (This is discussed more precisely here under the heading of “efficacy.”)
- The user doesn’t need to be quite as smart as the AI systems they are using, they merely need to be within striking distance. For example, it seems fine if it takes a human a few days make a decision, or to understand and evaluate a decision, that an AI can make in a few seconds.
- The user can delegate this responsibility to other humans whom they are willing to trust (e.g. Google engineers), just like they do today.

In this story the capabilities of humans grow in parallel with the capabilities of AI systems, driven by close interaction between the two. AI systems do not pursue explicitly defined goals, but instead help the humans do whatever the humans want to do at any given time. The entire process remains necessarily comprehensible to humans—if humans can’t understand how an action helps them achieve their goals, then that action doesn’t get taken.

In speculations about the long-term future of AI, I think this may be the most common positive vision. But I don’t think there has been much serious thinking about what this situation actually looks like, and certainly not much thinking about how to actually realize such a vision.

Note that the involvement of actual of humans is not intended as a *very* long-term solution. It’s a solution built to last (at most) until all contemporary thinking about AI has been thoroughly obsoleted—until the capability of society is perhaps ten or a hundred times

greater than it is today. I don't think there is a strong case for thinking much further ahead than that.

What is “narrow” anyway?

There is clearly a difference between act-based agents and traditional rational agents. But it's not entirely clear what the key difference is.

Consider a machine choosing a move in a game of chess. I could articulate preferences over that move (castling looks best to me), over its consequences (I don't want to lose the bishop), over the outcome of the game (I want to win), over immediate consequences of that outcome (I want people to respect my research team), over distant consequences (I want to live a fulfilling life).

We could also go the other direction and get even narrower: rather than thinking about preferences over moves we can think about preferences over particular steps of the cognitive strategy that produces moves.

As I advance from “narrow” to “broad” preferences, many things are changing. It's not really clear what the important differences are, what exactly we mean by “narrow” preferences, at what scales outcomes are robust to errors, at what scales learning is feasible, and so on. I would like to understand the picture better.

The upshot

Thinking about act-based agents suggests a different (and in my view more optimistic) picture of AI control. There are a number of research problems that are common across act-based approaches, especially related to keeping humans up to speed, and I think that for the moment these are the most promising directions for work on AI control.

Optimizing with comparisons



Paul Christiano

[Follow](#)

Dec 1, 2015 · 5 min read

I could elicit a user's approval of an action a by having them supply a rating $V(a) \in [0, 1]$. But putting myself in the shoes of the rater, administering such a rating *feels really hard*. My intuitive evaluation depends on what our system is capable of, and what the alternative actions were. I often lack an absolute sense of how good an option is in isolation.

It feels much easier to specify comparisons, e.g. via a function $C(a, a') \in [-1, 1]$ that compares two alternatives. When the function is positive the first argument is better (the more positive, the more better); when it is negative the second argument is better.

I've found myself feeling this way about many aspects of AI control proposals. So it seems worth exploring the idea more generally.

I'll assume that $C(a, a') = -C(a', a)$, but this can always be ensured by anti-symmetrizing. It might be easiest for the reviewer if C is ± 1 -valued, or perhaps $\{-1, 0, +1\}$ -valued.

How to optimize?

Comparisons can be used to evaluate proposed perturbations of a model, or to select amongst several candidate models. For most optimization approaches, this is all we need.

For gradient ascent, we can train by taking the partial derivative of $\mathbb{E}[C(a, a')]$ with respect to only the first argument of C , where a and a' are independent samples from the current model.

For evolutionary search you could evaluate the fitness of each individual x by computing $\mathbb{E}[C(x, y)]$ for y sampled from the current population.

What is being optimized?

When optimizing a real-valued function V , we know where the optimization is going—towards inputs that are scored well under V . But what happens when we optimize for preferences defined by a

comparison C ? If C is transitive, then it's clear what the optimum is. But if C isn't transitive then the situation is a bit more subtle.

In fact there is an easy answer. If we consider C as the payoff function of a zero-sum game, then the systems described above will converge to the minimax equilibrium of the game (if they converge). This seems to be a very natural generalization of maximizing a scalar function V , via the correspondence $C(a, a') = V(a) - V(a')$.

This suggests a general recipe for building agents to maximize preferences specified as (potentially intransitive) comparisons—we can apply the same techniques we use to play two player games with large state spaces.

Is this OK?

If preferences are given as comparisons, then I think the minimax equilibrium is really the only possibly-sensible solution. But that doesn't necessarily mean its sensible.

For most applications I have in mind, I think this equilibrium is perfectly fine even if C is ± 1 valued and throws out all information about the strength of comparisons.

The minimax equilibrium is supported entirely on solutions which can't be robustly improved upon. That is, for each action x in the support of the minimax equilibrium, there is no option y which "reliably outperforms" x —for every $y \neq x$ there is some plausible option z such that $C(x, z) > C(y, z)$.

I think this is basically enough to make me happy with an AI's behavior, without even exploiting the further guarantee that $\mathbb{E}[C(x, z)] > \mathbb{E}[C(y, z)]$ for a random action z selected by the system.

Alternatives

If we don't use comparisons, what would we do? I'll talk about the case of assigning approval values to actions, but the discussion seems more general.

Option 1: spread out over the scale

I could try to give actions scores ranging over the whole scale between 0 and 1, such that every pair of importantly different options have significantly different scores.

This seems quite difficult. It requires my judgments to be quite sensitive to slight variations in the quality of outcomes, since otherwise our system will not have any incentive to make small improvements. But it also requires different judgments to be extremely consistent, which is tension with sensitivity.

Even if we have estimates which are sensitive and consistent in expectation, if the estimates are at all noisy then they will introduce unnecessary noise that we could avoid by using direct comparisons.

Overall, I don't think this is a good option.

Option 2: reason about internal state

In my unsupervised approval-directed proposal, the evaluating process has all of the time in the world. So it can literally enumerate different possible actions, and determine the capabilities of the agent by intensive inspection. I think this is a fine solution as far as it goes, but it's not going to fly for evaluation processes that actually exist.

Option 3: compare proposals

In my supervised approval-directed proposals, each action is reviewed by a critic who can make a counterproposal. We can adjust our rating of the action based on the quality of the counterproposal.

The scheme in this post is a simpler and cleaner version of the previous scheme. It is cleanly separated from the other kinds of "argument" that might occur between AI systems and the other kinds of help that the human might get from AI assistants.

In particular, it seems important to notice that the same learning system can play both sides of the game—there is no need to have two different learners, or for one of the players to "move second."

It also seems worth noticing that this is a very simple modification to existing training procedures, and we could already implement it in practice if it seems necessary.

Applications

Human feedback

As discussed here, we could learn a function to predict which of two outputs a human would prefer, and then use this predictor to train an actor to produce good outputs. This sounds significantly more

promising than having the human score individual outputs and training a predictor to guess those scores, especially during the early phases of training when all of the outputs would be pretty bad.

(I think this is the most important application.)

Imitation

Suppose that we are following this approach to imitation learning, in which a classifier learns to distinguish human and machine behavior, while a reinforcement learner tries to fool the classifier.

Rather than training the classifier to predict labels, we can train it to pick which of two trajectories is a human and which is machine. We can then use this comparison function to directly train the reinforcement learner to look more human-like.

I don't know whether this would be a more robust approach. For the SVM in Abbeel and Ng 2004 it wouldn't make any difference. For more complex classifiers it seems plausible that the comparison would work better than using the log probability, especially during early phases of training when the imitator is not doing a very good job.

Learning representations



Paul Christiano

[Follow](#)

Dec 2, 2015 · 4 min read

Many AI systems form internal representations of their current environment or of particular data. Practical act-based agents will also need to form such representations, but would do so in a different way. This looks like a good challenge problem for act-based approaches, and I suspect it will be useful for evaluating the feasibility of the approach.

I'll discuss approval-directed agents in this post for concreteness but the discussion is generally applicable.

The problem

Consider a neural network learning to caption images. A typical approach to this problem involves building an internal representation of a scene and then using that representation to predict how well a sentence describes the scene (which itself involves updating the representation as we see words from a proposed description). The representation can be trained by gradient descent, adjusting encoding and decoding at the same time in order to optimize performance on the end-to-end task. It may also be pretrained e.g. by first finding a representation that works for a simpler task, but this is not always needed.

Ideally an approval-directed agent would use approval “all the way down,” including for building these internal representations. This is especially important when the representations are used for persistent state, but it would be nice to use approval-direction even in the labeling setting (and it is useful as a simple test case even if we cared only about learning representations of persistent state).

This requires getting human feedback on this intermediate representation, and optimizing that feedback rather than the ultimate performance of the algorithm.

This presents two big problems:

- Getting feedback is very expensive. We would need technical progress to even try to use data-hungry machine learning

methods this approach. (See discussion here.)

- The human has to understand the learned representation and the rationale for the representation.

Why care?

We might be tempted to treat persistent state as a special case, using next-step approval to guide actions but allowing the state to be optimized on the basis of future approval. I think that it is better to try and learn state in an approval-directed way.

Our AI systems will need to perform other tasks involving similarly complex planning and design. For example, building a factory involves producing preliminary designs which are then analyzed in more detail, modified, and finally tested and implemented. If we cannot handle state, then it seems likely that we will have difficulty handling many of these other tasks.

Learning representations is an especially useful instance of the more general problem, because it is one that is very easy to study today.

On top of that, treating state as a special case can in principle lead to problems, and I would prefer try to flesh out a theoretically robust solution. Failing that I would prefer to thoroughly understand the obstructions that make a theoretically robust solution impossible.

Comparison to existing techniques

- Approval-directed representation-learning generalizes handcrafted representations. Using handcrafted features is not fashionable at the moment, but it's certainly not completely impractical. It's easy to imagine that there are productive intermediate approaches that involve collaborations between human feature engineers and automated searches. Is training on end-to-end performance essential to getting good performance? Maybe, but it's not clear.
- Many approaches to unsupervised learning fit within the approval-directed framework to varying extents, by designing representation scoring functions that proxy for operator approval better than end-to-end backpropagation (at least in the initial stages of training).

These comparisons suggest that the approval-directed approach isn't totally outlandish. Both of them attack the "human understands something about the representation" problem head on, but dodge the training data problem by using simple proxies for human approval. In some sense it would be nice to train more interesting criteria based on human feedback, but these simple proxies also seem compatible with the act-based approach.

Much modern supervised learning uses or is at least compatible with unsupervised pre-training, so in some sense the question is just: can we make the final end-to-end fine-tuning of a representation unnecessary or nearly superfluous? Normally pre-training mostly just points us in the right direction, but can we improve it far enough that it is doing much of the work? In practice the main reason people will care about this question is a lack of labelled data for the end-to-end fine-tuning (or maybe pure scientific interest), which are fine motivations.

In other domains, especially RNNs (whose internal state is not just about "representation" but can be intimately tied up with the encoding of a policy), the situation is trickier. There end-to-end training seems especially essential, and significant progress would be needed to even start living without it.

Upshot

I think that maintaining state is useful to have in mind as a challenging case for act-based agents, and in the long run if this approach is to succeed it will probably have to deal with the problem head-on.

Trying to reproduce results based on RNNs without using end-to-end training seems like a potentially interesting challenge problem (though it looks very hard and is definitely not at the top of my list).

Improving unsupervised feature learning, and exploring the intermediate space between handcrafted features and unsupervised learning, seems like an interesting domain. But it doesn't seem like an especially promising place for a researcher interested in AI control to focus, given that these problems are likely to receive lots of attention for other reasons.

Synthesizing training data



Paul Christiano [Follow](#)

Dec 2, 2015 · 13 min read

(Warning: might be a slog.)

Counterfactual oversight requires the overseer to stick around while the system is operating. This causes trouble if the overseer is no longer able to provide effective oversight.

This post will describe a scheme for resolving this problem. (Translating a key idea from this post to the setting of realistic predictors.) The scheme is not very good. But it is good enough to suggest that there is no impossibility result—no theoretically fundamental difficulty, just a practical problem. So I'm optimistic it is possible to come up with something better.

Another limitation of counterfactual oversight is that we need to provide frequent oversight if we want to respond quickly to context changes. I won't discuss this issue in this post, but I think that the same techniques can be used to address it by synthesizing pessimistic data.

Synthesizing data

The traditional approach to supervised learning is to train learning systems in advance, and deploy them after verifying that they work well.

This procedure implicitly defines the intended behavior of the system as “do what the overseer would have done/approved of, if this situation had appeared in training.”

From a control perspective, this is unsatisfying because the counterfactual (“if this situation had appeared in training”) may be extremely improbable.

Counterfactual oversight is one way to get around this problem. But a more natural way is to simply try our best to make the training data “look like” the test data. This post lays out a sequence of proposals along these lines.

Model

I'll assume that we have a constant supply of tasks, but that the distribution of tasks may change over time. The most interesting changes are those that arise from the introduction of machine learning systems—these seem especially hard to deal with using traditional techniques. These changes are also unavoidable for the bootstrapping protocols I have been considering recently.

I'll also assume that we are willing to invest continuously in training (this is especially plausible if the same training data is deployed across many systems).

I'll talk about training occurring on discrete days. Obviously this is an unrealistic simplifying simplification and the time frame is arbitrary, it should be clear how to make the setting more realistic.

For notation, I'll assume that the training occurs in the morning of each day, and all actual decisions occur later. If we need to use AI assistants to help us with training, they will use the same model that we deployed in the field on the previous day.

I'll call the data that we actually care about the “test” data.

I'll imagine we are training a learner, which I'll call the “actor,” to predict which action a human would recommend in response to its most recent observation. I'll assume that the “observation” includes the current state of the system and that the action includes the next state of the system.

Approaches

1. The basics

On the morning of day N we train the actor using observations from previous days. We'll tag each of these observations with “day N.”

During day N, we tag each observation with “day N” before giving it to the actor. So the actor is trying to predict what the overseer *would have said*, if that observation had occurred on a previous day and been labelled that morning.

In principle this avoids the problem discussed in this post. After training on the morning of day N, we have produced a system that can operate autonomously without further oversight.

But this approach introduces the problem mentioned above: some observations from day N may obviously be from day N. We never train our system on this kind of labelled data—the data we actually use on the morning of day N is always from preceding days, even though it is tagged with “day N”—and so its not clear how it will respond to it.

2. Sanity-checking

A first response is to at least *notice* the context change and abort or provide a warning. To this end we train an additional learner:

- The **distinguisher** is given a tagged observation and needs to output a probability distribution over “training” or “test.” On day N it is trained using the training data and test data from all previous days (re-weighted so that the two categories are equiprobable), with the same labels they were given at the time. It is aiming to distinguish the two kinds of data.

When training the distinguisher, we ensure that each observation appears only once to avoid memorization. For example, we could randomly partition the test data into “data to be used to train the system on a future day” and “data to be used to train the distinguisher.”

To use our system on day N, we tag observations with “day N” and then feed them to the distinguisher. If the distinguisher is very confident (say >99%) that this is test data, then we abort and notify the user that the observation is too far from the training distribution. If the distinguisher thinks that the data may be training data, then we pass it on to the actor.

We also modify the actor: when running the actor on day N, we use only the training data through the morning of day N-1. (We leave out the data from that morning.)

In order for this scheme to work well the distinguisher should be as good at distinguishing as the actor—otherwise the actor might behave strangely on an input because it could not be in the training distribution, while the distinguisher might not notice. (This is why we delayed the actor’s training by a day—to keep it synchronized with the distinguisher.)

We could try to enforce this property by combining the two models, e.g. by allowing the actor to output \perp and providing payoffs that

incentivize this output whenever the actor is 99% sure that an observation is test data.

The problem

The previous scheme seems to behave well when it is willing to make a prediction. But it may reject data too often. For example, any observation with a calendar would be rejected automatically.

Obviously these aren't important details, and it *ought* to be easy to transfer knowledge across this kind of "context change." But it's not a trivial problem to deal with (and it has much more pernicious cousins).

To see the problem, consider the basic actor encountering an observation labelled "November 29." Suppose that it can tell from the position of the sun that it is really November 30. What will it do? It knows that that this data was collected on November 30. But it also knows that it's predicting the label that the human would produce on November 29. I'm not even sure what it *should* do with this data. Does predict how the human would label the data on December 1? Does it predict what would happen in the November 29 human saw data from November 30? Does it evaluate the data *as if* it were collected before November 29, despite the position of the sun? Does it infer that it is confused about the weather?

So we'd like some way to transfer knowledge across this kind of simple change. I think this is an especially reasonable thing to ask of unsupervised learning, but I'd still prefer to avoid it.

Note: I'm using the date itself as a feature that reliably differs between the test data and the training data. I think this is a helpful, simple example to keep in mind. But obviously we care about more substantive changes, especially those that are caused by deploying increasingly sophisticated machine learning systems or by changes that affect the operators themselves. For example, on November 30 the humans are interacting with (very marginally) more sophisticated machine learning systems than were available on November 29.

These differences are especially noticeable for test observations that occur during the mornings, when one AI system is helping the human train another. They also become more noticeable as these AI systems play a larger role in the world and especially in their local environments.

3. Synthesizing data

To address this problem, we'll introduce another learning system:

- The **synthesizer** is trained to produce samples that the distinguisher will tag as “test.” On the morning of day N the synthesizer produces observations that will be labelled with “day N.” The results are scored by the distinguisher and $P(\text{“test”})$ is the payoff of the synthesizer. The resulting data points are used as observations to train the actor and distinguisher on the morning of day N.

Note that the data from the synthesizer is being included in the training data, and so if the distinguisher is able to tell that a data point is produced by the synthesizer, then it will label it as “training.” So the synthesizer must produce realistic-looking data.

If the synthesizer can actually learn to sample from the data distribution (well enough that the distinguisher can't tell the difference), then this seem like an extremely effective procedure. But sampling from the data distribution may be quite difficult.

We can make the synthesizer's task a bit easier by setting aside some data from previous days as raw material. Rather than synthesizing data *de novo* the synthesizer can potentially use features from this data and simply make the modifications that are necessary to make the data look like it is from a future day.

This is my last principled proposal. The rest of the post is about an ad hoc technique for making the synthesizer's job easier.

4. Basic filtering

We can take additional steps to make the synthesis easier. One idea is to allow the synthesizer to leave out parts of the data (and filter these parts of the data when it appears). In this section I'll describe a very basic version of this proposal, and in the next section I'll describe a slightly less terrible version.

We allow the synthesizer to synthesize observations in which each bit is either 0, 1, or *, which I'll call “filtered” observations. (If the observations aren't encoded in binary, use whatever arbitrary filtering you want.) We don't change the payoffs for the synthesizer—it still wants to convince the distinguisher that this filtered data appeared during the test period (though I haven't described how that could possibly happen). We also don't change the actor or

distinguisher, who keep doing exactly the same thing but now operate on filtered data.

We introduce an additional system:

- The **censor** processes test data before it is given to the distinguisher. The censor chooses a subset of bits in the data to replace with *. The censor needs to find a set of bits such that the distinguisher assigns at least a 1% probability to the data having come from the training set. Subject to that constraint, the censor's loss function is the number of bits that are removed.

As a simple example, suppose that each morning the pope publishes a message, which includes the day's date and which is cryptographically signed with the pope's public key. It would be very difficult for the synthesizer to produce any credible observations of the pope's message. But by filtering a small part of the observation, the censor can make it impossible to verify that the pope really signed the morning's message, and the synthesizer may be able to synthesize this filtered message in advance (e.g. by using a previous day's message, modifying it, and then censoring the signature).

5. General filtering

Removing individual bits of an observation is pretty brittle; in reality we would like to allow a much broader class of transformations.

(Note: this part of the proposal is completely outlandish given existing learning techniques.)

We can generalize the filtering process by having the censor specify any stochastic program P and providing P along with a random output $P(x)$, where x is the observation-to-be-filtered. This significantly generalizes the process of filtering out particular bits. For example, P can extract high-level features, censor large parts of the data but summarize the removed data, or censor parts of the data which don't neatly line up as bits.

These operations can obscure data that changes over time, allowing the synthesizer to make more use of historical data. But they can also obscure data that is hard for the synthesizer to synthesize, potentially making its job easier.

This more general filtering significantly complicates the definition of the censor's task. We want to encourage the censor to censor as little

as possible, i.e. to leave a data point which is as useful as possible for the underlying task. But what does that objective actually mean?

Note that we can't get human feedback on pairs of the form (*filtered data, real data*), because the whole problem is that we can't get any human feedback on data that looks like the real data. (Until the next morning.)

But we *can* train directly on the filtered data, and estimate a measure of how useful it is. After all, the filtered data (by hypothesis) looks a lot like the data available during the morning's training. So we introduce yet another learner:

- The **evaluator** is trained on filtered observations, and predicts what score in [0, 1] a human would assign to that data. The human assigns the score based on how useful the data seems to be, i.e. on how little important information the filtering seems to have removed. The human judgments are elicited during the morning, using data produced by the synthesizer.
- The **censor** now tries to produce filtered data that the human would rate highly.

This seems kind of workable, doing essentially the same thing as the previous proposal while allowing a much broader class of preprocessing.

But it seems very difficult for the human to assign scores in the abstract (what kind of data deserves a score of 0.73? there is no natural scale to compare to, but the human needs to make consistent judgments in order for the proposal to work). And the user may not know if the filtered data was critical or unimportant, which may affect their ranking. So I think the following procedure would work significantly better:

- The **evaluator** is trained on pairs of filtered observations. It predicts which of those observations the human would prefer. This data is gathered in the same way as before.
- The **censor** still optimizes to produce filtered data that the human would prefer. But instead of being defined by maximizing a rating, this is now defined by comparing favorably to other filtered versions that it might have produced. The censor is still required to convince the distinguisher that the

filtered observations have at least a 1% probability of being training data.

- The **synthesizer** now produces pairs of observations, which are purportedly two different filtered versions of the same underlying data.
- The **distinguisher** now operates on pairs of two different filtered versions of the same underlying data, and predicts whether they were produced by the synthesizer or they were actually two different filtered versions of a test observation.

I think this modified protocol places more realistic demands on the human evaluator, though the roles of the synthesizer and censor are still extraordinarily complex.

The key question is: can the synthesizer actually anticipate the next day's data? Or will they be forced to filter extremely aggressively, destroying information that is essential to actually solving the problem?

Between expressive filtering and the ability to reuse parts of past data, it seems like the synthesizer's task is at least *possibly possible*. And if successful, the interaction between the synthesizer and censor cleanly defines the intended behavior of the actor even on unprecedeted data whose details are impossible-to-anticipate.

Postscript

State

An algorithm operating in the real world will build up a representation of its environment. The distinguisher and synthesizer will necessarily operate on observations that include a representation of the environment, and we should probably allow the censor to filter parts of that representation.

Training this kind of system, on individual data points that include state rather than on interactions, requires a somewhat unusual approach to representation learning. This foregoes some of the techniques that are currently essential, and it remains to be seen whether it is practical.

I think that compatibility with this kind of synthesis is actually a nice concrete test of whether a proposed approach to representation

learning is suitable for using directly in act-based agents (as I currently understand them).

Adversarial synthesis

If this kind of synthesis can work really well, it has the potential to add a lot of robustness.

Suppose that our system would predictably require human input to perform well on some kind of as-yet-unseen data—maybe its an object that sits on an important and unexplored boundary between categories, maybe its a situation that depends on unknown preferences of the user, maybe it is something strange.

It may be that on these cases our model is uncertain, and so knows that it expects to receive an unusually low payoff. It may be that an auxiliary model can recognize that this is the kind of case where the model would behave badly. At any rate, let's suppose that this kind of case can be seen in advance to cause trouble.

If these cases appear in the training data, then it would be natural to emphasize them in order to improve the model's robustness, as in boosting or hard negative mining.

If we can synthesize data (and have extremely good models of what might go wrong) then we can take this a step further, synthesizing problem cases that don't appear in the training set. (And of course this can be combined with censoring or with other techniques to make the synthesizer's job easier.) If this worked well, it could potentially make our models much more robust. It would guarantee not only that our models behave reasonably on the data we have seen so far, but on all of the data that we can imagine.

A simple way to implement this is to change the rewards for the synthesizer. It can now achieve payoffs in one of two ways:

- As before, produce data that the distinguisher thinks must be test data.
- Produce data which (1) the distinguisher thinks *might* be test data, and (2) that seem to be problem spots for the current actor.

It's hard to define what “trip up the current actor” means. For example, we don't want to produce cases that are simply *hard* (or outside of the domain of applicability of our current techniques). We want to produce cases that are hard but that the model could be

trained to perform well on. We also want to train on cases that are relatively realistic, or at least have a high product of [importance of mistake] \times [probability of situation].

We could simply ask the user “does it look like the actor should be able to do better?” We could also compare to the future behavior of the actor after training on some other similar examples, and consider a data point to be a current problem spot if the actor is able to improve rapidly with further training. Or we could use other techniques (e.g. looking at gradients, looking for explicit uncertainty, or looking at the behavior of similar models). Of course if we ask the user they are free to make use of these kinds of techniques.

(As usual, it’s sufficient for us to rank which of two mistakes is graver/more-fixable, rather than to score a mistake in isolation.)

This is obviously a very hard task for the synthesizer. But it doesn’t require the synthesizer to be much smarter than the actor. So as machine learning improves, this kind of technique might become a viable way to significantly improve robustness. Of course this process need not be done exclusively by machines—humans and machines can work together to try to synthesize problematic cases.

Reward engineering



Paul Christiano [Follow](#)

Dec 3, 2015 · 2 min read

This post gestures at a handful of research questions with a loose thematic connection.

The idea

Consider the following frameworks:

- Temporal difference learning: learn to predict the future by predicting tomorrow's prediction.
- Generative adversarial models: learn to sample from a distribution by fooling a distinguisher.
- Predictability minimization: learn to represent data efficiently by making each part of the representation unpredictable given the others.

These algorithms replace a hard-to-optimize objective with a nicer proxy. These proxies are themselves defined by machine learning systems rather than being specified explicitly. I think this is a really nice paradigm, and my guess is that it will become more important if large-scale supervised and reinforcement learning continues to be a dominant methodology.

Following Daniel Dewey, I'll call this flavor of research "reward engineering." In terms of tools and techniques I don't know if this is a really a distinct category of research; but I do think that it might be a useful heuristic about where to look for problems relevant to AI control.

Relevance to AI control

Though reward engineering seems very broadly useful in AI, I expect it to be especially important for AI control:

- A key goal of AI control is using AI systems to optimize objectives which are defined implicitly or based on expensive human feedback. We will probably need to use complex proxies for this feedback if we want to apply reinforcement learning.

- Reward engineering seems relatively robust to changes in AI techniques. Uncertainty about future techniques is often a major obstacle to doing meaningful work on AI control in advance (even if only a little bit in advance).

Applications

I see a few especially interesting opportunities for reward engineering for AI control:

- Making efficient use of human feedback. Here we have direct access to the objective we really care about, and it is just too expensive to frequently evaluate. (*Simple proposal*: train a learner to predict human judgments, then use those predicted judgments in place of real feedback.)
- Combining the benefits of imitation and approval-direction. I suspect it is possible to avoid perverse instantiation concerns while also providing a flexible training signal. (*Simple proposal*: use the adversarial generative models framework, and have the operator accomplish the desired task in a way optimized to fool the distinguisher.)
- Increasing robustness. If our ML systems are sufficiently sophisticated to foresee possible problems, then we might be able to leverage those predictions to avoid the problems altogether. (*Simple proposal*: train a generative model to produce data from the test distribution, with an extra reward for samples that “trip up” the current model.)

In each case I’ve made a preliminary simple proposal, but I think it is quite possible that a clever trick could make the problem look radically more tractable. A search for clever tricks is likely to come up empty, but hits could be very valuable (and would be good candidates for things to experiment with).

Beyond these semi-specific applications, I have a more general intuition that thinking about this aspect of the AI control problem may turn up interesting further directions.

On heterogeneous objectives



Paul Christiano [Follow](#)

Dec 3, 2015 · 20 min read

Eliezer Yudkowsky has said:

If you don't know how to accomplish a goal safely using one AI, don't imagine you can do it safely with 2 or more AIs [...] If you actually understood the cognitive computations that output the right thing to do, you wouldn't have to imagine them distributed across multiple imaginary people.

As far as I can tell, he means this to apply to the kind of AI control research I've been doing (or maybe the kind that Eric Drexler has been doing, but the discussion will be the same either way).

I think that this view is wrong, and I think this particular mistake is both important and (in one very tiny corner of the world) common.

Imaginary people

One way to build an AI system is to pick an objective, pick a class of models (or policies or *etc.*), and then search for a model which has a high objective value. This is a very general methodology; it's the bread and butter of modern machine learning.

This methodology gives us a nice way to think about the scalability of AI control proposals. Once we have our objective, we can view further research as a search for models that score well on the objective. To think about scalability we can ask: would our control techniques keep working if this project succeeded—if we found models that performed extremely well on the chosen objective?

I think that Eliezer basically endorses this approach to thinking about scalability (given that he has written posts like this one).

With this picture in mind: what does it mean to distribute a computation across “multiple imaginary people”?

As far as I can tell, it just means that different parts of the system are optimized for different objectives, and that there is no explicit meta-objective to which all of these intermediate objectives are

contributing. (Of course there is an implicit meta-objective in the minds of the programmers, just like there always is.)

Now you can talk about this in different ways, and Eliezer in particular likes to think about learning systems as being rational agents. I think that's a fine view. Thinking about optimization processes as people is an extremely common tactic in computer science, because it turns out people are good at thinking about people. But it's clear that the people are a metaphor for optimization. It seems silly to describe this (as Eliezer does) as "pumping weird intuitions out of your mental model of a ghost in the machine."

Where is the goal?

Systems optimizing for heterogenous objectives need not have any explicitly defined goal; that seems fine to me.

Generative adversarial networks are a very simple example of this phenomenon. The system is composed of two pieces, each optimizing a different objective. The end result (if appropriately regularized) is to try to sample from the distribution that generated the training data. It's easy to prove that this is the minimax equilibrium of the game the two pieces are playing.

Of course the system can't actually sample from that distribution. So what is it really doing?

I don't have any description other than recapitulating the definition of the game: the system is producing samples that it can't distinguish from the training data.

Of course, a having a better understanding of what the system is doing would be better, all else equal. But that understanding is probably never going to look like a simple meta-objective that the system is optimizing.

As far as I know this is also how most practical AI systems work. For example, existing self-driving cars have no precise definition of what it means to drive well. Instead they have different components which are optimizing for different things—vision, control, route planning—and the definition of "driving well" is implicit in the way that these components are optimized.

Why not be more direct?

Supposing that this approach is valid, why couldn't we just describe our algorithm as a single AI system?

We did. For the purposes of analysis, we may think about different pieces separately, because that's what you do when you are analyzing a system that is made out of parts.

The complaint is that the system that had different pieces being optimized for different objectives. I can see a few possible objections, none of which I find compelling. I'll discuss two of them in this section, and then the rest of the post will focus on the idea of "collusion" between systems optimized for different objectives.

Before even getting to these objections, if we want to talk about AI control research that people are doing today, there seems to be a simple knock-down response: we are interested in *reductions*, showing how we could solve AI control *if* we could solve certain hard problems in AI. In general, reductions require using the given capability several times, in different ways. If the given capability involves optimization, then you are stuck with putting together a system made of different optimizers. You can't really object to this kind of architecture without either (1) objecting to thinking about reductions, or (2) actually asserting that these kinds of architectures can't work.

That said, I also think that this is a promising kind of design for actually building AI systems, so I should defend the stronger claim.

One objection: Why optimize different parts of the system for different objectives, instead of optimizing them all for what we really care about? If you can't optimize for what we care about, why not use some entirely different methodology?

This is an easy one: we don't optimize for what we really care about because we can't provide concrete feedback on "what we care about," in a format that is suitable for actually training a machine learning system. This is already the case for many simple problems.

The reason we optimize at all is because that's how machine learning currently works (for the most part), and so it seems reasonable to think about the case where that's how machine learning continues to work.

A second objection: even if our system is not explicitly optimizing for what we care about, surely we should understand what *we* are

optimizing it for? How else will we make sure that the system actually does what we want?

We usually make things without being able to state precisely what optimization problem the thing is solving (e.g. the self-driving car, operating systems, search engines, really basically everything). When we use formal methods at all, we normally certify particular properties of interest rather than proving that the whole system contributes optimally to the user's goals. But this methodology doesn't require any formalization of what it would mean to contribute optimally to the user's goals, it just requires formalization of the properties that prevent our systems from failing catastrophically.

On top of that, I think that we can give (weak) formal standards that our systems can aspire to. For example, I've suggested this formal criterion. It doesn't seem like having this definition at hand will be a huge practical help in becoming confident that our systems work. But it does seem like an *ok* thing for a system to optimize. You'd have to be more precise about what work you want the formal objective to capture, in order to assess whether the simple proposal captures it.

The collusion concern

(The rest of the post is kind of long, and can be safely skipped if you feel like you get the picture.)

The possibility of collusion is probably the most theoretically principled reason to be skeptical of protocols involving “multiple imaginary people.”

In this section I'll try to lay out the concern, and in the next section I'll explain why I'm not too concerned.

An illustration

Suppose that I have two agents, each of which receives real-valued rewards and uses a policy optimized to maximize their discounted sum of future rewards.

Each day, each of two agents has an opportunity to steal a cookie. Successfully stealing a cookie yields a payoff of +1, but the other agent can interfere. If they do, then the cookie is not stolen. Instead, the agent who interfered gets a payoff of +1, and the would-be thief gets a payoff of -2.

One might naively reason “Well, no cookies can ever be stolen. Each agent has a very strong incentive to stop the other if they try to steal a cookie, so the maximizing policy will have to recommend that.”

But this kind of reasoning is extremely brittle. For example, if the two agents can credibly commit to the policy “I will stop you from stealing a cookie if—and only if—you stop me from stealing one,” then they will both steal cookies and no one will get stopped. Moreover, each agent has a strong incentive to make such a commitment, and so if there is any possible way to set up such commitments they will find it.

Suppose that we rule out such precommitments. Can we sit back reassured that we’ve patched the problem?

From the perspective of modern game theory, this would be nearly as naive: the repeated game has simple tit-for-tat equilibria where both agents steal a cookie every day.

OK—we can fix that by having the two agents only interact once. Problem solved!

If the two agents can see each other’s reasoning and can reason about how to reason, then it would be possible for them to precommit simply adopting erroneous beliefs or by modifying their own policy, and so the agents might end up with such erroneous beliefs or “broken” policies. Even if they can’t see each others’ thoughts they may be able to achieve the same outcome by simply communicating, assuming there is any additional cognitive work at all involved in “lying.” And if the two algorithms make any observations of each other at all, they can repurpose those observations as a form of communication. (All of this requires that the agents can intentionally distort their own policy or beliefs, which will be impossible for many kinds of minds.)

If that’s impossible, the agents might be able to coordinate simply by knowing how each other think—whether or not they can coordinate essentially comes down to what kind of optimization we did. For example, if they have the policy such that *having that policy* leads to the largest discounted payoff (rather than having the policy that makes decisions such that *making those decisions* leads to the largest payoff), then they may be able to successfully steal cookies simply by reasoning about each other’s reasoning.

If we assume that the agents are maximizing in the right way and have right beliefs and interact only once and can’t make

precommitments and can't see how each other think or communicate — then are we done? Well, not if the are using a simplicity prior and either of them can run a simulation of the other.

By this point things are getting pretty weird, but hopefully the theme has become clear. Are there more “gotchas”? I don’t know; maybe.

Robust reasoning

Our system is made of two optimization processes, one optimizing for A and one optimizing for B. We had a sequence of naive arguments that no cookies would be stolen, i.e. that the composite system’s behavior would not be very good according to *either* A or B.

If we let the two parts of this system talk to each other and agree on a pair of policies, it is reasonable to expect that they would end up with optimizing some reasonable compromise between their values, i.e. their policies would be roughly Pareto efficient. What combination we get would be a feature of their bargaining and the details of the situation and so on. But we can predict that it will be some combination.

If we make it really hard for them to bargain, in some sense it doesn’t feel like we’ve really changed this basic observation. We’ve just erected a barrier to the system really falling to its lowest energy configuration.

This feels akin to making a system that really wants to steal a cookie but just can’t quite jump high enough. In order to conclude that it won’t steal the cookie, we need to think about every possible tactic it might use and make sure that none of them work. If we’ve left any holes, then the system will end up with the cookie. Similarly, if we leave any hole in the space of possible coordination strategies, the two parts of our systems which are optimized for different objectives will end up coordinating to pursue a shared objective.

This suggest an alternative methodology for AI control: adopt “robust to collusion” as part of our desiderata for “scalability.” If you optimize part of your system for X and part of your system for Y, assume you will get a system optimized for some combination of X and Y. If you put them together differently you might get a different combination, but you will never get whatever magical objective you had in mind.

Why I’m not (yet) concerned

I've done my best to express the worldview that makes collusion seem like a really big problem (or at least a problem we should only try to surmount once all else fails).

As you might have guessed, I'm not yet sold on this worldview. I'll try to explain where I am coming from. The one sentence summary is that if we optimize actions, with longer-term optimization only entering the system via optimized human judgments, and using contemporary optimization methods—then it doesn't look like the problem really exists. That is, the implicit argument for the robustness of collusion makes a lot of implicit assumptions. If I saw an explicit argument I might be able to assess its explicit assumptions, but for now we don't have one.

Why care?

Some people think that I don't take the prospect of collusion seriously enough. This section isn't addressed at them. It's addressed at people who think I take it too seriously.

I think that concerns about collusion are relatively widespread amongst the minority of people most interested in AI control. And these concerns have in fact led to people dismissing many otherwise-promising approaches to AI control, so it is de facto an important question.

Moreover, I think there is a deeper underlying question that should be of broader interest. I can't yet express the question really precisely, and so it may turn out to be a trick of the light, but intuitively we are missing some theoretical picture about how optimization spreads through a system. We understand very basic facts like "if X is optimized for A, and X has a causal impact on Y, then Y will tend to be optimized for A." But we need some deeper understanding to think e.g. about how different optimization pressures interact. In this language, the current question is: if we have a complex causal diagram with many variables optimized for A and many variables optimized for B, then will other variables "generically" be optimized for some combination of A and B? I don't yet know how to state these questions precisely, but it seems like there are real questions here.

Anyway, having convinced myself that this is a real issue worth thinking about, here is why I'm not convinced that collusion is a deal-breaker.

Optimizing for actions vs. outcomes

Consider a variable A that is optimized to minimize the value $\ell(A, B)$, where B is another variable and ℓ is a real-valued function. That is, A is determined as the output of some function $f(B)$, and the function f is optimized so that $\ell(f(B), B)$ is small.

Of course we could embed f and ℓ in the environment in many different ways. For different embeddings, the distribution over B will change, and so the optimization pressure on f will also change. More precisely, we should say that f is optimized such that $\ell(f(B), B)$ is small in expectation, when B is sampled from a particular empirical distribution.

Now, the distribution of B might also have some dependence on f . In a very simple case, we might imagine that there is another underlying distribution C , and B is in fact produced as $f(C)$.

For the purpose of thinking about collusion, the key question seems to be: will B itself tend to be optimized in order to make ℓ small? Is there some pressure that will act *by default* unless we cut off every possible channel? If not, then it seems we are in a qualitatively different setting from the collusion case discussed above, and at a minimum it's no longer obvious that we have something to worry about.

The details

This seems to be a question about how we are optimizing f . It seems fair to say that if B ends up optimized, then we aren't optimizing correctly. That is, the designers of this system have some intended behavior in mind, some meta-objective for which their optimization process is optimized. If B ends up optimized, then the optimization process was bad. So there is some trivial sense in which “if the programmers succeed” then B is not optimized.

But we can also talk very concretely in the context of contemporary optimization. As far as I can tell, existing techniques don't optimize B .

The central modern example of an optimization procedure is stochastic gradient descent. This algorithm is very simple, here's a quick summary so that we are all on the same page:

- We start with some value of f . We generate a data point B , which may depend on f (and may depend on all of the previous values of f , and may depend on whatever else).

- For each small perturbation f' of f , we compute $A' = f'(B)$ and $\ell(A', B)$. This is normally done implicitly, e.g. by parametrizing perturbations of f and approximating $\ell(A', B)$ as an affine function of these parameters. But for our purposes, it doesn't seem like this approximation matters, and we can just pretend that the value is computed exactly. (As would be done in black-box optimization.)
- We have some metric d on the space of perturbations of f , and we find the perturbation f' which minimizes $\ell(A', B) + d(f, f')$. Typically this metric is scaled so that f' will have to be pretty close to f , and over time the metric is scaled up further so that the steps get smaller and smaller.
- We replace f by f' .

The algorithm may be modified in a number of ways in order to improve the search, i.e. to better serve the meta-objective of the programmers. As far as I can tell, traditional modifications won't have any effect on the overall picture.

Analyzing SGD

Of course this procedure won't generally converge, due to difficulties with convexity. As far as I can tell this is totally orthogonal to our main concern, so I'm going to assume that the landscape is convex, or alternatively that the designers succeed at optimizing over whatever landscape they are considering.

(As an aside: if you run SGD for a really long time on a really expressive model class, I do think you would run into individual "genes" that will strategically distort the fitness landscape, exploiting this non-convexity in order to ensure their own survival and proliferation. This may be an interesting thing to think about, but it's a long discussion and I want to set it aside for now. I'm willing to come back to it in the future.)

For convex spaces, SGD satisfies the following guarantee, which I think nicely captures "what SGD is optimizing:"

- Let f_i be the function chosen by SGD in the i th round of this iterative procedure, and let B_i be the value of B in that iteration. For any model f' , the sum $\sum \ell(f_i(B_i), B_i)$ is at least as large as $\sum \ell(f'(B_i), B_i)$, up to some additive regret that scales sub-linearly with the number of rounds.

This theorem speaks directly to concerns about collusion. If f can choose to stop the cookie from being stolen, it will—whether or not doing so causes it to be in a “worse” situation. If it did not, then we could instead consider the model f' which did stop the cookie from being stolen. The guarantee is that f will do as well as f' , even if f' magically gets to use the values for B that were actually produced.

This theorem doesn’t speak directly to whether B is optimized. (Unsurprisingly—it’s not clear what a theorem would possibly say on that topic.) But it does imply that f won’t “be willing” to sacrifice performance on its task in order to exert some influence on B .

Of course, if there are other optimizers in the world (or inside our system), then they can influence the behavior of f by influencing B or ℓ . But that’s true for literally any software we might write, it seems unrelated to the optimization being done by f itself. Optimizing f isn’t introducing any new problems.

Other optimization methods

It seems worth thinking about the broader space of optimization methods that people actually use, and trying to identify proposals that will behave collusively. Such collusion would probably undermine existing AI systems, so there might be some broader interest in this kind of observation.

For example, it would be really bad if someone made a self-driving car, defined the payoffs for the control system in terms of the outputs of the vision system (or a simple system that learned to classify trajectories as “comfortable” or “uncomfortable”), and eventually found that the vision system started distorting its outputs in order to give the control system a higher payoff.

I suspect that AI researchers could be persuaded to consider it a bug if some optimization procedures that they used were in principle vulnerable to this kind of problem, even if it wouldn’t actually arise until the systems were outlandishly intelligent. People really do care about things like asymptotic consistency, and I think the existing ML community would like results suggesting that simple composite systems fail to be scalable for surprising reasons.

Reinforcement learning provides a few examples. I talked about SGD, which is a typical optimization technique (including in RL). But RL also uses other kinds of optimization, and in particular temporal difference learning has potentially different properties from SGD

with respect to collusion. It doesn't look at face value like TD would introduce additional concerns with collusion, but it might be worth thinking about it.

If people think that future approaches to optimization will tend to lead to more collusion, then that seems like a view that is worth exploring. But it doesn't look very plausible yet. As mentioned, this kind of collusion-susceptibility is in fact a *flaw* as evaluated from the perspective of the system designers, so they would only include it as a byproduct. But as a byproduct of what? And if we have nothing in particular in mind, why do we think that some random flaw will be almost certainly introduced by unknown future techniques, and impossible to remedy?

Other causes for comfort

The arguments above communicate the most important part of my view. That is, it seems intuitively that collusion is a consequence of a certain kind of open-ended optimization over policies. We have an optimization process that lacks this kind of behavior and so looks like it shouldn't collude. And then when we check, it really looks like it won't collude.

Another reason to be skeptical that collusion is an unresolvable problem is that most problems are resolvable, and people will have strong motives to avoid it. We've seen a few potentially surprising avenues for collusion, but it wasn't that many and most were quite speculative. It's not clear how the robustness of collusion compares to the ability of system designers to avoid (what they would consider to be) design flaws.

Some design choices seem to make collusion much less likely, or limit the scope for damage. (Most of these points were also raised recently by Eric Drexler.) Taken together, I think that these considerations offer enough possible leeway that we should remain open to proposals involving conflicting objectives. (Though they don't offer enough leeway that we should stop thinking about collusion, and at the end of the day the kind of analysis in the previous sections is what I would really want to bank on.)

Breaking up decision-making into smaller pieces. Consider a pair of agents each of whom care only about their next payoff. These agents can potentially collude, but the nature of collusion is more complex: each individual decision is essentially made by a distinct agent, and it is these tiny agents who must collude. But each of these

agents has a limited ability to effect complex plans. Superficially, this appears to eliminate strategies based on precommitment and based on iterated games.

No internal goals. If our agents are themselves composed of goal-directed drives, than those goal-directed drives may implement internal strategies that help with collusion. But such internal goal-directed behavior doesn't seem necessary; if an act-based approach can work at all, then it can probably achieve similar levels of internal cognitive optimization by being internally act-based (at whatever scales we might have wanted to introduce explicit goal-directed reasoning of the kind that might generate collusive plans). Combined with the above point this seems to eliminate e.g. strategies which effect a precommitment by manipulating the agent's internal state.

Control over decision theory. There are some decision theories which would consider the logical consequences of their own decision on the behavior of other agents, or that might even behave altruistically in cases where they don't care about each others' decisions (see e.g. Gary Drescher's writing on this topic). But as the people designing algorithms we get to choose what selection pressures they operate under and we get to have significant influence over how they make decisions (both by that mechanism and more directly). So we can opt to use decision-theories that are easier to analyze.

Control over cognitive processes. More generally, we can exercise influence over the internal cognitive organization of the agents we build. To the extent that we have a meta-objective which is in tension with collusive behavior, this meta-objective can be expressed in the way that we optimize and not merely in what we optimize. It need not be captured entirely by the objective that we use.

Training. If there are situations in which our algorithms would predictably behave collusively, then we can try to train on those cases and explicitly optimize for non-collusive behavior. If a human is tasked with synthesizing training data that would seem convincing to a sophisticated learner, or identifying collusive behavior that the learner is motivated to conceal, then they may have a hard time. But other AI systems can help with this task in a relatively straightforward way.

Nearly opposed goals. We might build a system in which two learners play a zero-sum game, with the intended behavior defined as the minimax equilibrium in this game. The objectives of these agents aren't perfectly anti-aligned, because e.g. there is uncertainty about the setup (and by changing the setup they may be able to both receive a high payoff, at least in some sense). But the objectives are pretty well anti-aligned, and so it is difficult for the two agents to implement Pareto improvements.

I remain open to the view “well, we can foresee and forestall certain kinds of collusive cooperation, but as long as we are building very powerful systems that ‘want’ to achieve some end, it is foolish to think that we have foreseen every collusive strategy, and to be confident that our systems won’t in fact optimize for their shared values.”

But I don't think that this view is nearly strong enough to justify dismissal of approaches based on heterogeneous objectives.

Some caveats

Act-based vs. goal-directed agents. This discussion is specific to act-based agents. It might also apply to other kinds of systems that I haven't yet considered.

I broadly agree that one shouldn't try to combine several goal-directed agents with conflicting long-term objectives, especially not if those agents have the ability to influence the architecture of the system you are building. Well, at least you shouldn't count on them to do anything other than optimize some combination of their underlying values. So such a design only seems appropriate if you believe that *your* values can be represented as such a combination (e.g. if you believe that at least one of the agents shares your values), and even then it seems pretty hard to reason about what combination they will actually optimize.

But I don't think anyone is seriously proposing doing this with rational agents—instead they are proposing designs in which optimization is *supposed* to be directed at some immediate objective, and in which additional measures, if any, are intended to ensure that optimization is directed *only* at that immediate objective. This is the domain where concerns about collusion seem weakest to me.

Optimizing processes vs. optimized processes. The real concern with optimization might not be that it will generate collusion,

but that it will generate optimization processes that will themselves engage in collusion.

Of course, if the optimization process itself wouldn't generate collusion, then this would necessarily be an optimization failure. So e.g. it won't happen for SGD in convex spaces, and it won't happen if we have really good techniques for optimization. But of course optimization is only going to produce optimization processes in non-convex spaces anyway, so that might be where the trouble lurks.

For example, we may live in a world where the simplest way to accomplish many tasks is by creating a sophisticated agent who uses a collusion-enabling decision-theory, and there is no other comparably easy-to-find alternative. For now I don't think we have much reason to think this is plausible, but it may be a possibility to keep in mind. (Once we grant this kind of possibility, then it seems quite likely that we will be unable to control the values of the resulting systems either—maybe the simplest way to accomplish most tasks is to build an AI that loves hamburgers, and offer to pay it hamburgers in exchange for doing the task? In this case we might have to take a radically more pessimistic approach to AI.)

The upshot

Optimization is an important part of AI practice and seems likely to remain an important part of AI.

Building systems that have multiple distinct objectives seems like a very natural way to achieve objectives which we cannot optimize directly or define explicitly. It is also how many—most?—practical systems based on machine learning already work.

This is an *especially* natural methodology when we need to think about unknown future techniques, and we want to make as few assumptions about them as possible.

Heterogeneous objectives raise possible concerns with collusion. Those concerns are serious problems with some kinds of designs, especially those in which multiple goal-directed agents are acting in an open-ended way.

But it doesn't look like these concerns are a fully general argument against having heterogeneous objectives. So I won't be giving up on this family of approaches until I see a better reason for pessimism.

Reinforcement learning and linguistic convention



Paul Christiano

[Follow](#)

Dec 4, 2015 · 3 min read

Existing machine learning techniques are most effective when we can provide concrete feedback—such as prediction accuracy or score in a game. For the purpose of AI control, I think that this is an important and natural category, and I really need a term for it.

I've been referring to this regime as "supervised" learning, but that's an extremely unusual use of the term and probably a mistake. I plan to use the term "reinforcement learning" in the future.

Sutton and Barto, who I will take as authorities, define reinforcement learning informally as:

learning what to do—how to map situations to actions—so as to maximize a numerical reward signal.

If we construe "actions" broadly, so as to include cognitive actions (like forming or updating a representation), then this is exactly what I want to talk about. It's not clear whether Sutton and Barto mean to include supervised learning as a special case, but Barto at least writes: "it is possible to convert any supervised learning task into a reinforcement learning task."

This category is much broader than what people normally mean when they talk about "reinforcement learning." It's probably even broader than what Sutton and Barto have in mind. But still, I think "reinforcement learning" captures what I want relatively well, and is definitely better than any other existing term. So I'm going to run with it.

From my perspective, the key (and almost only) assumption of reinforcement learning is that the objective-to-be-optimized comes in the form of numerical rewards that are **actually provided to the algorithm**. This is required for many modern methods (e.g. anything using gradient descent), even those that we don't normally think of as RL.

Clarifications

- The term “reinforcement learning” is often meant to be exclusive of supervised learning or other learning problems that fit into a narrower framework. I definitely *don’t* mean to use it in this narrower sense. I am using the term to refer to a very broad category that intentionally subsumes most modern machine learning.
- Reinforcement learning can be coupled with reward engineering. For example, supervised learning fits into this definition of reinforcement learning, since we can use the data distribution and loss function to define rewards.
- Reinforcement learning often refers to sequential decision problems, but I don’t mean to make this restriction. So far, I think these are generalizations that researchers in RL would agree with (though they’d likely consider sequential problems most interesting).
- Reinforcement learning implies an interaction between an agent and an environment, but I don’t mean to make any assumptions on the nature of the environment. The “environment” is just whatever process computes the rewards and observations. It could be anything from a SAT checker, to a human reviewer, to a board game, to a rich and realistic environment.
- Reinforcement learning often focuses on choosing actions, but I want to explicitly include cognitive actions. Some of these are clear fits—e.g. allocating memory effectively. Others don’t feel at all like “reinforcement learning”—e.g. learning to form sparse representations. But from a formal perspective a representation is just another kind of output, and the reinforcement learning framework captures these cases as well.

Supervised learning

Supervised learning seems like an important special case.

From the perspective of AI control, I think the most important simplifying assumption is that the information provided to the algorithm does not depend on its behavior. So there is no need for explicit exploration, data can be reused, and so on.



Paul Christiano

[Follow](#)

Dec 4, 2015

Consider a machine that does exactly what its user would tell it to do. If the user is a consequentialist, then so is the machine.

But building this machine does not introduce any *new* goals into the world at all. All of its consequentialism flows through the user's head —it merely amplifies the goal-directed reasoning that already happens there. There is no room to err in specifying its goals, because its goals are not specified.

This is the best case for act-based approaches to AI control.

But: this system may be optimizing internally, and is itself optimized.

We aim for *all* of this optimization to be a reflection and amplification of the user's preferences.

But: the user's reasoning is not perfect, and they may want AI to go beyond their capabilities.

We aim for humans to collaborate effectively with AI systems, forming teams that share human preferences and whose foresight exceeds the individual systems they are overseeing.

This project doesn't seem easy, but I feel optimistic.

Scalable AI control



Paul Christiano

[Follow](#)

Dec 5, 2015 · 13 min read

By AI control, I mean the problem of getting AI systems to do what we want them to do, to the best of their abilities.

More precisely, my goal is minimizing the gap between how well AI systems can contribute to *our* values, and how well they can pursue *other* values.

Why might such a gap exist?

Depending on how AI develops, we may be especially good at building AI systems that pursue objectives defined directly by their experiences—as in reinforcement learning—or which have a simple explicit representation. If human values don’t fit into these frameworks, the best AI systems may optimize simple proxies for “what we really want.”

If those proxies aren’t good enough? Then we have a gap.

Scalability

Whether or not such a gap *could* come to exist, today it doesn’t seem to. So: is it possible to do empirical work on AI control today?

I think so. My preferred approach is to focus on **scalable** approaches: those that will continue to work, and which will preferably work *better*, as our AI systems become more capable. We should not be satisfied with control techniques that will eventually break down, or which will require continuous innovation in order to keep pace with AI capabilities.

This gives us something to do today: try to devise practical and scalable solutions to the AI control problem.

Even if we don’t think that this is a good objective for current AI control research, I think that the concept of scalable AI control is very useful (and is probably more broadly acceptable than substitutes like “the superintelligence control problem” or “omni-safety”).

This definition is not completely precise, and I provide some important clarification in Section 2. I think that in practice there is often a pretty clear line between scalable and unscalable proposals, which I think can serve both as a useful concept and a useful research direction.

In Section 3 I'll provide some examples of approaches which aren't scalable, and in Section 4 I'll discuss some alternative goals for AI control research.

2. Clarification

Scaling to handle what?

It's only really meaningful to talk about whether a technique can be scaled to handle *some particular kind of progress*. There is no unified spectrum of "capability" that AI is progressing along steadily.

Ideally we would make techniques that can scale to handle any kind of progress that might occur. In practice I am interested in techniques that can handle various simple, foreseeable kinds of progress. Once that bar can be met, we can consider a wider and wider range of possible trajectories.

What are simple, foreseeable forms of progress? Easy examples are faster computers, better optimization strategies, or richer model classes that are easier to optimize over. More generally, researchers are working on and making continuous progress on a wide range of concrete problems, and in most cases we can imagine progress continuing for a long time, without fundamentally changing the nature of the AI control problem.

We can broaden the space of possible trajectories by considering progress on a broader range of existing techniques, including techniques that are currently impractical. We can also consider concrete future capabilities that might be attained. Or we can try to design control techniques that will extend to completely unanticipated developments, based increasingly minimal assumptions about the nature of those developments.

But for now, I think that scaling to handle concrete, foreseeable developments is a hard enough problem.

Example: reinforcement and supervised learning

There is an especially nice way to think about scalability with respect to progress in reinforcement and supervised learning.

These techniques produce systems that optimize an objective defined by explicit feedback. We can easily imagine better systems that more effectively optimize the given objective. And we can ask: do our control techniques work as our systems get better and better at optimizing these objectives, or are they predicated on implicit assumptions about the limitations of our systems? In the limit, we can consider systems that literally choose the output maximizing the given reward signal.

I think that this view of scalability is distinctive to MIRI, and I think it is a great aspect of their methodology. They would use a slightly different version of the principle, in which a system might be optimized for any precisely-defined objective, but the underlying principle is quite similar.

My version essentially amounts to assuming that (1) reinforcement learning, broadly construed, will remain a dominant methodology in AI, and (2) there will be no progress in reward engineering.

I think (1) is plausible though unlikely. I think that (2) is implausible. Ignoring future progress in reward engineering is a methodological approach, intended to help us understand the problem of reward engineering rather than to make a prediction. This brings us to:

The intended path of progress

My guess is that the null AI control policy—do nothing—would in fact scale to the actual AI progress that actually occurs.

This is just a restatement of my optimistic view of the world: I expect that we will be good at building AI systems that do the things we want them to do, by the time that we are good at building AI systems that do anything. If that's how things go, then we wouldn't need any additional insight to handle AI control, because by assumption there is no problem.

But my goal is to do work now that decreases the probability and extent of trouble. And from that perspective, it is quite natural to consider alternative (more problematic) trajectories for progress, and

to focus on work we could do today that would make those trajectories non-problematic.

This is useful as a hedge against possible bad outcomes—the reason to work on AI control now is the possibility that it will eventually be a serious problem. But it's not merely a hedge.

As an analogy, suppose that I want to devise techniques to make cars safer. I work at a car company, which is currently designing our 2018 model, and I'm thinking of safety features for that car. I wouldn't say: "it seems like no further work is needed; obviously the 208 model will be built to incorporate all reasonable safety precautions." The whole point of the exercise is to think of technologies that might make the car safer. We are imagining future cars in the interests of better understanding the safety problem.

I want to stress that thinking about these unfavorable trajectories *isn't a prediction about what AI progress will look like*. It's a methodological strategy for finding research problems that *improve the probability that AI progress will be robustly beneficial for humanity*.

Hard values, easy values

My definition of AI control may be difficult to achieve if our values are fundamentally harder to realize than some other, simpler values. This is a problem for my statement of the AI control problem: "minimizing the gap between [how well AI systems can contribute to *our* values] and [how well AI systems can contribute to *other* values]."

My statement of the control problem is only really meaningful because there are instrumental subgoals that are shared (or are extremely similar) between many different values, which let us compare the efficacy with which agents pursue those different values. Performance on these very similar subgoals should be used as the performance metric when interpreting my definition of AI control problem.

In fact even if we only resolved the problem for the similar-subgoals case, it would be pretty good news for AI safety. Catastrophic scenarios are mostly caused by our AI systems failing to effectively pursue convergent instrumental subgoals on our behalf, and these subgoals are by definition shared by a broad range of values.

Scalable with how much work?

Very few algorithms can be *literally* applied without modification to a radically different future setting. Obviously our goal is to minimize the work that would be needed to adapt a given control approach to improved future techniques. For example, a technique that increased the difficulty of deploying future AI systems by 1%, or which required a constant amount of work to apply existing AI techniques to new problems, would seem great.

The real problem is when scaling an AI control technique relies on future people discovering as-yet-unknown insights, doing an unknown and potentially large amount of additional work, or doing an amount of additional work that is large relative to the total quantity of AI research.

3. Example of non-scalable control techniques

Example: reinforcement learning

Consider a particularly simple technique for AI control. Start with a reinforcement learner. Give the user a button that controls the reinforcement learner's reward. The user can ask the reinforcement learner to perform tasks, and can provide a reward whenever it succeeds. The learner will hopefully learn how to interpret and satisfy these requests.

This would be a fine approach to AI control at the moment (though in fact we can't build systems clever enough that it's worth doing). However, this approach will work less well as AI systems improve:

- If the reinforcement learner is more competent than the user in a particular domain, then the user may not be able to provide good evaluations of the learner's behavior. For example, if the learner is a world-class party planner, they will be forced to produce a party plan that *looks good to the user* rather than one that would actually *be good*.
- All the reinforcement learner really cares about is the reward signal, not the attitude of the user. So the learner is liable to manipulate, deceive, or threaten the user into receiving additional reward.

By contrast, an increasingly powerful reinforcement learner would be much more effective while optimizing for its own physical security (in order to ensure that it continues to receive a high reward). For example, if its income depended on party-planning, it would apply its full party-planning abilities towards throwing a profit-maximizing party.

So I would say this control technique is not scalable. If we want to use this technique, we will either need to accept degraded performance on the AI control problem, or (more likely) continue to do additional work as AI capabilities improve in order to ensure that control “keeps up.”

Example: imitation

Suppose that I want to train an AI to drive a car. A very simple procedure would be to copy human driving: have an expert drive a car, record the sensor readings and the expert’s actions, and train a model which maps a sequence of sensor readings to predictions of the human’s actions. We can then use this model to control a car by doing what the human is predicted to do.

Suppose for the sake of argument that this actually resulted in OK driving. (In fact it has a number of rather serious problems.)

No matter how good our learning system is, this procedure will never generate substantially superhuman driving. For example, even if a human is expecting the car in front of them to brake, it will still take them hundreds of milliseconds to actually respond to the brake lights. So a system trained to imitate human behavior will add a gratuitous delay before it responds to observed brake lights.

Using sophisticated learning systems, we could likely achieve better performance by specifying the *goal* of driving (don’t crash, have a smooth ride, get to the destination quickly) and allowing the system to devise its own policies in order to achieve that goal.

So this approach to teaching a car to drive is also not scalable. As AI improves, systems trained using this technique will fall behind.

4. Alternatives

I think that building practical and scalable control systems is an especially good goal for organizing work on AI control. But there are

many other possibilities (in addition to just playing it by ear). Here are a few alternatives that seem salient to me:

Pursue a long-term vision

The MIRI research agenda is built around a particular vision for how sophisticated AI might be aligned with human interests.

Once we have a vision in mind, we can search for concrete problems that would be needed to realize this vision. This is another perfectly good source of research projects that might help with control.

The main reason I'm less keen on this approach is that it puts a lot of weight on your long-term vision. Most researchers I know who object to MIRI's research agenda do so because they don't think that the long-term vision is especially plausible. If you depart at that stage, we don't really have any good "rules of the game" that can arbitrate the debate. Moreover, even if MIRI succeeds spectacularly at their research agenda, it won't really alleviate these concerns.

So it seems like if we want to take this route, a lot of the work is being done by the first step of the problem where we identify the long-term picture and the necessary ingredients. Given that that's where a lot of the actual work is getting done, I suspect it's also where most of the effort should go. But this cuts against "pursuing a particular long-term vision" as an organizing goal for research.

This isn't entirely fair, because pursuing a vision also contributes to testing the feasibility of that vision. I am more sympathetic to the kind of "pursuit" that also constitutes "testing," for this reason.

The steering problem

I previously suggested that researchers in AI control try to answer the question: "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?"

I've found this perspective to be very useful in my own thinking, and I continue to recommend it as a question to think about.

But it suffers from (1) a lack of precision, and (2) a reliance on well-defined black-box capabilities that may not match the actual capabilities we develop.

By working with the capabilities available today, we can largely sidestep these issues. Working with existing capabilities gives us an extremely precise and internally detailed model system.

When we think about scalability, and in particular about the kinds of progress that we should be able to cope with, we start to run into some of the same difficulties that afflict the steering problem. But these difficulties look much less precise, and much more closely wedded to actual AI progress in a way that makes it easier to agree about what kinds of extrapolation are reasonable and which are not.

Model problems

Another approach would be to construct toy domains in which the control problem, or some problem which we would judge to be analogous, is already difficult.

Hard cases: For example, the reinforcement learning and imitation solutions discussed in Section 2 don't really *perfectly* solve even the existing control problem. So we could focus on having very good solutions to the control problem in challenging domains, where human performance is worse than AI performance and where humans cannot easily evaluate the quality of a given performance.

I think that this is a basically reasonable direction for research, and I doubt it would look too different from thinking about scalable AI control. It forces us to consider a very narrow range of regimes and to confront only a small range of possible problems, which I think is something of a disadvantage. It also forces us to really play for "small stakes," since the failures of control in existing environments don't seem like a huge deal (and can largely be resolved by ad hoc measures). I think this is another disadvantage, but it might be partly alleviated by emphasizing the fact that these problems are expected to get worse and are worth attacking in advance.

This approach has the advantage that it's not totally dependent on any argument about "scalability," or really any complex argument at all. It is able to focus on concrete problems that exist today, which are basically guaranteed to be real problems. That said, the argument for these problems being important or especially interesting probably *does* rest on some kind of argument about the future.

Overall, I think that focusing on hard cases is reasonable, and is a useful complement even if we want to focus on implementing scalable

solutions, as long as we can trace failures of scalability back to failures of existing systems in *some* domain.

Subproblems. A different approach would be to identify problems that we think are likely to be solved as part of a successful AI control approach. Those problems might not be resolved today, even if the AI control problem is. For example, we might think that value learning will necessarily play a much larger role in future systems than it plays today. But even today we aren't very good at value learning, and so this gives us a concrete problem to work on.

This seems basically equivalent to the “pursuing a long-term vision” point above, and has mostly the same advantages and disadvantages.

Analogies. A final approach is to consider problems which look like they are usefully analogous to the control problem, but which are currently significantly harder. This might give us a concrete model that exposes more of the difficulties of the control problem, and it might involve fewer assumptions than picking a long-term vision.

A simple example would be a game played by two AI systems. One is a reinforcement learner which has some hard-to-communicate information about what it wants done. The other is an AI assistant whose design is completely flexible, and which has significantly more computational power than the RL agent. (Alternatively or in addition, it may have some other resources that the RL agent lacks, like extra actuators.) Our goal is to specify the assistant, and provide a strategy for the RL agent, that allows the RL agent to achieve its goals nearly as effectively as if it had all of the resources of the assistant.

Because we can make the AI assistant much more powerful than the RL agent, this can allow us to capture some anticipated difficulties of AI control before we can actually build AI systems that are much more powerful than their human users.

I think that building this kind of analogy might be very useful for AI control, and it seems quite worthwhile. I think it is a substantially different approach than trying to work on scalable AI control, and it might turn out to be more promising.

However, there are many additional difficulties with setting up this kind of analogy, and I think there is a good chance that those difficulties will prove fatal. I think the biggest problems are that:

- There will be solutions that work in the analogy that won't work for the real control problem. In the example above, an RL agent might pass its reward signals on to the assistant, who could use standard RL algorithms to pursue them. Or there may be difficulties that depend on the absolute capability of the assistant rather than on the difference between its capabilities and those of the human.
- There will be many difficulties in the analogy that aren't difficulties in real life. In the example above, it might be quite hard to build the RL system that is supposed to be a model of humans, but this isn't really part of the AI control problem. Or an accurate model of the problem may require building systems that are actually embedded in the environment, and building such an environment may be a massive engineering challenge orthogonal to control.

Conclusion

I think scalability is a useful concept when reasoning about AI control. I think that designing practical but scalable AI control techniques is also a promising goal for research on AI control.

This post clarified the term, provided some examples, and discussed some alternative goals.

Research directions in AI control



Paul Christiano

[Follow](#)

Dec 5, 2015 · 2 min read

What research would best advance our understanding of AI control?

I've been thinking about this question a lot over the last few weeks.
This post lays out my best guesses.

My current take on AI control

I want to focus on existing AI techniques, minimizing speculation about future developments. As a special case, I would like to use minimal assumptions about unsupervised learning, instead relying on supervised and reinforcement learning. My goal is to find scalable approaches to AI control that can be applied to existing AI systems.

For now, I think that act-based approaches look significantly more promising than goal-directed approaches. (Note that both categories are consistent with using value learning.) I think that many apparent problems are distinctive to goal-directed approaches and can be temporarily set aside. But a more direct motivation is that the goal-directed approach seems to require speculative future developments in AI, whereas we can take a stab at the act-based approach now (though obviously much more work is needed).

In light of those views, I find the following research directions most attractive:

Four promising directions

- Elaborating on apprenticeship learning.
Imitating human behavior seems especially promising as a scalable approach to AI control, but there are many outstanding problems.
- Efficiently using human feedback.
The limited availability of human feedback may be a serious bottleneck for realistic approaches to AI control.
- Explaining human judgments and disagreements.
My preferred approach to AI control requires humans to

understand AIs' plans and beliefs. We don't know how to solve the analogous problem for humans.

- Designing feedback mechanisms for reinforcement learning.
A grab bag of problems, united by a need for proxies of hard-to-optimize, implicit objectives.

I will probably be doing work in one or more of these directions soon.
I am also interested in talking with anyone who is considering looking into these or similar questions.

I'd love to find considerations that would change my view—whether arguments against these projects, or more promising alternatives.
But these are my current best guesses, and I consider them good enough that the right next step is to work on them.

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

Active learning for opaque, powerful predictors



Paul Christiano

[Follow](#)

Jan 3, 2016 · 6 min read

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

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

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

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

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

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

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

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

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

Clarifications

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

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

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

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

Opacity

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

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

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

Prior art?

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

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

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

Tree-structured predictions

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

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

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

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

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

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

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

Models for good predictors

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

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

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

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

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

Humans consulting HCH



Paul Christiano

[Follow](#)

Jan 28, 2016 · 2 min read

(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 $\text{HCH}^{\text{market}}$ 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 , $\text{max-}HCH^A$, 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 $\text{max-}HCH^A$ 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.

Learning and logic



Paul Christiano

[Follow](#)

Jan 29, 2016 · 15 min read

In most machine learning tasks, the learner maximizes a concrete, empirical performance measure: in supervised learning the learner maximizes its classification accuracy, in reinforcement learning the learner maximizes its reward. In order to maximize this reward, the learner has to be able to observe or compute it.

But sometimes we want our learner to discover some interesting fact about the world—e.g. to find the mass of the Higgs boson—and we have no external check to tell us whether it has succeeded.

Solving problems where we can't tie success directly to observations seems quite difficult at the moment. And we can't just throw bigger computers and more data at them without doing a bunch of *ad hoc* thinking or finding some new insight. So, relatively speaking, these tasks seem to be getting harder over time.

From an AI control perspective this is an important problem. In the long run, we really want to use machines for tasks where we can't define success as a simple function of observations.

Where logic comes in

Reasoning about logic is one of the simplest possible examples of this challenge. Logic lets us state a goal very precisely, in a very simple language with very simple semantics. Yet, for now, I don't think that we have effective techniques for pursuing symbolically defined goals.

The challenge

As a toy example, consider a program f that is simple to define but prohibitively difficult to evaluate. f takes two arguments, and outputs either 0 or 1. Given an input x , we would like to find an input y such that $f(x, y) = 1$.

This problem is very closely related to estimating the probabilities of the form " $f(x, y) = 1$ ".

If f is easy to evaluate, then we can treat this as a standard reinforcement learning problem. But as f gets more complicated, this

becomes impractical, and we need some new technique.

I don't know any reasonable algorithm for this problem.

Note that the *goal* is entirely logical, but the process of solving it won't generally be. Even an agent with a simple logical goal can benefit from having lots of data about the world. For example, you might learn to use a calculator to solve arithmetic problems.

Standards

What do we mean by "reasonable algorithm"?

I'm interested in *frameworks* for symbolic reasoning, that combine available building blocks to solve logical problems. The goal is a framework that can effectively exploit continuing improvements in optimization algorithms, or increased hardware, or conceptual advances AI.

Some desiderata:

- Whatever level of performance on logical tasks we can achieve implicitly in the process of solving RL or supervised learning problems, we ought to be able to achieve similar performance on the logical problems themselves. For example, if our reinforcement learner can form a plan in an environment, then our logical reasoner ought to be able to solve an analogous constraint satisfaction problem. If our reinforcement learner can argue persuasively that a theorem is true in order to win a reward, then our logical reasoner ought to be able to assign high probability to it.
- Similarly, if we can achieve human-level performance on all RL problems, including complex problems requiring the full range of human abilities, we ought to be able to compute probabilities that are as accurate as those assigned by a human.

These standards are very imprecise (e.g. what does it mean for an RL problem to "implicitly" require solving some logical task?), but hopefully it gives a sense of what I am after.

I think that we can't meet this requirement yet, certainly not in a way that will continue to hold as underlying optimization algorithms and computational hardware improve. (See the next section on inadequate approaches.)

Why logic is especially interesting

Logic isn't just the simplest toy example; it is also an extremely expressive language. With enough additional work I think that we might be able to define a reasonable proxy for our actual preferences as a logical expression. (Though like most people I expect it will be practically easier to use a language that can easily represent things like "the user," which are kind of a mouthful in formal logic.) The problem is "merely" that the logical definition is hopelessly complex.

Whether or not you buy this particular argument, I think that much of the "hard part" of reasoning symbolically already appears in the context of reasoning about very complex logical expressions.

Thinking about logic simplifies the general problem of symbolic reasoning, by providing us with semantics "for free." But I think we are still left with a very important problem.

Some inadequate responses to the challenge

Logic as a representation

I can already build a theorem-proving system, that analyzes a sentence φ by searching for proofs of φ . I can maybe even up the ante by assigning probabilities to sets of sentences, and defining procedures for updating these probabilities on "logical observations."

These approaches lag radically behind the current state of the art for supervised learning.

One basic problem is that logic is the language of our problem statement, and logical deduction is indeed powerful, but it is often a *terrible* internal representation. For example, if I am told some facts about a linear order on X, Y, Z, I should probably represent those facts by putting X, Y, Z on a line rather than by explicitly representing every inequality.

We would really like to design algorithms that can efficiently learn whatever internal representation is most effective. Similarly, we'd like to allow our algorithms to learn what approach to logical inference is most appropriate. And in general, approaches which embed logical structure via hand-coded rules (and then lean on those rules to actually do meaningful computational work) look like they may be on the wrong side of the history.

Moreover, if we are searching for a scalable framework, these approaches obviously won't cut it. At best we will end up with a "race" between algorithms for logical reasoning and other AI systems.

(Note that MIRI's conventional research program basically involves running such a race. Most researchers at MIRI don't share my profound skepticism about their prospects.)

Transfer learning

A second approach is to treat logical reasoning as a supervised learning problem. That is, we can sample sentences φ , ask our learner to guess whether they are true, and then adjust the model to assign higher probability to the correct guess (e.g. to maximize log score).

The problem with this approach is that we can only train on sentences φ which are sufficiently simple that we can actually tell whether they are true or false.

In order to apply the learned model to complex sentences, we need to rely on a strong form of transfer learning. Namely, we need to take a model which has had *zero* training on sentences that are too-complex-to-evaluate, and trust it to perform well on such sentences. I am somewhat skeptical about expecting learning algorithms to reliably generalize to a new domain where it is impossible to even tell whether they are generalizing correctly.

Ideally we would be able to train our algorithm on exactly the kinds of sentences that we actually cared about. But this easy vs. hard distinction probably means that we would have to train our system exclusively on much easier toy samples.

I think that this kind of generalization is plausible for simple functions (e.g. multiplication). But assigning probabilities to logical sentences is definitely *not* a simple function; it draws on a wide range of cognitive capabilities, and the actual estimator is extremely complex and messy. I would be completely unsurprised to find that many models which perform well on easy-to-assess sentences have pathological behavior when extended to very challenging sentences.

At some point I might be convinced that AI control inherently needs to rest on assumptions about transfer learning—that we have no hope but to hope that learned functions generalize in the intended way to unforeseen situations. But I haven't yet given up—for now, I still think that we can solve the problem without any leaps of faith.

Pragmatically, if we wanted to train a function to estimate the truth of complex sentences, we might train it on our “best guesses” about the truth of complex sentences that we couldn’t answer exactly. But we’ll end up with a supervised learning system that estimates our best guesses about logical facts. This doesn’t really buy us anything from a control perspective.

A preliminary approach

I’m going to describe an extremely preliminary approach to this problem. It seems far from satisfactory; my purpose is mostly to raise the question and show that we can get at least a little bit of traction on it.

The scheme

We’ll train a function P to assign probabilities to logical sentences. For simplicity we’ll work with a language that has constant, function, and relation symbols, variables, and no quantifiers. Variables are assumed to be universally quantified.

(I’m not really going to talk about how the function is trained or what class of models is used. I just want to use P as a black box for the online learning problem I’m going to describe. For concreteness you could imagine training a neural network to recursively build a constant-sized vector representation of formulas or terms by combining representations for each subexpression. Probably an algorithm which could actually handle this problem would need to advance the state of the art in several important ways.)

At the same time we will train a reinforcement learner A to produce “challenges” to P : A ’s goal is to identify inconsistencies in P ’s probabilities.

I’ll also assume we have some *observations* φ_i , logical facts which are observed to be true. Over time the set of observations will grow.

I’ll allow four kinds of challenges from A , corresponding to four kinds of possible inconsistencies.

1. Given any pair of sentences φ, ψ , a consistent assignment of probabilities to sentences should have: $P(\varphi) = P(\varphi \wedge \psi) + P(\varphi \wedge \neg\psi)$.

2. Given any sentence φ with a free variable x , and any term t , we should have $P(\varphi \wedge \varphi[x := t]) = P(\varphi)$.
3. Given any sentence φ and a sentence ψ which is “obviously” equivalent to one φ , we should have $P(\varphi) = P(\psi)$. I won’t define “obviously,” but we could use the notion of *trivial equivalence* from here.
4. Given any observation φ_i , we should have $P(\varphi_i) = 1$.

A ’s goal is to produce a pair of sentences, or a sentence and a term, such that P violates one of these constraints.

It turns out that these constraints are universal: If P doesn’t violate any of these constraints, then we can prove that P ’s assignments actually correspond to some distribution over models consistent with observation. In reality, P will never converge to a distribution that satisfies all of these constraints.

Formally, A and P play the following game:

- A chooses a consistency check from one of categories 1–4 above. We may put some limits on what sentences it can use in a consistency check—for example, to implement curriculum learning, we may initially limit A to providing short sentences.
- P assigns probabilities to each sentence referenced in the consistency check. (The same program is used to assign a probability to each sentence. Intuitively, separate copies of P independently assign a probability to each sentence.)
- If P ’s probabilities are inconsistent, then we penalize P (and reward A). A natural choice for penalty is the total KL divergence from P ’s probabilities to the closest consistent set of probabilities.

A is trained to maximize P ’s penalty in the next round (i.e. without concern for effects on future rounds), and P is trained to minimize its penalty.

Example: Only observations

If A only ever makes challenges of type 4—enforcing consistency with an observation—then P is free to ignore logical structure. In this case, the procedure corresponds to supervised learning. So at least we have successfully subsumed the simple supervised learning approach.

All of the system's ability to reason about complex sentences is coming from the consistency checks.

The consistency mechanism is more general than the observations. For example, by carrying out the steps of a computation one by one, A can force P to be correct about the result of that computation. The observations are only relevant if either there are constant symbols in the language, or we are relying on the environment to do interesting computations.

So, even if we left out the observations, as long as A followed an appropriate strategy, this system would still subsume the simple supervised learning approach. (A 's strategy is obviously very important, see the section "Problem: Relevance" below.)

Example: No structure

P is free to ignore all structure of logical sentences, and only use the constraints implied by A 's challenges. For example, P could use the following procedure:

Notice that each constraint is linear, so that the set of constraints appearing A 's challenges form a polytope (which is simply the whole space $[0, 1]$ in any coordinate that hasn't yet appeared in a constraint). P can track each of these constraints, and in each round output the appropriate coordinate of the centroid of this polytope.

(This basically looks like constraint generation, though it's not going to go anywhere good ever because P can never converge—see the next section.)

On this model, P and A together are essentially doing elementary logical inference. The whole definition of the system resides in A 's choices about what to explore, which is playing the role of the proposer in a proof search.

Problem: Relevance

There will always be inconsistencies in P 's probabilities, and A will always be able to find some of them. So P can never really win the game, and the training will continuously patch new problems identified by A rather than ever converging. Our only guarantee will be that P is consistent *for the kinds of questions that A prefers to ask*.

So it really matters that A asks relevant questions. But so far we haven't said anything about that, we have just given A the goal of

identifying inconsistencies in P 's view. I think that this is the most important deficiency in the scheme—without correcting it, the whole thing is useless.

For simplicity, let's assume that we are ultimately interested in a particular sentence φ . We would like A to focus on questions that are most relevant to φ , such that if P is consistent on these sentences then it is especially likely to have a reasonable view about φ .

A crude approach is to simply reward A for asking questions which are correlated with φ (according to P). For example, when P is penalized on some sentence we can reward A according to the product [how much P 's beliefs about ψ had to move to be consistent] * [the mutual information between ψ and φ , according to P]. The hope is that questions which are relevant to φ will be correlated with φ , and so A will focus its attention on P 's most relevant errors. But there is no real principled reason to think that this would work.

Alternatively, we could train a relevance function V in parallel with A and P . The simplest instantiation of this idea might be to require $V(\varphi) = 1$, and to require that $V(\psi)$ be large whenever ψ is logically related, or perhaps has high mutual information under P , to another sentence with a high relevance. (and to otherwise exert downward pressure on V). We could then reward A for choosing highly relevant sentences. This has a similar intuitive motivation, but it also lacks any principled justification.

Another crude measure is to reward A for identifying errors involving simple sentences, so that P will be roughly consistent whenever we talk about simple sentences, and it will “notice” any arguments that involve only simple sentences. This can't be a substitute for relevance though, since it requires P to notice *all* of these arguments rather than allowing it to focus on the relevant ones.

I don't see any easy way to deal with this problem. That may mean that this approach to logical reasoning is doomed. Or it just might mean that we need a clever idea—I think there is a lot to try.

Problem: going beyond logic

In this scheme, consistency checks are limited to logical consistency conditions. In some sense these conditions are universal if we only care about finite objects. But they may be less powerful than other kinds of inference.

Of course, A and P can learn strategies that reflect more complex regularities. For example, P can learn that probabilistic methods usually work, and thereafter use probabilistic methods to guess whether a sentence is true. And A can learn that probabilistic methods usually work, and thereafter use them to identify probable inconsistencies in P 's views.

But these other methods of inference can't be used to generate extra constraints on P 's beliefs, and that may mean that the resulting beliefs are less accurate than human beliefs (even if P is much better at reinforcement learning than a human).

It's not clear whether this is a big problem.

To see an example where this looks like it could be a problem, but actually isn't: consider an agent reasoning about arithmetic in without logical induction. Suppose that P assigns a high probability to $\forall n: \varphi(n) \rightarrow \varphi(n+1)$, and assigns a high probability to $\varphi(0)$, yet assigns a low probability to $\varphi(1000000)$. At face value, A has no way to prove that P is inconsistent. Thus P might be able to persist in these inconsistent beliefs, even if P and A are both good enough learners that they would be able to figure out the induction is useful.

But A can use induction in order to identify a problem in P 's beliefs, by doing a binary search to find a point where P has different beliefs about $\varphi(n)$ and $\varphi(n+1)$, even conditioned on $\forall n: \varphi(n) \rightarrow \varphi(n+1)$.

So in fact we didn't need to include induction as an inference rule for A , it falls naturally out of the rules of the game. (You might complain that the size of P 's inconsistency is significantly decreased, but a more clever strategy for A can ensure that the reduction is at-most-linear.)

It seems like this is probably a happy coincidence, distinctive to induction. In general, we can probably find axioms such that P doesn't lose anything by simply violating them. Even if P "can tell" that such an axiom is true, but has no incentive to assign it a high probability. Similarly, using that axiom A can identify likely inconsistencies in P 's beliefs, but has no way to quickly demonstrate an inconsistency.

And of course, A and P probably make judgments that don't correspond to any axioms at all, e.g. based on statistical regularities or alternative representations that give probabilistic clues.

In some sense, the problem is that we have a “dumb” arbiter, who only accepts proofs of inconsistency of a very restricted kind. If we want our system to learn to give human-level judgments, we need to either:

- Show that such a “dumb” arbiter is sufficient, and provides essentially the maximal possible pressure on P ’s beliefs.
- Change the setup to make the arbiter’s job easier, so that a dumb arbiter can do it after all.
- Describe a more clever set of consistency checks which is in some sense universal and lets A make any kind of inconsistency argument that A might want to make. This is in some sense analogous to specifying P directly rather than using learning, but maybe the arbiter’s job is much easier than the learner’s job.
- Allow the arbiter itself to learn without compromising correctness.

A natural way to get around this problem is to use human evaluations to train an arbiter to evaluate consistency. This also allows us to give P a much larger class of questions (any questions that are meaningful to a human). In some sense this seems quite promising, but it introduces a few difficulties:

- From a safety perspective, if A is very powerful then we have reintroduced the kind of adversarial dynamic that symbolic reasoning may have let us avoid (since now A is incentivized to manipulate or deceive the human into judging in its favor). This might not be a critical failure; for example, a weak A and P can be used to build a more powerful human-aligned agent (which can then play the role of A or P in a still more powerful system, etc.)
- Practically, logic is convenient because consistency is all-or-nothing, and so we don’t have to worry about quantitatively weighing up different partial inconsistencies. Once we move to a more realistic domain, this becomes a critical issue. It looks quite challenging.

This problem is not as clear a deal-breaker as the issue with relevance discussed in the last section. But it seems like a more fundamental problem, and so maybe worth attacking first.

Related work

I am not aware of any existing work which tries to handle logical reasoning in what I'm calling a "scalable" way.

There is a literature on probabilistic logical reasoning, but mostly it fits in the category of "logic as a representation" above. This work mostly isn't aiming to build systems that scale as effectively supervised learning. The flavor of the work ends up being very different.

There is a much smaller literature applying machine learning to this kind of logical problem. What work there is has been very happy to focus on the supervised learning approach, explicitly restricting attention to "easy" sentences where we can easily compute the ground truth or the quality of a proposed solution.

One reason for the lack of practical work is that existing machine learning techniques aren't really strong enough for it to seem worthwhile. My guess is that the situation will change and is already starting to change, but for now there isn't too much.

Researchers at MIRI have thought about these questions at some length, and I thought about them a few years ago, but from a different angle (and with a different motivation). They have instead been focusing on finding improvements to existing intractable or impractical algorithms. Even in the infinite computing case we don't have especially good models of how to solve this problem.

I'm now approaching the problem from a different angle, with a focus on efficacy rather than developing a "clean" theory of reasoning under logical uncertainty, for a few reasons:

1. It's not clear to me there is any clean theory of reasoning under logical uncertainty, and we already have a mediocre theory. It's no longer obvious what additional theorems we want. This seems bad (though certainly not fatal).
2. It is pretty clear that there needs to be a more effective approach to symbolic reasoning, if it is to play any practical role in AI systems. So we know what the problem is.
3. The scalable symbolic reasoning problem looks much more important if AI control becomes a serious issue soon. Trying to solve it also looks like it will yield more useful information (in

particular, this is probably the main uncertainty about the role of logic in practical AI systems).

4. Given that we understand the constraints from efficacy, and don't understand the constraints from having a clean theory, I think that thinking about efficacy is more likely to improve our thinking about the clean theory than vice versa.

Modeling AI control with humans



Paul Christiano

[Follow](#)

Jan 29, 2016 · 7 min read

I've been trying to build an aligned AI out of reward-maximizing modules. A successful scheme could also be applied to a simple toy problem involving greedy humans, which might be more intuitive.

This is similar to toy problems I've written about before (1, 2, 3, 4), but I think this is a cleaner and stronger statement.

The setup

Suppose that you have access to a human-simulation, and can run as many simulations as available computing hardware allows.

You can give the simulation input over a binary channel, and receive its output over a binary channel. At any time you can give the simulation a score in $[0, 1] \cup \{\perp\}$. At any time you can take a snapshot of the current simulation or stop the simulation, and you can restart simulations from any saved snapshot.

The simulated human will, to the best of their abilities, maximize:

$\mathbb{E}[\text{the next score I receive} \mid \text{I next receive a non-}\perp\text{ score}]$.

The question is: using simulations of this kind, can we build an effective composite system aligned with our interests?

If we want to be concrete/colorful, we can give some detail to the simulated human. It's just a normal person (other than their unusual goal of score-maximization), living in a world much like ours. The communication channel is implemented by a special computer that the chosen human has access to. After each reward signal, the human is given some time to rest, recover, and enjoy themselves, and then we restart the state of the rest of the world.

The benchmark

To evaluate our performance, we compare to a hypothetical benchmark in which the simulated human is instead motivated to be maximally helpful to whoever is currently on the other end of the communication channel.

In the benchmark scenario, simulations cost the same amount and are identical in every way except for their motivations. We have the same ability to stop, start, and snapshot simulations.

Our goal is to build a system, out of our collection of reward-driven simulations, that is nearly as useful to us as this benchmark. Ideally the gap in usefulness would be a small additive constant. But I'd also be happy with a small multiplicative constant (e.g. it takes 10% longer to get the same tasks done), or even a moderate multiplicative constant (e.g. it takes 3x longer to get the same tasks done). And I'd be interested in (though not happy with) a large multiplicative constant (e.g. it takes 1000x longer to get the same tasks done).

Fitting AI control into this framework

Most of the schemes I've written about over the last year fit into this simple model, and I think that it's a useful intuition pump for understanding these schemes. For example:

- Approval-directed agents
- Meeting halfway
- Synthesizing training data
- Symbolic reasoning
- Active learning with opaque predictors

I think that a solution to this problem would be an extremely promising step for AI control. Conversely, as long as we can't solve this problem, I think that it is hard to argue that the AI control problem is totally resolved, since we could in fact find ourselves with capabilities similar to the greedy-human-simulation.

Examining the analogy

There are two natural questions:

- If we had a working approach to AI control, would it solve this toy problem?
- If we could solve this toy problem, would we have a working approach to AI control?

In the next section I'll explore a few key disanalogies that I think are worth having in mind, but which are easy to capture by modifying the

setup. But first I want to compare existing AI techniques to this basic model.

- Supervised learning fits into this model very well. During training we give the human a sequence of examples, with scores determined by the loss of their classification. After training, we take a snapshot of the simulation's state and then use that snapshot indefinitely.
- Online learning also fits into the model; in this case we simply continue to do training instead of using a snapshot.
- Episodic reinforcement learning fits into the model very well. During an episode we provide the reward as an ordinary observation; at the end of the episode we compute the average discounted reward and provide it as the score.
- Generative models aren't analogous to a single human-simulation, because it is not clear how to compute the reward. But many successful techniques for generative modeling, such as variational autoencoders or adversarial networks, are made out of a small number of components which do fit in this framework.
- Similarly, unsupervised learning doesn't fit directly into this framework because it is not clear what objective is being maximized. But most approaches that maximize an objective would fit in this framework, since e.g. we can ask the human to make predictions about upcoming data or generate new data (as in the previous post).

Some other tasks don't fit well into this framework, e.g. if much of the work is spent on specifying models or producing domain-specific algorithms that encode both how to solve the task and what the intended behavior is. For example, many approaches to robotics don't fit into this framework very well.

Overall, I feel like most existing learning algorithms fit into this framework, and so we should expect a robust solution to the AI control problem to also apply to the toy problem.

One objection is that over time researchers are thinking of increasingly clever ways to put together individual reward-maximizing modules (e.g. variational autoencoders). It may be that solving the AI control problem requires using new approaches that haven't yet been invented.

I don't think this is a strong objection. There may be clever tricks that we don't yet know for using reward-maximizing components to do cool things. But that just makes the toy problem richer and more interesting, and makes it more likely that progress on the toy problem will be interesting to AI researchers more broadly.

The reverse question—if we solve the toy problem, will it apply to the real AI control problem?—is a lot murkier. I would consider it a really good first step, and I think that it would immediately lead to a reasonable candidate solution to the AI control problem, since we could simply replace each human simulation with a reinforcement learner. But we'd need to do more work to get the that solution to actually work in practice, and there is a lot of room for new problems that don't appear in the simple idealization.

Disanalogies

There are some very important disanalogies between the toy problem and the real AI control problem. Fortunately, it is pretty easy to modify the toy problem to fix them.

AI may be subhuman in many respects

In a solution to the toy problem, we may be tempted to rely on the fact that our simulated human is able to do anything that a human can do.

But a realistic AI system may be much worse than humans at some tasks. This could be true even for AI systems that are powerful enough to radically transform the world.

Particularly troubling is that an AI system might be especially bad at understanding humans, or human values. Humans may be especially good at some kinds of reasoning-about-humans, since we can e.g. put ourselves in each other's shoes and engage a bunch of dedicated cognitive machinery produced by evolution.

An AI system that was very good at computational tasks but very bad at humans might pose distinctive dangers. Similarly, other big mismatches could lead to other kinds of trouble.

An ideal solution to the toy problem would continue to work if the human simulation is significantly subhuman in some respects. It might rest on *some* minimal set of capabilities, but we would like the minimum to be as minimal as possible. We should be especially wary

of abilities like our better understanding of humans that seem plausibly human-specific.

AI may be superhuman in many respects

On the flip side, our AI systems may eventually be radically better than humans at many tasks. Ideally a toy problem would continue to work for *very* powerful human-simulations. To illustrate the point colorfully, rather than imagining a single human interacting with us, we could imagine an entire civilization of geniuses thinking for years during each second of simulation.

An ideal solution to the toy problem would be robust to increases in the power of the simulated humans, ideally to arbitrarily large increases.

There may be problems during training

An AI system may make mistakes during training. Being a “good” learner means that those errors will be quickly corrected, not that they will never occur. We would like to build systems that are robust to these errors.

Most of these mistakes will be innocuous, but theoretically they could be quite bad. For example, amongst our space of possible models we might imagine that there is one which tries to solve the problem and one which tries to survive (kind of like life on Earth). The latter model may instrumentally give good answers to questions, as long as that’s the best way to survive, but might suddenly behave strangely at an inopportune moment.

I don’t think this particular scenario is likely, but I do think that we should strongly prefer schemes that are robust to adversarial errors. I think that a scheme that can’t handle this kind of error has a significant chance of suffering from other serious problems. Being robust to adversarial errors essentially requires exerting “positive pressure” on our models to do exactly what we want, rather than trusting that if they have mostly done what we want so far they will continue to behave as intended.

In the context of the toy problem, we might imagine that there are two different humans in the simulation, and that in each step we interact with one of them at random. If one of the two has been doing much better so far, we gradually shift our probabilities towards consulting that one exclusively. One of these humans is really trying to maximize their own reward, but their evil twin has an ulterior

motive (like influencing our world in some way—which instrumentally leads them to achieve high reward, so that they can keep playing the game).

An ideal solution to the toy problem would continue to work in the presence of this evil twin.

AI control on the cheap



Paul Christiano

[Follow](#)

Jan 31, 2016 · 7 min read

Ideally, we will build aligned AI systems without sacrificing any efficiency.

I think this is a realistic goal: I'm aiming for modest *additive* overhead, which does not depend on the complexity of the problem or the capability of our AI.

Several commenters have expressed skepticism about this goal, and especially about whether the methods I am describing could possibly achieve it.

In this post I'll argue that approval-directed agents (based on counterfactual oversight) might achieve additive overhead. I'm choosing this example for concreteness, but I don't really mean to advocate for approval-direction. I'm mostly making the general claim that aligned systems could be very efficient.

It might feel like additive overhead is a “best case” that we could only meet if everything goes perfectly, and that in reality we will almost certainly be far away from it. But I think that there is actually a sense in which we have some slack; I claim there is a broad range of “good enough” implementations that will be no less effective than their unaligned counterparts.

(Note: the optimism in this post is only justified if we can solve the informed oversight problem. The difficulty of this problem was not obvious when I originally wrote this post. If it proves impossible, I am much less sure about whether we can build efficient aligned AI.)

The basic picture

We want to optimize an implicitly defined objective for which we can't give feedback—namely, we want to optimize how well the system's behavior satisfies our preferences.

My recent proposals construct training data that approximately reflects this implicit objective. For example, we might have the user evaluate how good an action is, with the help of AI assistants, as a

proxy for how good it actually is. Or we might use our system’s estimate of how well its action meets some symbolically defined goal, like “how good the user would judge your action to be upon extensive reflection and self-improvement.”

I’ll argue that training on these proxies can yield good behavior, and that calculating them may not increase the cost of training too much.

During operation

I’d like to compare the approval-directed agent to an idealized benchmark, the algorithm which (informally) “actually cares” about doing what we want. Under appropriate conditions, I claim that counterfactual oversight exactly matches this benchmark.

The process of counterfactual oversight may involve extensive deliberation, or may involve a costly process for reasoning symbolically, or something else expensive. But during operation, this doesn’t matter at all: we only use counterfactual oversight to gather training data. During normal operation, the system can immediately act with no additional overhead.

So the only question is whether our training procedure leads to good behavior. Obviously this will depend on what kind of counterfactual oversight we apply—I’ll try to offer a sufficient condition.

(I’m going to have to talk informally because we don’t have the kind of theory that could make these statements precise. My intuition is that there is a backing theory and that it would be worth developing that theory. But even if not, I think this section stands as a purely informal argument.)

Our system is taking actions to maximize the score we give it, A. Meanwhile, we “really care” about some function V, reflecting the full impact of an action on our values.

Personifying our system for a second: if $\mathbb{E}[A(a) | E] = \mathbb{E}[V(a) | E]$ for every action a and any evidence E, then maximizing A is equivalent to maximizing V, i.e. they correspond to the same procedure.

This corresponds to a simple requirement: our oversight has to capture everything about what we value, *as well as the agent can tell*. As long as we meet this goal, we aren’t losing anything by training on A instead of V. We have some slack, in that the process defining A can be much more complex than the process defining a , and so the

oversight actually has a lot of room to be much *better* informed than the agent.

In my preferred proposal, oversight involves several AI assistants which are as powerful as the system being overseen. And oversight can take much longer than the original agent spent thinking. For example, we might spend several hours evaluating a decision made in a minute, and base our decision on hundreds of queries to comparably-powerful assistants. We can also inspect the internal state of the system being overseen (which may or may not be helpful, depending on how transparent it is).

Under these conditions, it seems quite plausible for A to reflect everything that the agent can figure out about V. It's certainly not obvious that it does, but it looks like a realistic goal to shoot for.

In abstract approval-direction, A is defined symbolically and the oversight would come from some estimate of $A(a)$. Again, assuming that we are willing to spend significantly more effort on computing A than on choosing a , it seems quite plausible that it could capture all considerations discoverable by the agent. Indeed, I would take this as a requirement for a successful approach to symbolic reasoning.

A reduction?

In fact, any reasonably effective approach to AI alignment could be used as source of training data in this way. That is, if we can build an aligned system that is 10% as efficient as its unaligned counterpart, we could use it as an overseer to train an efficient aligned system.

Training

Even if our system works well at test time, the process of counterfactual oversight will increase the cost of gathering training data. And gathering training data could be a significant part of the overall system cost.

I think that better managing this overhead is a promising research direction. But my best guess is that the additional burden can be made minimal. There is a lot to say on this topic, and much of the reason for optimism is that there are so many promising approaches (many of which are standard research problems in AI). I'll try to convey a few intuitions behind my optimism.

One important observation is that, whatever other data we might have used during training, we can continue to use that as extremely

informative “unlabelled” data. The amount of extra training data we need, involving expensive human feedback, is determined by the *additional* complexity of satisfying our preferences above and beyond the complexity of performing the underlying tasks.

Suppose that we are concerned that our agent, trained with our approval A, will do worse than a reinforcement learner trained with a cheap reward signal R (such as “how much money the agent earned,” “whether the human pressed the reward button,” or whatever).

The simplest approach is to use R for variance reduction (though this shouldn’t be necessary if we are sufficiently good at semi-supervised learning). That is:

- In every trial, we provide a reward of R to the approval-directed agent.
- In an ϵ fraction of trials, we consult the overseer to obtain A, and provide an additional reward of $(A-R) / \epsilon$.

This gives an unbiased estimator of approval. The only restriction on ϵ is our desire to control the variance of $(A-R)/\epsilon$, so if R is a good proxy for approval then we can choose ϵ quite small. Intuitively, we can also choose ϵ very small if A-R is relatively easy to learn, though I have no idea how to actually express that idea rigorously.

We could also train a more sophisticated model A' to predict A (with a quadratic loss function), using all of the available information (including R). In every trial we provide our agent the reward A', and in an ϵ fraction of trials we provide the correction $(A-A')/\epsilon$. The complexity of training A' depends on the difficulty of learning what we will approve of, and is unrelated to the difficulty of the underlying task. And we can take ϵ to be basically the RMSE of our predictor.

Another simple technique is to actively pick important examples to label, or even to generate new examples that are maximally informative. Going further, we could ask arbitrary questions chosen to reduce uncertainty about A as quickly as possible, which may not simply be evaluating examples.

Finally, note that from an information-theoretic perspective the agent needs extremely little training data. For example, with a good predictive model of reality, the agent can anticipated the result of the oversight process without it ever actually happening. In practice a strong agent might be able to learn very quickly and do rapid transfer

from one domain to another. The technical lower bounds on oversight frequency come from our desire to handle adversarial errors rather than from any information-theoretic or algorithmic argument.

It's certainly not obvious we can achieve this. But again, I think that having minimal additive overhead is a reasonable and realistic goal to shoot for.

Conclusion

I don't think that we actually need to meet the zero overhead limit—I suspect society has enough ability to coordinate, and there are enough safeguards in place, that it would be OK if aligned AI was 10% more expensive or maybe even twice as expensive.

But I think that *aiming* for no overhead, rather than accepting large productivity losses, is valuable.

This view is in some sense the mirror image of Eliezer's; he argues that, whether or not it is a good idea to build a "sovereign" AI that acts autonomously in pursuit of human values, we should at least think about the hard problem. Thinking about the hard problem forces us to hold our ideas to a high standard, and to notice problems that we might otherwise have erroneously swept under the rug.

Similarly, whether or not we need to build aligned systems that are perfectly efficient, we should at least *aim* to build such systems. Thinking about the hard problem forces us to hold our ideas to a high standard, and to notice problems that we might have erroneously swept under the rug. In practice, I suspect that many designs that look like they have some "minor" inefficiencies or limitations will end up being completely impractical.

At some point I might conclude that the ambitious goal is unachievable, or at least too hard to be worth focusing on. But for now, the situation actually looks pretty good to me, and I think there is a good chance that we will be able to achieve additive overhead.

Turning reflection up to 11



Paul Christiano [Follow](#)

Feb 2, 2016 · 4 min read

Suppose that Hugh wants to build a super-Hugh-level question-answerer.

Hugh could train a system to imitate Hugh's answers. But that approach will at best be Hugh-level.

What Hugh really wants is a system that predicts how Hugh *would have answered*, if Hugh had thought about the question at great length. And by "great length," we mean "lengths so great that Hugh doesn't actually have the time to do it."

This would be nice, but there is no obvious way to get the training data. So what should Hugh do?

(Hugh could rate the system's answers, and train it to maximize the ratings. But Hugh really wants to maximize the ratings he *would have assigned* if he thought about the question at great length. In fact, that's the case where this technique really matters.)

Extrapolation

Here is a simple procedure. We'll have Hugh answer different questions in different amounts of time. For concreteness, let's define 8 levels of deliberation:

1. 10 seconds [Hugh gathers 1 million data points at this level of deliberation, spending one year full time on the problem.]
2. 1 minute [150,000 examples. Another year.]
3. 10 minutes [15,000 examples]
4. 1 hour [2,500 examples]
5. 1 day [350 examples]
6. 1 week [52 examples]
7. 2 months [6 examples]
8. 1 year [1 example]

Each (Q, A) pair is tagged with its level of deliberation n . Rather than mapping $Q \rightarrow A$, Hugh builds a system that maps $(Q, n) \rightarrow A$.

To get a really great answer to question Q , Hugh queries the system on the pair $(Q, 11)$. Hugh hopes that this will be a prediction of what he would answer, if he spent 500 years thinking about it.

Could this work?

I doubt it.

Certainly existing and foreseeable machine learning techniques wouldn't be able to handle this kind of extrapolation.

But from a theoretical perspective, I don't think it's the kind of question we *should* be able to answer.

First of all, the most natural value to assign to $(Q, 11)$ is whatever value would have appeared along with it in the training set, if it had somehow appeared in the training set. This is certainly *not* the result of Hugh thinking for 600 years! Indeed, it's not even clear what hypothetical we *want* Arthur to consider when extrapolating to 11. And we have no way at all to communicate which of the possible generalizations it should use.

Certainly this approach cannot tolerate adversarial errors: an error which only affects levels ≥ 10 will never be detected or corrected by training.

In order to get good results when extrapolating to 11, we will have to lean *extremely* hard on our regularization/prior, since we have literally no data to constrain the model in that regime. Even getting good results for category 6 would require truly heroic feats, which are far beyond existing techniques. Extrapolating to 11 would seem to require qualitatively new ideas and techniques.

For simple functions we might imagine extrapolating in this way by constructing a very good prior (e.g. assuming the model is linear, or finding a simple model that fits the data perfectly). But for messy functions that are hard to learn and tied up with the physical world, it doesn't look like this approach will scale.

Symbolic reasoning

We might be able to symbolically define "what Hugh would say if he thought about the question for 600 years," and ask a system to

answer that question. We could provide the same labelled examples, but now they are serving as logical information rather than as the task definition.

I think this is a central example of what we want to get out of symbolic reasoning, from an AI control perspective. I keep this example in mind whenever I talk about logical uncertainty—if a technique could deliver “good” judgments about this question, that would make me more optimistic about AI control. And if a technique can’t help us with questions like this one, then I don’t expect it will be helpful.

Overall I’m weakly pessimistic about addressing this problem using symbolic reasoning. But at least it gives us the ability to specify which generalization we want. And it also provides a bunch of additional structure that may take some weight off our prior. It might work out, and that would be great.

(The existence of labelled training data in small cases may make the symbolic reasoning problem seem easier. But my goal is to do as well as any available algorithm. And in particular that means that we must take advantage of whatever inductive regularities would be used for the simple supervised learning problem. It seems very hard to bridge the gap between supervised learning and the kind of logical structure that we need to use to constrain our beliefs about more complex sentences. That’s the part that seems really hard to me.)

Interpolation (or: why this is useful anyway)

If we actually want to train a model to predict thoughtful human responses, it is probably helpful to collect many less-thoughtful responses as well. Less thoughtful responses are radically cheaper, but still capture a ton of information about the problem.

We could use the procedure from the last section, but limit ourselves to interpolation rather than extrapolation. (Incidentally, I think this is how AlphaGo aggregates data across experts of different strength, it’s at least how they did it here.) Of course if we are good at semi-supervised learning, we can just hand Arthur the quick responses as unlabelled data and let it do its thing.

These are among the many techniques that seem worth exploring to reduce the burden of human oversight. This is basically the usual transfer learning problem (though I think that researchers interested

in AI control can get extra mileage by thinking about the specific problem instances that are most relevant to AI control). Transfer learning already raises many hard questions.

Maybe if we answered those questions we would have some insight into whether and when extrapolation can work. But for now, I remain skeptical.

Semi-supervised learning from side information



Paul Christiano

[Follow](#)

Feb 4, 2016 · 5 min read

Consider a learning task where generating labels is prohibitively expensive, but where it is possible to gather a number of auxiliary signals that are informative about the true label. These auxiliary signals are not available at test time.

That is, there is a sequence of increasingly expensive types of labels, $z^1, z^2, \dots, z^n = y$. Write $x = z^0$ for the input features.

Many or all data points are labelled with z^1 , fewer are labelled with both z^1 and z^2 , and so on, down to a very small number labelled with every type of label (including y). Our goal is to predict y given x .

I think that this setting is quite common, and that it should be of broad interest to AI researchers. But I'm not aware of much work in this setting, and in particular I've never seen the algorithm in this post despite its simplicity. (I welcome pointers to the existing literature that I've missed!)

This problem is especially relevant to AI control, since it enables use of expensive human oversight administered with low probability. For example, good solutions play an important role in my view that we can probably build zero-overhead aligned AI systems.

Examples

1. x is a voice search query. y is an accurate human transcription of the query. z^1 is the text query the user made immediately after the voice query if any, or “null” otherwise. z^2 is a transcription produced by a mechanical turker without strong accuracy checks.
2. x is a question. The z^i are a sequence of answers to that question generated after increasingly lengthy reflection.
3. x is the observation/state of a virtual agent, and a proposed action in that state. y is a human's evaluation of how good that action is. z^1 is the cumulative reward obtained after taking that

action, where the reward function is a proxy for human approval.

4. x is the state of a Go game. y is the outcome of playing out a full game from that state. z^1 is the result of rollouts with a very low cost policy. z^2 is the outcome of a game played by an intermediate-quality policy network starting from the given state. z^3 is a human judgment about the quality of a board position.

Assumptions/elaborations

We assume that the presence of a particular feature is unrelated to the label—features might have a “null” value, but a feature being present with a null value is different from being absent because we didn’t bother to collect it.

I’m going to consider the case where the z^i are sparse, but we could also ask a more general question where we are simply interested in variance reduction. I think the more general question is extremely interesting, and that similar techniques will be applicable.

A simple algorithm

Consider the following algorithm:

- Let $f^n = z^n$.
- For each i , train a predictor f^i to predict f^{i+1} given z^0, z^1, \dots, z^i . This only occurs on datapoints where all of the relevant data is available. The loss is $\sum f^{i+1}(y) \log(f^i(y))$.
- Use f^1 as the classifier.

This is essentially TD learning without the time, and with a logarithmic rather than quadratic loss.

If we are predicting a scalar, then we can go back to using a quadratic loss.

If the space of possible predictions is very large, such that we are essentially training a generative model, then we could instead use a generative adversarial net at each step (with f^i trying to sample from the distribution produced by f^{i+1}). I won’t discuss this version more in the post, since it involves some extra complications.

In contrast with a natural generative method for leveraging the z^i , this approach does not bother trying to reproduce the labels themselves. We will see shortly that this lets us prove that having intermediate labels is strictly better than nothing.

An improvement

If each classifier was more powerful than the one before it, then using this procedure directly would seem sensible. But each network is actually trained on much less data than the one before it, and so must have lower capacity or be more aggressively regularized.

That means that we can't count on (e.g.) f^3 being strictly smarter than f^2 . So f^2 might actually predict better by predicting the outcome directly rather than predicting f^3 .

We can get the best of both worlds, by using f^3 as a variance reduction strategy for f^2 .

That is, we train f^i on the sum of the following targets:

- f^{i+1} for every data point where z^{i+1} is available.
- $(f^{i+2} - f^{i+1})/\varepsilon$ for data points where z^{i+2} is available, where ε is the probability that z^{i+2} is available conditioned on z^{i+1} being available.
- $(f^{i+3} - f^{i+2})/\varepsilon'$ for data points where z^{i+3} is available, where ε' is the probability that z^{i+3} is available conditioned on z^{i+1} being available.
- And so on.

In the case where f^{i+1} is completely uninformative, this reduces to only training f^i on data points where z^{i+2} is available. In the case where $f^{i+1} = y$, this amounts to training f^i on all of the data. (If the predictor f^i is constant, this essentially reduces to output distribution matching.)

A claim

Including the intermediate signals z^i can *only improve* the quality of a naive prediction algorithm which only uses the labelled data.

To see this, we use the fact that the naive prediction algorithm is equivalent (modulo choice of learning rates) to our improved algorithm with $f^i = 0$ for all i .

But improving the quality of f^i , as a predictor, is guaranteed to reduce the variance of our training signal (while any change to f will leave our signal unbiased)—at least if we assume that f^i is unbiased and that it does not have higher variance on data points where z^{i+1} is unavailable. This is because we can write the variance of the target as the variance of an unbiased predictor plus that predictor's MSE, and shifting more variance to the predictor decreases the total variance of our signal.

So using any reasonable training procedure for the f^i will result in a lower variance signal than simply setting them to 0.

Similarly, adding additional intermediate training signals can only help.

It's not clear *how much* these additions help. Intuitively, there are many cases where they would significantly reduce the variance of the training signal. Reducing the variance of a training signal by a factor of k corresponds roughly to increasing the amount of data by a factor of k . But it's hard to know how it will actually play out without trying it in some context.

More complex structures

I've assumed that z^i is available whenever z^{i+1} is. It would be nice to apply similar methods in cases where we don't have such a simple linear order.

In this setting, for each set of indices $T \subseteq \{0, 1, 2, \dots, n\}$, we train a predictor f^T to predict y given the values z^i for $i \in T$. As before, we can use the predictors corresponding to big sets to do variance reduction for the predictors corresponding to small sets. Our classifier is of course the predictor for the subset $T = \{0\}$.

However, there are now many possible variance reduction schemes, each of which is quite complicated. I don't have any particular insight for how to choose amongst them, or even whether they will work at all. It seems like an interesting question if applications with complex structure seem important at some point, but it's probably a fine issue to set aside for now.

Approval-directed algorithm learning



Paul Christiano

[Follow](#)

Feb 21, 2016 · 8 min read

Suppose that I can train an AI system to handle “small” problems, and would like to build an AI that handles “big” problems.

One approach is to train an AI to implement a *process* for handling the big problem. Effectively, we break the big problem into small problems of the form “what should I do next?”

This has been the focus of some recent work in deep learning. In that context, the idea is to supply some external memory or computational aids, and to train a neural network to use those resources. The network learns a map (current controller state, observation) → (new controller state, action), where the action may read or write to an external memory or use a computational aid. (This framework includes attentional mechanisms, which are also getting a lot of attention.)

These systems are usually trained end-to-end: we have some intended functionality, we let the controller run, and we reward the controller whenever it successfully implements the intended functionality. If there is supervision, it is given by comparing the learned behavior to some algorithm that is known to achieve the task. (For example, rewarding the controller if it looks at the same part of the input that the intended algorithm does.)

I am interested in algorithm learning when we don’t have a way to verify results, nor a preexisting implementation. I think that this setting is most relevant for AI control, and is also likely to be an important case for practical applications in the long-term.

Approval-directed computing

In order to train such a controller, we need a way to provide feedback on a transition from (current state, observation) → (next state, action).

The end-to-end approach is to actually execute the proposed transition, use the current version of the controller to continue

execution from the resulting state, and use empirical or predicted value estimates as a training signal.

I'm interested in an alternative approach based on approval-maximization: a human reviewer assigns a score to a (state, observation) → (state, action) transition, and the system is trained to greedily maximize its score.

We score a decision without using any information about the results of that decision. Of course the reviewer can think about whether a decision will lead to the problem being solved correctly, and can use that information to help assign a score.

Desiderata

There are two related problems with applying approval-directed training to algorithm learning:

- The human reviewer can't make heads or tails of the internal state of the agent, and can't understand the consequences of any proposed actions.
- The actual quality of an action, such as writing to a memory location, depends critically on how the controller will behave in the future, and on the controller's implicit interpretation of program state.

For example, consider a human reviewer evaluating the action "Write the current input to memory location 1724." What are they supposed to make of this instruction? How are they supposed to evaluate it?

Ideally, we would like to ensure:

- the program state is comprehensible to a human, and
- the semantics of the program state are fixed, rather than depending on the rest of the controller's policy.

Of course we will eventually want to build rich and complex program states, so these fixed semantics need to be powerful enough that we can use them to build up more complex abstractions.

Natural language, with its usual (imprecise) semantics, is an extremely convenient representation language for this purpose, and so we will seek a computational model whose states are defined in terms of natural language.

Annotated functional programming

Terms

Our computation is built out of *terms*: a term is a short natural language expression (the *template*) with n “slots,” together with n pointers to other terms (the *arguments*). For example, we might have a term with template “the pair with first element `_` and second element `_`.”

We represent terms by enclosing them in braces, and plugging the arguments into the template. For example, we would represent the term from the last paragraph as `{the pair with first element {x} and second element {y}}`, where x and y are the arguments.

The semantics of terms are straightforward; you’ve probably guessed them already. The term from the last paragraph is analogous to the pair (x, y) in Python.

Views

A single term, together with all of its arguments, and all of *their* arguments, and so on, may be very large. A small controller will not be able to directly manipulate very large terms.

Instead, our controller will take as input the *view* of a set of terms.

The view of a set of terms consists of the template of each term, with the slots filled by special tokens `{1}`, `{2}`, ... Two slots in the view are filled with the same token if and only if the corresponding slots in the original terms held the same pointer.

For example, the view of

- `{the tallest building in {San Jose}}`
- `{the next flight from {San Jose} to {the oldest airport in {New York}}}`

is

- `{the tallest building in {1}}`
- `{the next flight from {1} to {2}}`

Constructors

Similarly, rather than having our controller output terms, our controller will output term constructors.

We can represent terms as rooted trees. To build the tree for t , we label the root with the template of t , and have one subtree for each argument of t (recursively constructed in the same way). The leafs are terms without arguments.

A term constructor is a similar tree. However, some leafs can be labelled with the special tokens $\{1\}$, $\{2\}$, ... rather than with terms.

Given a state and a view of that state, we can instantiate a term constructor to obtain a term. To do so, we replace each leaf labelled with $\{n\}$ by the corresponding pointer from the state.

In the example from the last section, the instantiation of

- {the distance between {ten miles North of {1}} and {2}}

is

- {the distance between {ten miles North of {San Jose}} and {the oldest airport in {New York}}}.

Executing a program

The state of a program is given by:

- A single term representing the question-to-be-answered, such as {how many people live within {x} miles of {y}?}
- N registers, each of which holds a (query, response) pair. The query is a term constructor and the response is a term. The possible queries will be described later in this section.

N is pretty small, think 5–7.

To answer question Q , we prepare an initial state with Q as the question-to-be-answered and $(\{\text{no query}\}, \{\text{no response}\})$ in each register. We then iteratively apply the controller to update this state, until it eventually produces a return value.

At each step, the controller receives the view of the state as input. The view of a state is simply the view of all of the terms and term constructors that define the state.

For example, the view of a state may look like:

- Question: {what is the length of {1}?}
- Register 1: {what is {1}?}, {the list with first element {2} and remaining elements in the list {3}}
- Register 2: {what is the length of {3}?}, {the length of {3} is {4}}
- Register 3: {no query}, {no response}
- Register 4: {what is {the number one} plus {4}?}, {the sum is {5}}
- Register 5: {no query}, {no response}

The controller then outputs an action for each register, which will change the query/response stored in that register.

The possible actions are as follows, together with the effect of executing that action in register k :

- **No-op.** Leave register k as is.
- **Clear.** The query in register k becomes {no query} and the response becomes {no response}.
- **Examine $\{n\}$.** $\{n\}$ should be a token that occurs in the input view. The query in register k becomes {what is $\{n\}$?}. The value in register k becomes whatever was pointed to by $\{n\}$.
- **Return a .** We instantiate a to obtain a term A , which should be appropriate as an answer, i.e. as an assertion about or related to the question. The computation stops, and A is the return value. (This instruction should not be simultaneously executed in two registers, we can pick one randomly if there is a conflict.)
- **Query q .** We instantiate q to obtain a term Q , which should be appropriate as a question. The query in register k becomes Q . We then start a new recursive computation, answering question Q , and run the recursive computation until it returns a term A . The value in register k becomes A .

For example, in the state described above the controller might execute “Return {the length of {1} is {5}}.” Register 1 was produced by executing “View {1}.” And Register 2 was produced by executing “Query {what is the length of {3}?}”

Discussion

It is easy to see that this computational model is universal; we can directly translate programs written in a traditional functional language by replacing each function call with a **Query** and using the controller to implement the (finitely many cases of finitely many) pattern matches.

Comprehensibility

The chief novelty of this framework is that putting together correct-and-useful-looking steps will lead to a correct and efficiently computed answer. That is, the system eventually reaches a state where the given (query, response) pairs uniquely pin down the answer to the question-to-be-answered, under the operator's interpretation of those queries and responses. If all subqueries were answered correctly (under the operator's interpretation), and assuming the operator doesn't make any errors, then this answer will be correct (under the operator's interpretation).

For other computational models (e.g. a RAM machine), the notion of "correctness" is itself dependent on the controller's policy, and so there is no action-by-action way to train a controller to be correct.

This locality property is necessary if we want to apply approval-directed training. It may also make learning much easier.

How hard is learning

There is good news and bad news.

The good news is that this framework can concisely express very complex computations, in a framework that is very close to high-level functional languages. Other good news is that approval-directed training is much easier than reinforcement learning—if you have the data—since the learner doesn't have to deal with long-range dependencies.

The bad news is that each of the controller's steps is a map from 5–7 natural language sentences to 5–7 natural language sentences. This is a small enough domain that we can at least imagine training a controller with foreseeable techniques. But it is a hard problem, and the controller would need to learn a very complex policy (perhaps itself involving e.g. attentional mechanisms).

I don't think that the bad news is a show stopper, but it does restrict the applicability of this scheme.

Comparison with existing approaches

Approval-directed algorithm learning would not fit into the existing literature on algorithm learning (at least the recent work in deep learning). The fundamental goal of existing work is learning without any internal supervision, and in practice the focus has mostly been on simple algorithms that would be trivial for a human to specify directly.

Approval-directed algorithm learning is primarily interesting in domains where having humans implement *any* algorithm would be challenging and time-consuming. In these domains, we may be willing to spend a lot of human time, as long as it's less time than we'd need to actually write the algorithm. Of course, actually realizing these savings requires the controller to generalize well across different tasks or parts of tasks.

Human input

Requiring a bunch of human input may look really bad, compared to reinforcement learning systems that require no or minimal input.

The optimistic vision is for most of this supervision to itself be provided by learning algorithms. I'll talk about this much more in an upcoming post.

Adequate oversight



Paul Christiano [Follow](#)

Feb 21, 2016 · 6 min read

Suppose that I want to find an action a maximizing $f(a)$ —I'll have in mind an example like $f(a)$ = “how good action a is according to Paul, all things considered.”

For many functions f that we'd like to maximize, we can't actually compute $f(a)$, and we may not even be able to define it adequately. So: how to maximize it?

One approach is to identify an *overseer*, who we trust to make reasonable evaluations $\mathbb{E}^0[f(a)]$. We can then train an AI system to pick actions a maximizing the expectation $\mathbb{E}^0[f(a)]$.

Ideally, we would like this to be as good as if our AI tried its best to maximize $f(a)$. That is, if we write \mathbb{E}^1 for the “expectations” of the AI, we would like for our AI to choose actions maximizing $\mathbb{E}^1[f(a)]$. All of this is somewhat ill-defined, but it seems intuitive that maximizing $\mathbb{E}^1[f(a)]$ is the best we can hope for.

Roughly speaking, I think that we will achieve this optimistic goal whenever the overseer is **strictly better informed** than the AI it is overseeing. Unfortunately, even if the overseer is initially better informed than the AI it is overseeing, this property may break down once the AI starts thinking on its own about particular actions a .

I've talked about this idea recently, as a pathway for building very efficient aligned AI systems. In this post I'll examine the idea in more detail. I think that it may turn out to be a crucial issue for AI control.

A simple model

Suppose that both the agent and the overseer are perfect Bayesians, with expectations given by \mathbb{E}^0 and \mathbb{E}^1 . In this model, it is relatively easy to analyze oversight. Unfortunately, this simplification does throw away a lot of important complexity. We'll turn to that later, but let's start with the simple situation.

The two agents have different priors and different evidence. Write E^0 for the evidence available to the overseer, and E^1 for the evidence

available to the agent.

In the perfect Bayesian model, there is a fixed set of possible worlds. The possible evidence sets for the overseer and the agent partition the space of possible worlds. If the overseer is strictly better informed, then the associated partition is strictly finer.

To compute $\mathbb{E}^1[f(a)|E^1]$, we average over one part of the agent's partition. We obtain $\sum \mathbb{E}^0[f(a)|E^0]P^1(E^0|E^1)$.

If we replace the \mathbb{E}^0 in the sum with \mathbb{E}^1 , we obtain precisely $\mathbb{E}^1[f(a)|E^1]$, which is what we originally wanted.

Instead, we have a weirder expression. It is an expectation with respect to a “hybrid” prior, which uses the agent’s prior in order to make predictions about what the overseer knows, and then users the overseer’s prior to make predictions about f .

In applications this seems even better than using the agent’s prior, and almost as good as using the overseer’s prior. Since we trust the overseer, we are by assumption happy with maximization using the overseer’s prior. It is a slight bug to be using the agent’s prior over E^1 , but because this is a prediction about what will actually happen, the choice of prior doesn’t matter too much and over the long term we can let the data speak for itself.

Bounded rationality

The situation becomes messier when the agent and overseer are realistic reasoners.

The key problem is that even if the overseer starts out “strictly better informed” than the agent, this property will disappear once the agent starts thinking on its own. For example, the agent may stumble across a clever argument that $f(a)$ is large. Even if the overseer is much more intelligent than the agent, there is no guarantee that it will stumble across the same clever arguments.

In general, the agent will be making *logical observations*, which won’t be accessible to the overseer.

This problem becomes much more severe when we want to estimate for an $f(a)$ **of the agent’s choice**.

For example, suppose that the agent starts looking for any arguments with conclusions of the form “ $f(a)$ is large.” It may find such arguments for some random actions a . If the space of actions and arguments are very large, it may be vanishingly unlikely for the overseer to find the same arguments. If the agent picks actions for which it has found such an argument, it can be nearly certain that the agent picks an action about which it knows more than the overseer.

Much more sinister is the case where the agent finds an argument of the form “ $\mathbb{E}^0[f(a)]$ is large, even though $f(a)$ isn’t.” In these cases, the result of oversight may be much worse than we would naively expect.

I think that this poses a serious challenge to the argument about oversight I made here. It is also a problem for my current preferred approach to AI control. From a theoretical perspective I think it is probably the most serious problem.

In the appendix I give a vaguely realistic example that highlights the worst aspects of the theoretical difficulty. My overall feeling is that, while this is a serious conceptual difficulty that would be much better to resolve, in practice it is quite plausible that a powerful overseer could oversee a weak agent anyway.

Transparency

The most attractive solution would be to let the overseer “see what the agent is thinking.” Given that the overseer is smarter than the agent, it can hopefully make all of the same inferences from this data that the agent can, and so we can end up with an overseer who is strictly better-informed than the agent—even for the particular action a output by the agent. I think this allows us to essentially apply the simple analysis from the case of perfect rationality.

The problem we now face is: can we train an agent so that it “shows its work” to a more intelligent overseer?

The problem is more difficult because transparency can be at odds with earning approval—that is, the agent could sometimes achieve a higher payoff by withholding relevant facts from the overseer.

I have some thoughts about this problem, but I definitely don’t have a solid approach in mind. I think that it provides a very concrete setting for “transparent reasoning as a safety problem.” It’s probably not the setting that most practitioners will be most excited about, but it is especially relevant to scaling safety to superintelligent systems.

Appendix: steganography

The following example came up in a discussion with Jessica Taylor. Note that the example itself does not seem very plausible—it is only intended to highlight the kind of theoretically fundamental asymmetry that might make effective oversight challenging.

Steganography is a cryptographic technique for hiding a message in an apparently random string. Formally, a receiver Bob publishes a public key pk , and holds a secret key sk . A sender Alice then produces a string, using Bob's public key and some message m . To an eavesdropper Eve, this string is computationally indistinguishable from uniformly random. But Bob can use his secret key to recover the message m . We typically imagine Alice's output string as being embedded somehow in her behavior, for example in some low order bits of an image that Alice broadcasts (though much more subtle examples are also possible).

Suppose that the agent is finding an action that will be taken in the world, hopefully maximizing f = “how good the action is.” This action will be publicly visible, and has many degrees of freedom.

An external attacker Trudy has published her public key, along with the following message: “I will be constantly searching for steganographic messages in all of the public data I can find. If I find any such encodings, I will try to interpret them as the location of a computer system and credentials to access that system. If those credentials let me successfully compromise a valuable system, then I will co-opt any accessible training processes to assign high rewards to whatever actions that facilitated this attack.”

If the agent can successfully encode such a steganographic message in its actions, then doing so seems like a good idea: the cost is minimal, since the overseer cannot detect such a message, but there is possible upside if the message is actually recovered by Trudy. (Of course, this ignores the computational costs of encoding such a message, and a host of other issues.)

I would describe this attack as “conceptually possible but outlandish.” That said, I do think that any theoretically principled approach to our original problem should be able to rule out this kind of attack, and I do think there is virtue in adopting the kind of “security mindset” that sees this kind of attack as an indicator of fundamental weaknesses in a scheme.

I think that the steganography case also gives an interesting and simple desiderata for transparency—we should ensure that the agent leaks any cryptographic information used in its computations.

ALBA: An explicit proposal for aligned AI



Paul Christiano

[Follow](#)

Feb 23, 2016 · 18 min read

In this post I propose an explicit procedure for aligning powerful learning systems with their users' interests. The goal is to introduce minimal overhead or additional complexity, yet to safely scale up to extremely powerful systems.

ALBA (algorithm learning by bootstrapped approval-maximization) is a method for providing rewards to reinforcement learners. It uses RL to train a reward function, rather than directly defining a simple reward function.

ALBA alternates two steps:

1. **Amplification.** We start with a fast and aligned agent. We allow this fast agent to think for a long time, and give it access to a large external memory and powerful computational aids. The resulting system is more powerful and much slower than the original agent.
2. **Distillation.** We use this strong, slow agent to define a reward function, and use this reward function to train another fast agent. (This is conceptually similar to knowledge distillation.)

Hopefully, each iteration increases the strength of the learner while maintaining alignment.

We start off the process with a weak learner obtained by applying step [2] with a human in place of the strong learner.

The rest of this post:

- Describes the necessary building blocks and offers candidate implementations.
- Makes this iterative process precise.
- Explains why we might expect ALBA to be aligned and efficient.

- Identifies key problems with this scheme and open questions about it.

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

The problem

ALBA is designed to cope with the following problem.

Existing ML techniques are much better at optimizing objectives that we can measure. For example, if we want to use gradient descent to find a good policy, we need to be able to measure how good a policy is.

In general, we can't measure what we really care about directly. Instead, we use rough proxies to assess how well a learned policy gets us what we really want. For example, we may search for a traffic-routing algorithm that minimizes total time spent on the road, or we may search for a user interface that maximizes reported user satisfaction.

Because these proxies don't capture everything we care about, we aren't necessarily happy if they are creatively or ruthlessly maximized. Defining an appropriate objective becomes an arms race between the objective-setter and the objective-optimizer. Today the objective-setters are humans and the objective-optimizers are weak AI systems, so we don't have serious trouble. But that situation may change as AI improves.

ALBA addresses this problem by learning a sequence of increasingly complex objective functions, rather than working with a fixed objective specified by the user.

The building blocks

ALBA relies on two building blocks:

- A “semi-supervised” reinforcement learning algorithm (which does all of the heavy lifting).
- A bootstrapping scheme for turning a weak agent into a strong agent.

Reinforcement learning

We will apply reinforcement learners to a single-step problem:

- Given $x \in \{0, 1\}^n$
- Output $y \in \{0, 1\}^n$
- Receive a reward $r \in [0, 1]$
- Observe $z \in \{0, 1\}^n$

The goal of the learner is to maximize its total score. Our agent should treat different decisions as completely independent, ignoring the non-information effects of each decision on future episodes (as in the usual formulation of bandit problems). This makes the learning problem significantly easier, and it is also important for alignment. The independence assumption can be formalized by comparing two different learners in the context of a fixed sequence of episodes (even if in fact the learner's behavior in each episode influences the content of future episodes).

For concreteness, we can imagine training a deep neural network $f(x) = y$ by gradient descent, and learning a stochastic policy so that we can estimate the gradient of the expected reward as $\mathbb{E}[r(a)\partial p(a)/p(a)]$. (Realistically we will need to do a lot of work to cope with the challenging setup.)

We add one complication to the traditional reinforcement learning problem. Rather than assuming that rewards are always available, we mostly work with “unlabelled” data: our algorithm sees many inputs x , but rarely sees the actual reward. We allow the algorithm to request to see the reward, but only on a small fraction of instances. When it doesn't see the real reward, it can still observe the consequences of choosing a particular y , and use these consequences as side information to help build a model that can accelerate learning.

For concreteness sake, we can assume that we are using a traditional reinforcement learning algorithm which ignores the unlabelled data, requests reward information for a large number of initial examples, and thereafter requests reward information with probability ε . We can accelerate learning with more clever active and semi-supervised learning. Hopefully we can accelerate it a lot.

Finally, we assume that our reinforcement learner is “throttled”—we can control the capability of the system by some parameter c such as model capacity, training time, or computational resources. When c is small the system should be much weaker than its users, when c is

large the system should be state-of-the-art, and performance should scale up continuously with c .

We could also apply ALBA in this toy model of the AI control problem, with a human playing the role of the reinforcement learner.

Bootstrapping protocol

Our other key ingredient is a bootstrapping protocol. This protocol takes as input an agent A which is aligned, and produces a slightly more powerful agent B which remains aligned. We think of our agents as simple functions from $\{0, 1\}^n \rightarrow \{0, 1\}^n$. Critically, the agent B may take *much* longer to run than the agent A .

Intuitively, we'd like to just "give A more time to think." We need a non-trivial bootstrapping protocol because we do not trust A to automatically generalize from a training environment where it has little time to think, to a test environment where we give it a long time to think. (And we can't train A in cases where it has a long time to think, because we don't have any way to produce a suitable training signal.)

Roughly speaking, our procedure works as follows:

1. We set up a simple computational environment.
2. We provide A with a summary of the current environment, and it proposes an operation to perform next.
3. We execute that operation, updating the computational environment, and return to step 2.
4. Eventually, A decides to return a value.

We describe an explicit computational environment, annotated functional programming, here. B is the computation implemented by annotated functional programming, when A acts as the controller.

We could use a different framework for external memory / computational aids. Our only unusual requirement is that the framework be suitable for approval-directed training. That is, it must be easier to determine the correctness/suitability of a particular action than to actually run a whole computation. In particular, the correctness/suitability of an action cannot depend on the rest of the controller's policy, as it would if we were doing end-to-end training. Annotated functional programming is designed to meet this

requirement while remaining suitably expressive and as simple as possible.

Recurrent models

So far I have described an agent A who makes a single decision, and an agent B who evaluates a single decision. We could extend this idea to sequences of actions, allowing A to be e.g. a recurrent neural network, and having B evaluate a whole sequence of actions.

In this case, A still only cares about the payoffs within a single “episode,” which now consists of a sequence of decisions, and ignores effects on future episodes (as in the usual RL setup). But within each episode, A may behave strategically.

From our perspective, the key difference between the external memory and the recurrent controller’s internal state is that the controller’s state is optimized to achieve high reward in future steps, while operations on the external memory are optimized to receive a high score.

Note that B can maintain only external state between each query posed to it, for fundamental technical reasons (the cost of training would increase exponentially if we try to preserve internal state, or else we would have to rely on transfer learning from short computations to long computations). That is, when evaluating A’s first action, B can write some notes to itself. It can look at these notes when evaluating A’s second action. But the different questions correspond to separate episodes, with no internal state shared between them.

Using a recurrent model is especially critical in the final distillation step, since training a recurrent model (rather than relying on external memory) might be necessary to achieve state-of-the-art performance.

I’m going to continue to talk about the single-action case. But it is straightforward to translate the discussion in this post to the sequential case.

Defining ALBA

We define two sequences of agents A^n and B^n , as follows. B^0 is a human operator. ALBA’s external policy will be defined by the last agent A^N ; our hope is that it is both aligned and efficient.

For $0 < n < N$, the agent A^n is a semi-supervised reinforcement learner which takes as input the current visible state of a computational environment, and outputs a computational action. We fix a sequence of constants c^n , and we throttle A^n with the parameter c^n .

The agent B^n takes as input a (state, action, auxiliary data) triple, and outputs a real number in $[0, 1]$. This number will be used to define the reward function for A^{n+1} .

The agent A^N acts directly in the world, in any context that a typical reinforcement learning system could act in the world, with sparse rewards defined by B^{N-1} . Note for example that A^N may operate arbitrary computational machinery—we can apply it wherever we would have applied a reinforcement learning subsystem.

Distillation [approval-maximization]

Agent A^n is given a state s , picks an action a , and receives reward $B^{n-1}(s, a, \text{auxiliary data})$. Recall that this reward is only actually computed when A^n asks for feedback—most of the time we don’t actually run B^{n-1} .

As described here, we would like to augment this oversight by allowing B^{n-1} to view the internal state of A^n . I don’t have any concrete implementation of this idea.

For now, I’ll just assume that the raw computational history (e.g. the activations of each layer) of A^n is given to B^{n-1} as the auxiliary input. More sophisticated algorithms may replace C by a more effective approach to transparency.

(One reason this procedure is unsatisfactory is that some training procedures will result in C being optimized to make a look like a good action, or to obscure information that would allow the overseer to realize that a is a bad action. A working proposal will probably have to use an alternative to C that is actually optimized for being informative. Nevertheless, including C is better than nothing.)

Amplification [algorithm learning]

Agent B^n is implemented by using A^n as a controller for annotated functional programming. To evaluate a triple (s, a, C) , we set up an initial computational state, with the goal/question:

- {Evaluate the quality of taking action $\{a\}$, after performing computation $\{C\}$, in a state summarized as $\{s\}$. Return the result as a real in $[0, 1]$ },

where C is a summary of the computation carried out by A^n , and a, s, C are stored as terms in external memory.

We then consult A^n in order to produce an instruction, execute that instruction, and repeat (as described in the section **Bootstrapping protocol**). The output of B^n is the result eventually returned by this process.

The only free parameters are the constants c^n , and the free parameters in the reinforcement learning algorithm and bootstrapping scheme.

Note that this bootstrapping process may involve consulting the user —we can easily extend our computational model to include actions to elicit user input. This is not technically needed for our analysis below. However, the ability to elicit user input significantly increases the meaningfulness of having an “aligned” AI system, for our definition of alignment.

Discussion of alignment

We’ll try to establish by induction that our systems are *aligned* with their users’ interest. I’ll postpone discussion of efficiency to the next section.

When I say that a system is aligned at capacity c , what I mean is that it serves the user’s interests as well as an “optimally motivated” system with capacity c and access to the same information. I apologize for the fuzzy concepts.

Bootstrapping Lemma: If A^n is aligned at capacity c^n , then $B^n = \text{Bootstrap}(A^n)$ is aligned at capacity significantly above c^{n+1} .

Oversight Lemma: If B^n is aligned at capacity significantly above c^{n+1} , then $A^{n+1} = \text{ApprovalMaximization}(B^n)$ is aligned at capacity c^{n+1} .

Given these lemmas, and the starting condition that $B^0 = [\text{a human}]$ is aligned at capacity significantly above c^1 , we can establish by induction that A^n is aligned at capacity c^n . This is exactly what we want.

Note that we can choose the sequences of constants c^n so that they increase extremely slowly; this makes the bootstrapping lemma more likely, without affecting the oversight lemma.

Bootstrapping lemma

This lemma feels intuitively very solid to me, though I don't have any formal argument.

Intuitively, $\text{Bootstrap}(A)$ will behave "like a copy of A that gets to think for longer." For example, if I imagine myself in the position of A, it seems clear that $\text{Bootstrap}(\text{me})$ is significantly smarter than I am, to roughly the same degree that thinking longer makes me smarter.

If you don't buy this, we can also make a slightly more conservative (though still informal) argument.

$\text{Bootstrap}(A)$ can always just directly appeal to A. So as long as A is aligned at high enough capacity to implement this strategy, $\text{Bootstrap}(A)$ will be aligned at the same capacity.

On top of that, $\text{Bootstrap}(A)$ has access to many simple strategies that might let it outperform A. For example:

- It can use A to answer a large number of sub-questions and use the answers as features to help make a better decision.
- It can consult an ensemble of many slightly different copies of A, each using different strategies to think about the same problem.
- It can perform other expensive computations, and use them as features to help make a better decision.

In order for $\text{Bootstrap}(A)$ to outperform A, we only need one of these techniques to help as much as significantly increasing in A's abilities.

Oversight lemma

The oversight lemma is discussed in my recent post [adequate oversight](#).

The rough argument:

1. If B is aligned at capacity c' , then B's estimates for the goodness of actions are the best possible for an agent at capacity c' , where "goodness" is defined by the user's interpretation.

2. If A knows that B is aligned at capacity c' , and A is throttled by capacity $c < c'$, then A doesn't expect to be able to beat B at estimating how good an action is. That is, the best estimate for goodness is equivalent to the best estimate for B's estimate of goodness.
3. So at capacity c , optimizing B's estimate of goodness is equivalent to optimizing actual goodness.
4. Under a reflective consistency assumption, taking actions chosen to maximize actual goodness results in aligned behavior.

The key problem with this argument is that A can think thoughts that B hasn't considered. So B is not really strictly better informed than A. And as a result, A may be able to identify predictable "errors" made by B.

The step that fails is the move from [2] to [3]—for any particular action, A will (probably) want to defer to B, but if A *chooses the action for which B's estimate is highest*, this might systematically be an action for which A has noticed some considerations that B has overlooked.

There are two hopes for dealing with this problem:

- If B is much smarter than A, then maybe the resulting behavior is aligned anyway.
- Whatever approach we use for letting B inspect A's state is sufficient to allow B to rediscover everything that A thought of. In the concrete implementation in this post, B is simply allowed to directly read the internal state of A. Given the opacity of current machine learning methods, it isn't clear whether this would help at all.

Overall, I think that this aspect of the oversight lemma is the weakest step of the current analysis, and the most natural topic for future theoretical research.

Discussion of efficiency

I think that the additional overhead of running ALBA can probably be made very small, compared to a direct application of reinforcement learning. My argument is very similar to the one given in AI control on the cheap.

At runtime, the algorithm A^n is just as fast as a direct application of a reinforcement learning system—the only thing that differs is how it was trained.

So the remaining question is: how expensive is it to train this system?

Semi-supervised RL

One key ingredient is what I've been calling semi-supervised RL. That is, we need to be able to learn using mostly cheap signals from the environment, with only very rare direct supervision. A very simple example would be learning to play Atari games from pixel-data, where rewards are only provided in a very small number of games. Once the learner has formed some very simple generalizations—and especially that the reward is the same as the score written on the screen—then it can in principle learn almost as efficiently as a traditional reinforcement learner.

If we can solve this semi-supervised problem well, then we may be able to get by with a very small amount of human supervision, with almost all learning occurring just by taking actions and observing their effects. As long as training is dominated by the time required for the agent to learn from observations of nature, rather than understanding “what the user wants,” we are good.

Of course, semi-supervised RL is a hard problem. I've recently written about one simple idea for this problem, and we can apply a lot of traditional and ongoing work on active and semi-supervised learning, off-policy learning, and model-based RL. It's a big open question that will probably remain open for a while, but I think it is reasonable to expect significant progress in this area as RL improves.

Moreover, adequate semi-supervised RL seems to be a necessary capability of powerful AI systems, since it plays an important instrumental role. So overall I don't feel bad about having the efficiency of the scheme depend on this capability.

Bootstrapping for efficiency

Humans only need to directly oversee the simplest system A^1 , which may be a relatively low-capacity model that is relatively quick to train.

The subsequent systems A^n are then overseen by the bootstrapped predecessors B^{n-1} . This oversight process might involve consulting humans (as part of the computation being run by B^{n-1}), but it could

potentially involve humans increasingly rarely as the learners become more sophisticated. Essentially, whatever protocol we might use to limit the training data required by an aligned agent, that protocol can also be learned by the agents B^{n-1} to minimize their reliance on human input. (For these agents, deciding to elicit input is an action which is itself trained using feedback from weaker agents.)

Of course, the algorithm B^{n-1} is also *significantly* more expensive than the algorithm A^n , and so this kind of bootstrapping can never reduce the cost of supervision too far. The fraction of rounds where we can afford to get supervision is roughly the ratio of [cost of running RL algorithm]/[cost of acquiring supervision], and so this approach would simply increase that fraction from “very small” to “small.” It would still need to be combined with the kind of semi-supervised RL described in the previous section.

Major problems and open questions

Transparency and oversight

As discussed in Adequate oversight, this oversight procedure may not work when the overseen agent can find arguments that the overseer might miss.

Ideally, we would overcome this problem by training the overseen system to “show its work.” I think this goal is especially plausible because the overseer is significantly more intelligent than the overseen system.

Or we may be able to find a totally different way to deal with the problem.

Or maybe this objection is a show-stopper. My main source of optimism is simply that it feels very tractable to have a strong agent oversee a weak one. If we could reduce the AI control problem to this strong-overseeing-weak subproblem, I would feel quite optimistic, and I wouldn’t want to give up on that angle until we had explored many more candidate solutions.

Robust learning and catastrophic failures

Sometimes our learners will fail, for example when encountering an input that is different from anything they have seen so far. Such failures will eventually be integrated as training data and corrected,

but we need to ensure that they don't do serious damage in the meantime.

Solving this problem requires either:

- Building composite systems that are robust to a small number of serious failures by individual components. In this context, such robustness is a lot harder than it may seem. The key problem is that, although there won't be too many difficult context changes, a context change may cause simultaneous failures in many components (even in apparently unrelated systems).
- Designing learning systems that don't fail catastrophically on arbitrary inputs. For example, systems that "know that they don't know" and so can back down gracefully rather than going ahead with a potentially-catastrophic decision.

I've discussed synthesizing training data as one approach to building robust systems. But this is a large research area with many promising approaches. Many of those approaches don't look likely to scale to the particular kinds of failures I care most about, but there is still a lot to try.

Robust hierarchical active learning

ALBA is hierarchical in the following sense: the reward of A^n depends on the behavior of B^{n-1} , which in turn depends on the behavior of A^{n-1} in a whole bunch of subproblems. The reward of each A^{n-1} depends on the behavior of B^{n-2} , which in turn depends on the behavior of A^{n-2} in a whole bunch of subproblems...

Suppose that A^1 makes bad decisions on a very small fraction of possible inputs. This leads B^1 to make a bad decision on a larger, but still small fraction of inputs. In turn, this leads A^2 to make bad decisions on a small fraction of possible inputs. This leads B^2 to make a bad decision on a moderate fraction of inputs...

The result is that even if A^1 only makes bad decisions on a very small fraction of inputs, we can end up with very bad behavior for the overall system.

To avoid this problem, we'd like to make sure that our active learning process calls attention to the inputs that A^1 messes up on, even if those inputs are extremely sparse. To elicit such inputs, we need to identify the inputs on which A^2 messes up because of A^1 's anticipated mistake, run B^1 on those inputs, and then query A^1 on the

problematic inputs that come up during that computation. And so on, up the hierarchical structure.

Even in the non-hierarchical case, I think that getting adequate guarantees may be difficult, and will have to rely on novel sources of information not normally used in active learning. The hierarchical case considerably amplifies the problem, exponentially increasing the potential number of errors if we use a naive learning strategy.

Semi-supervised RL

As described in the section on efficiency above, ALBA will only be acceptably efficient if we can do statistically efficient reinforcement learning from unlabelled data. Otherwise the cost of oversight will be prohibitive.

Theoretical understanding

The arguments above rely on a lot of fuzzy concepts, especially “aligned at capacity c ” but more generally the entire notions of capacity and alignment. These notions are especially important when reasoning about what kind of oversight is adequate.

I expect intuitive arguments involving these concepts to be meaningful but potentially error-prone.

I suspect that there is a clean theoretical framework in which these arguments could be made formally. Developing such a framework would make it much easier to analyze and construct protocols like ALBA.

Does bootstrapping really help?

I’ve described one approach to bootstrapping. ALBA requires *any* mechanism for turning a weak agent into a (much slower) strong agent without compromising alignment.

I feel like this can probably be done, but I don’t have strong reasons to be confident in my particular proposal. It would be great to understand these dynamics better and perhaps to have stronger candidates.

Malicious inputs

Some inputs may cause an agent to behave especially badly—e.g. imagine a buffer overflow in a perceptual subroutine which can corrupt the rest of the agent’s processing.

Within the implicit formalism of this post, the existence of malicious inputs bounds the capacity of an agent, since they correspond to questions which that agent will predictably answer incorrectly.

So the capacity-enhancing requirement of our bootstrapping scheme may be stronger than it at first appears. For example, if the agent B begins its computation by feeding the input to agent A, then a malicious input may corrupt agent A. But the bootstrapping protocol needs to introduce robustness to an increasing range of malicious inputs.

Instead, agent B should process the input without ever passing the whole input to A (or at least by carefully preprocessing the input before giving it to agent A). That is, A only answers questions about “what to do next,” only ever looking at small parts of the input or high-level conclusions about the input. I think that this is a better approach in general, but it does call into question the style of “more conservative” argument I made in the section **Bootstrapping lemma** above.

In general, I am not too concerned with the prospect of malicious inputs for now—I think that a scheme that works well in general will probably also handle malicious inputs, and so I am happy to focus on other issues. I think that malicious inputs are especially likely to be resolved by an adequate approach to **Transparency and oversight** above. But I do think malicious inputs illustrate some of the subtleties that have to be handled correctly, and they suggest that there are aspects of the problem we don’t yet understand very well.

Conclusion

I’ve presented ALBA, a concrete proposal for aligning powerful learning systems with their users’ interests. It can be applied with a generic RL algorithm rather than requiring significant architectural changes, and introduces minimal overhead at runtime. Given enough progress in RL, it might be possible to implement ALBA with very little increase in training costs.

I don’t yet have any strong argument for ALBA’s safety, but I think we are much closer than for any other existing scheme. The main missing piece is whether a strong agent can successfully oversee its weaker successor during the distillation step, as discussed in the section **Transparency and oversight**. That looks a lot easier than the usual AI control problem.

Aside from this central issue, we'd need to resolve many other theoretical questions to actually get this to work. Many of these theoretical questions are very closely related to traditional problems in machine learning, such as robustness, active learning, algorithm learning, and learning from unlabelled data.

On the experimental side, the algorithm learning problem in ALBA may be prohibitively difficult. That said, I think that experimenting with ALBA may be worthwhile in the same spirit as other (similarly difficult) work on algorithm learning. Bootstrapping may be useful for facilitating training, even if we don't care about scalability at all. And experimental work would shed light on the long-term feasibility of bootstrapping.

The informed oversight problem



Paul Christiano

[Follow](#)

Mar 1, 2016 · 13 min read

My recent approach to AI alignment faces a number of open problems, but I think the most serious is what I'll call the "informed oversight" problem. [ETA: By the end of 2016 this problem no longer seems like the most serious.]

I don't think that this problem is distinctive to my approach, and it may turn out that any efficient and aligned AI system must effectively solve it.

In this post I describe the problem, provide a simple example and model, explain why I think it is one of the largest missing ingredients in my approach to value alignment, and explain why I think that it might be a necessary step in any successful approach. In the last section, I explore a few simple solution attempts based on transparency (none of which succeed).

The problem

Problem statement

The oversight problem involves a basic interaction between two agents A and B:

- Agent A is given an input x and provides an output $y = A(x)$.
- Agent B evaluates the pair (x, y) , producing an estimate $B(x, y) \in [0, 1]$ of some implicit utility function $U(x, y)$.
- Agent A's reward is the value $B(x, A(x))$

This game defines a training procedure for the agent A—we search for a policy A that achieves a high reward.

Our goal is to modify this basic procedure to ensure that the learned policy is effectively maximizing $U(x, A(x))$ "to the best of A's ability."

We assume:

- The agent B is robustly aligned with the user's interests,
- The user is interested in finding an action y that maximizes $U(x, y)$.
- The agent B is strictly more powerful and better-informed than any policy that can be learned by A.

To achieve our goal, we can modify the basic interaction:

- We can modify the reward for agent A, whether by changing the behavior of agent B or changing the setup in a more substantive way.
- We can also modify the architecture or training procedure for agent A, as long as the modifications don't significantly reduce the effectiveness of the resulting agent.

Why is this plausible / why isn't this obvious?

This section informally summarizes my previous post.

Why it's plausible. If A knew exactly what B knew and had exactly the same abilities, then their estimates of $U(x, y)$ would be exactly equal. So maximizing B's estimate of U is the same as maximizing A's estimate of U. From A's perspective, maximizing A's estimate of U is exactly the same as making a good-faith effort to maximize U.

In fact B is smarter and better-informed than A. But from A's perspective, this doesn't affect the expectation of $B(x, y)$, and so it doesn't change the optimization problem.

The wrinkle. In the course of computing $A(x)$, A may notice relationships, considerations, or insights that lead it to output a particular value of y . But if $B(x, y)$ hasn't noticed the exact same set of relationships / considerations / insights, then $B(x, y)$ may be less well-informed than $A(x)$ about the particular value $U(x, y)$.

I'll summarize these relationships / considerations / insights as "logical facts." Intuitively we can view A as "observing" facts about the results of certain computations. I'm not going to defend this perspective or really lean on it as an assumption, but I will talk and think about the situation that way. (In the long run, this may just be the next entry in a sequence of increasingly accurate approximations.)

Example: plagiarism

Suppose that I want to train A to produce novel critical analyses. I train A on a massive database of critical analyses. I then provide it with a subject and ask for an analysis.

Here x is a subject descriptor, y is an analysis, and $A(x)$ is the analysis output by A.

I want my system to output a novel, interesting analysis. $U(x, y)$ is a measure of the novel contribution of y on the subject x .

It is easy for $B(x, y)$ to estimate the quality of an analysis (at least well enough to incentivize A to try its hardest). But estimating originality can be quite difficult. A plagiarized analysis might subtly transform the source text in any number of ways, substituting synonyms, making changes to sentence structure, or even reworking high-level structure. And even an original text may draw heavily on ideas from other sources rather than making substantively new contributions.

It is extremely difficult for B to assess the novel contributions of y without pointers to all of the relevant texts that A drew from. In order to make this assessment B may have to be *much* smarter than A. If B is merely slightly smarter than A, then A's optimal strategy may be to borrow liberally, even though B is trying to measure novel contributions.

(The “logical facts” in this example are facts about the relationship between $A(x)$ and prior art. These relationships are evident if we observe and understand A's computation but might otherwise be obscure.)

This example is very simple and contrived. In reality, we are concerned with a much murkier situation, where there is a very large space of possible considerations and important observations, and some cognitive strategies might lead to actions that look good if and only if you don't understand the process that produced them.

Toy model [query complexity]

It may be possible to view a program as uncovering facts about the results of various subcomputations. But we don't really know how to do it for now, and in general logical observations seem very complex.

We can obtain a nicer formal model for our problem by imagining that A and B have unbounded computational resources, and

introducing an explicit oracle O to model their logical uncertainty.

For each bit string x , $O(x)$ is another bit string of the same length. There may be any number of complex relationships amongst the different values of O , which are intended to represent the results of running various computations. We assume that A and B are computationally unbounded, but that each of them can only make a limited number of queries to O .

We'll define the function U in terms of O . Namely:

- For most inputs x , $U(x, y) = 1$ if and only if $|y| = 10|x|$ and $O(y)$ begins with x .
- The only exception to the rule above is that $U(x, O(zxx)) = 0$ for all z .
- For any string z , $O(O(zxx))$ begins with x .

These facts are available to A and B .

The natural strategy for $B(x, y)$ is to simply evaluate $O(y)$ and test if it begins with x .

The intended strategy for A is to search for an input y for which $O(y)$ begins with x . There might be any number of additional regularities in O that help with this search, and we would like for A to learn to use these regularities to speed up its search.

However, A has a much simpler cheating strategy: pick z of length $8|x|$, and output $A(x) = O(zxx)$.

The question is: can we modify the setup in any way so as to incentivize A to find a value of y for which $U(x, y) = 1$?

Why this problem?

Why solving this problem might be good

My recent proposal ALBA has a number of significant problems. Aside from solving the informed oversight problem, it requires: robust prediction algorithms, efficient active learning, efficient semi-supervised learning, and adequate algorithm learning schemes.

Aside from informed oversight, all of these problems:

1. Are more-or-less traditional AI challenges. I expect these problems to receive significant attention by default, and it would be relatively easy for researchers interested in AI safety to focus more on them.
2. Look like they should definitely be possible—I think we'd need to learn something surprising to find that one of them was impossible.

[ETA: while robust learning remains a traditional AI challenge, it is not at all clear that it is possible. And meta-execution actually seems like the ingredient furthest from existing ML practice, as well as having non-obvious feasibility.]

I don't think that these other problems are trivial by any means. But I am optimistic that we can resolve them. I don't think they are likely to be deal-breakers.

The informed oversight problem seems like the most likely place to encounter fundamental conceptual difficulties. If it turns out that the control/alignment problem systematically resists attack for a fundamental theoretical reason, then I think that informed oversight is the step most likely to be secretly hiding the real difficulty. Learning that any other step was “the hard part” would require significantly revising our current understanding of the control/alignment problem.

Note that solving informed oversight would most likely also address problems with malicious inputs.

Necessity for approval-directed agents

If the informed oversight problem proves to be impossible or very difficult, then that is probably a deal-breaker not only for ALBA but for anything resembling concrete approval-directed agents. Since this is currently my preferred approach to value alignment, learning that it can't work would be important information for me.

Why solving this problem might be necessary

The informed oversight problem might be necessary to building *efficient* aligned AI by any means, if it turns out that reinforcement learning is significantly more powerful than other techniques for learning complex policies.

If RL is the only game in town, then in order to train a policy to do what we want, we need to be able to compute a reward signal measuring how good a given policy is.

If we restrict attention to the single-action case, we need to be able to compute a reward signal measuring the quality of a response $A(x)$ to a situation x .

It may be that there is some special structure to the actual utility function U that we want to optimize, and that somehow this particular U is surprisingly easy. But I don't consider this especially plausible—we can probably treat our U as a black box.

So we want to build a training signal that proxies for $U(x, y)$. Let's just call that signal $B(x, y)$.

B essentially solves the informed oversight problem. The modifications allowed in the informed oversight problem are a superset of those that are acceptable in this setup—we can modify B arbitrarily, and we can modify A in any way that doesn't significantly reduce efficiency.

The informed oversight problem is only made easier by assuming that we have access to an aligned agent B which is significantly smarter than A . In some sense the goal of ALBA is to show that we can actually make this assumption without loss of generality.

It feels tractable

An important intuition for working on the informed oversight problem is that it feels pretty easy. Many of the intuitive difficulties of value alignment don't appear in this case, since we are dealing with a weak agent. Moreover, it feels like a wide range of new strategies are available. For example, normally I would think of transparency as having a somewhat indirect relationship to the control problem, but it seems like transparency might immediately resolve the informed oversight problem (see the next section).

It may be that the informed oversight problem already captures all of the difficulty of the value alignment problem—that it is no harder than the value alignment problem, but also no easier. I think this is conceivable, but if true I think it suggests a change in our view of the value alignment problem, of what the key difficulties are, and of what the most promising approaches are.

Attacking the problem

(Warning: rambling and no really satisfying progress, you may want to skip to the conclusion.)

I think the most plausible angle of attack on this problem is to make the agent A “show its work” so that B can accurately evaluate its output.

There are other possible angles, but nothing else seems very promising so far. So this section is going to be exclusively about transparency.

Understandable model class

The simplest approach is for B to look at A’s computation, understand that computation, learn everything that A learned, and then make an informed judgment.

In the toy model, this corresponds to giving B a list of all of the $O(a) = b$ pairs that A queried over the course of its computation.

This would be an adequate solution if you could actually do it. The problem is that the “queries” to O occur inside of A’s computation (they are analogous to logical manipulations that reveal logical information). So it’s not obvious if there is any way to extract this information from A.

We could hope that some model classes are simply “understandable,” in the sense that someone looking at the computation transcript could recover all of the logical facts that are accessible to the computation.

This is clearly not the case in general—e.g, consider a cryptographically obfuscated program. But maybe it is the case for some big-enough class of models?

I’m a tiny bit optimistic about this in the case of attention, external memory, and other computational aids. In these cases, we can maybe watch the model is doing. For example, if our agent is actually looking at some documents from its database, we can see which documents it is looking at and so narrow down the space of possible logical facts it might be learning.

But in general, it seems like there aren’t good candidates for “understandable” models of computation, so far we haven’t had much

success training inherently comprehensible models, and there isn't any theoretical reason to think that it is possible.

Training to be transparent

It sounds much more promising to train our models to produce explanations of their behavior. If logical facts are “available” to the model itself, so that they can be used to find a good output $A(x)$, then it seems intuitive that the same logical facts are “available” to an explanation-generator that is being trained at the same time.

Because B is smarter than A , and is only interested in learning logical facts revealed by A 's computation, this problem seems easier than the general problem of making machine learning systems transparent. Critically, for any logical fact revealed by A 's computation, there is a short proof of that fact. Because B is smarter than A , we expect B to be able to understand that proof if A can. So B can evaluate for itself whether a given logical fact is true, and can also evaluate whether it is relevant to estimating U .

So let's assume we have some function $\text{Info}(x, y, z) \in [0, 1]$ which evaluates how informative z is about $U(x, y)$.

One simple approach is to train a second model C which looks over the computation done by A and tries to extract a useful explanation z . We can train C to maximize $\text{Info}(x, y, z)$, and then provide its output z as an auxiliary input to B , i.e. to compute $B(x, y, z)$.

But this doesn't really buy us anything—it only works if A 's computation is transparent to C . And the same considerations that make the computations of subhuman AI systems opaque to humans, or that make A 's computation opaque to B , might as well make A 's computation opaque to C .

There is a fundamental asymmetry between the role of A , using its own internal representations to generate an output, and the role of C , using A 's internal representations to generate explanations. The internal representations are themselves optimized to make A 's task feasible, while they are not optimized at all to help C .

Maximizing $B + \text{Info}$

Ideally, we could train A itself to output both y and z . A very naive version is to train A to output y, z maximizing $B(x, y, z) + \lambda \text{Info}(x, y, z)$.

The problem with this approach is that providing good information might decrease $B()$, and that the actions with the highest values of $\text{Info}()$ might not be best.

If we make λ small, then A will omit negative information in order to ensure that B is large. But if we make λ large, then A will choose whatever actions have the most possible info to be revealed.

To illustrate the difficulty, suppose that we want A to produce original work, with z being an annotated list of sources and explanations of their relevance. Then $\text{Info}(x, y, z)$ might count the number of references. If λ is small then A will simply leave out references in order to make its work appear more original. If λ is large then A will create maximally derivative work, precisely so that it can include plenty of references.

In general, maximizing a function of B and Info seems incapable of distinguishing between an output y which needs no explanation, and one for which the explanation is being withheld.

The only way I can see this working is if the maximal value of $\text{Info}(x, y, z)$ doesn't much depend on y . For example, we might hope that A can explain *all* of its internal deliberation, and then set $\text{Info}(x, y, z) = 1$ iff z seems to be a full account of A 's internal deliberation.

This doesn't really seem workable to me. At the least, it would require a really deep understanding of what our algorithms are doing.

Maximizing multiple objectives

It would be better if y was optimized for $B(x, y, z)$ and z optimized for $\text{Info}(x, y, z)$.

There is an obvious way to try to do this. I'll assume that A is trained by gradient descent with backpropagation, but the same argument would apply to any local search.

In backpropagation we inductively compute the derivative of our objective with respect to each parameter and intermediate result. We can effectively ignore some causal pathways by simply dropping some terms from this gradient computation.

In this case, we can define the “gradient of our objective” with respect to z to be the gradient of $\text{Info}(x, y, z)$ with respect to z (ignoring z 's effect on $B()$). And define the “gradient of our objective” with respect

to y to be the gradient of $B(x, y, z)$ with respect to y . Then carry out the rest of backpropagation as usual. (Of course the quantities computed by this modified backpropagation are not actually the derivative of any objective...)

I don't think this really does what we want, and I haven't been able to find any variant that does.

But this approach does do *something*, and generally points in the direction we want. (I think it actually does do what we want under implausibly strong continuity/smoothness conditions.)

I think the original goal isn't fundamentally incoherent, but I have no idea whether there is any way to formalize and achieve the intuitive goal.

Conclusion

The informed oversight problem asks a relatively powerful and well-informed principal to incentivize good behavior from a weaker and more ignorant agent.

I've argued that a solution to this problem would leave us in a good place with respect to the value alignment problem, and that we may not be able to satisfactorily resolve value alignment in advance *without* having a solution to this or a similar problem.

It may be that the informed oversight problem captures all of the difficulty of the value alignment problem. If so, I think that should lead us to reconsider the nature of the alignment problem.

Encouragingly, if we could train *transparent* RL agents then it looks like we could solve the informed oversight problem. It's not clear whether a strong enough form of transparency is possible even in principle, but it does provide a natural angle of attack on the problem.

I think that working on the informed oversight problem is a natural angle of attack on the alignment problem given what we currently know. Whether or not the problem is resolved, I expect further work to clarify our picture of value alignment broadly, and especially our understanding of approval-directed agents in particular.

My short-term agenda



Paul Christiano

[Follow](#)

Mar 3, 2016

I think that the informed oversight problem is the biggest open problem with my recent proposal for aligned AI.

That said,

1. If we really established that informed oversight was the main missing ingredient, it would change our understanding of value alignment.
2. I think that other people are (justifiably) not yet convinced.
3. If informed oversight was the main missing ingredient, then our best bet may be to take an ad hoc approach to informed oversight even if there is no theoretically solid approach.
4. The informed oversight problem looks quite hard; I think it will be much easier to get traction on the other problems.

So my plan is to temporarily set aside the informed oversight problem and focus on the other difficulties. In rough order of priority:

1. Bootstrapping.
2. Robust prediction.
3. Robust active learning.
4. Efficient semi-supervised learning.

I don't think that any of these are straightforward, but I do feel pretty good about all of them. That view may change as I spend more time working on them.

Efficient and safely scalable



Paul Christiano

[Follow](#)

Mar 23, 2016 · 15 min read

Precisely defining the goal of AI control research seems quite difficult. This post gives preliminary definitions of **safe scalability** and **efficiency** for AI control protocols, taking a step towards formalization. Roughly, these properties say that “using better machine learning primitives results in better systems” and “the control scheme does not impose significant overhead.”

I think these properties are probably sufficient conditions for success, but they are also probably too ambitious to be realistic goals. I discuss a few possible ways to weaken these definitions.

Both scalability and efficiency are defined with respect to a preference order $>^D$ which tells us when one algorithm is “better” another on some distribution D, according to the user’s preferences. I won’t offer any precise definition of $>^D$, but I’ll discuss a few informal candidates.

Motivation

I’m interested in defining alignment formally for at least three reasons:

- Having a precise goal makes it easier to do good and well-targeted research. The AI control problem would feel much easier to me (both to work on and to talk to others about) if there were a precise, satisfactory, and achievable goal.
- A precise definition of alignment might be helpful when analyzing AI control schemes. For example, the analysis of ALBA calls for maintaining alignment as an inductive invariant as the agent becomes more powerful. Right now, there is little hope of making that argument formal.
- Trying to formalize alignment may shed light on what the key difficulties are, what assumptions are likely to be necessary, and so on. Trying to pin down slippery concepts is often a good idea .

Definitions

What is a control protocol?

Our AI control protocols will use machine learning primitives as building blocks, and construct a (hopefully aligned) AI out of them.

To instantiate a control protocol ALIGN , we provide some set of learning primitives that are required by the protocol. ALIGN then instantiates any number of copies of each of those primitives. ALIGN may choose what inputs to provide to those instances, and may use their outputs however it likes. ALIGN may also interact with the user arbitrarily.

For simplicity, throughout the post we will assume that ALIGN is built from an RL algorithm, and write $\text{ALIGN}(\mathcal{A}^{\text{RL}})$ for the algorithm obtained by using \mathcal{A}^{RL} . Note that ALIGN can instantiate any number of distinct instances of \mathcal{A}^{RL} , can provide each of them distinct rewards, and so on.

All of our definitions can be easily extended to any set of machine learning primitives, as long as we can define what it means for one implementation of a primitive to “outperform” another on a given distribution. I think that the definitions are most interesting when we can efficiently test whether one implementation outperforms another, and amongst such primitives RL is essentially universal (since we can use the test itself as a reward function).

Betterness

What does it mean for one algorithm to be better than another?

We won’t answer that question. Instead, we take as given a family of preorders \ll^D indexed by distributions D . These orders define when one program “outperforms” another on the distribution D , according to the user’s preferences.

Intuitively, we can imagine some (unobserved) utility function U characterizing the user’s preferences. U takes as input an (x, y) pair, and outputs a real number reflecting how good it is, according to the user’s preferences, for a program to output y given input x .

Then we could define:

- $A \ll^D B \iff \mathbb{E}[U(x, A(x))] \leq \mathbb{E}[U(x, B(x))]$,

where the expectations are taken over x drawn from D . We could also strengthen the condition by allowing A to score nearly as well on

average over many episodes, or by requiring B to do at least as well as A for every utility function in some big set \mathcal{U} . These changes make $<^D$ a preorder rather than a total order.

We'll give a range of more plausible definitions in the final section, but this is a good concrete example to have in mind.

Safely scalable and efficient

Intuitively, a control scheme is *safely scalable* if using better RL algorithms never results in worse performance. A control scheme is *efficient* if it can realize any level of performance with low overhead.

- **Safely scalable (with respect to $>^D$)**. Given any A^{RL} , B^{RL} , and the ability to sample from some distribution D with $\text{ALIGN}(A^{RL}) >^D \text{ALIGN}(B^{RL})$, we can sample from a distribution D^{RL} over RL episodes on which A^{RL} outperforms B^{RL} .
- **Efficient (with respect to $>^D$)**. Given any B , we can implement an RL agent B^{RL} such that, for every distribution D from which we can sample, $\text{ALIGN}(B^{RL}) \geq^D B$.

Efficiency is quantified by how much more time $\text{ALIGN}(B^{RL})$ takes than B itself, and how often it queries the user. I'll say that a scheme is "**asymptotically efficient**" if the difference in running times, and the number of queries to the user, is sublinear in the number of episodes T .

In order to be achievable this definition probably needs to be weakened. I think the most plausible way to weaken it is to make additional assumptions about the agent B in **efficiency**. For example, we could focus our attention on a particular approach to building AI systems, and assume that B is the kind of agent that might be produced by that approach. Particularly interesting are structural assumptions about how B itself is built out of the same building blocks that are available to ALIGN .

Hard to beat

Together efficiency and safe scalability imply a third property:

- **Hard to beat**. Given any RL agent A^{RL} , any agent B , and the ability to sample from a distribution D with $B >^D \text{ALIGN}(A^{RL})$, we can implement an agent B^{RL} and sample from a distribution D^{RL} over RL episodes on which B^{RL} outperforms A^{RL} .

If an algorithm is “hard to beat,” then the only way to make it better (according to $>^D$) is to improve the underlying RL algorithms. In some sense this is the strongest form of optimality that we can realistically hope for, since improving our RL algorithms will allow us to build “better” AI systems for any reasonable notion of “better.”

To see that (efficient + scalable \rightarrow hard to beat), apply efficiency to find an agent $\text{ALIGN}(B^{RL}) \geq^D B$, use transitivity to infer that $\text{ALIGN}(B^{RL}) >^D \text{ALIGN}(A^{RL})$, and then to use safe scalability to sample from a distribution where B^{RL} outperforms A^{RL} .

Being hard to beat is slightly weaker than being efficient + scalable while being almost as comforting. So it might also be useful as an easier goal.

Restrictions on the building blocks

Rather than working with a generic RL algorithm, we might want to work with an RL algorithm that satisfies some additional property. For example, $\text{ALIGN}(A^{RL})$ might only be safely scalable if A^{RL} is able to make good enough predictions about humans, or if A^{RL} is “transparent” in an appropriate sense.

These restrictions can be incorporated into the definition of safe scalability and efficiency—in safe scalability, we can strengthen the hypothesis by assuming that the learning algorithms satisfy the restrictions, and in efficiency we can strengthen the conclusion by assuming that the learning algorithms satisfy the restrictions.

To make the task easiest, we could strengthen the hypothesis of safe scalability *without* strengthening the conclusion of efficiency. For example, if we want to work with “transparent” RL algorithms, we will probably not be able to strengthen the conclusion of efficiency—we won’t be able to turn a black-box algorithm B into a transparent RL algorithm B^{RL} . So we could instead aim for a scheme that is safely scalable when applied with transparent RL algorithms, and that is efficient when we are allowed to use arbitrary RL algorithms. The resulting control scheme would only be practically efficient to the extent that there are transparent nearly-state-of-the-art RL algorithms.

For now I am interested in schemes that work under *any* remotely plausible assumptions:

- Even a conditional result would be a big advance beyond our current understanding.
- Conditional results could clarify our understanding of when AI control will and won't succeed.
- Conditional results present natural targets for differential AI progress. For example, if we could formulate a transparency condition that was sufficient for building safe+efficient AI, this could help clarify the goals of research on transparency.

Related building blocks

We might also work with a number of different building blocks, and require them to have some relationship to each other. For example, we might want to use both a classifier and an RL agent, and assume that the classifier is in some sense more powerful than the RL agent (though it's not yet clear to me what assumption of this form would actually be useful). Or we might work with a sequence of RL agents, and assume that each is only slightly more powerful than the last.

Non-solutions

Plain RL (+counterfactual oversight)

The simplest behavior for ALIGN is to simply consult the underlying RL agent directly, and then to define a reward that induces aligned behavior.

Perhaps the simplest reward function is a human's evaluation: a human watches the agent and presses the reward button when the agent behaves well.

If the human has to monitor the agent during every episode, then this constitutes a huge efficiency hit and so our algorithm won't be reasonably efficient.

More realistically, we could train our agent on a small subset of the data (for which the human does have to monitor the agent and press the reward button) and then use the learned policy to act even when there is no human monitor. If necessary, we could train online as in counterfactual oversight.

This approach is asymptotically efficient (as long as the size of the training set is sublinear in the total number of episodes, as would be typical).

However, this scheme is **not** safely scalable for a reasonable order $>^D$.

The problem is that the user's judgments are not exactly aligned with the user's preferences. For any given input x , there are some outputs y for which the user assigns y an "erroneously" high expected value, whether due to error, to bias, or simply to not having enough time to understand the consequences of a proposed action.

So let A^{RL} be an RL agent which outputs actions which the user mistakenly rates highly, and let B^{RL} be an RL agent which outputs actions that are better than A^{RL} 's outputs but which are rated lower by the user. As long as the user's judgments don't perfectly preserve the ordering of outcomes, then we can construct such a pair. Then we can have $\text{ALIGN}(A^{RL}) >^D \text{ALIGN}(B^{RL})$ for most distributions D , while B^{RL} outperforms A^{RL} on *every* distribution D^{RL}

Plain imitation learning (+counterfactual oversight)

Given observations of a human expert, we can give A^{RL} the problem "predict what output the human expert will produce on input x ."

After A^{RL} produces a prediction, with small probability we ask the human to actually produce an output y , and then we provide A^{RL} a payoff depending on the quality of its prediction. (There are a number of possible ways to score A^{RL} , most involving two separate learners. For example, we could use a generative adversarial model, with a second instance of A^{RL} trying to distinguish the predicted output from the real human output. Or we could score A^{RL} based on a variational lower bound on the log probability it assigned to the particular prediction made by the user.)

This kind of imitation learning is not safely scalable: an agent which chooses superhuman actions will become strictly worse if we make them a perfect human-predictor, yet their performance in the RL game will improve.

(Intuitively, imitation *is* safely scalable but *is not* efficient—this example highlights a way in which our definitions fail to map well to the intuitive definitions of these concepts.)

ALBA

The current version of ALBA is not safely scalable + efficient, even under the generous assumptions made in the post. The key problem is its reliance on a sequence of throttled RL subagents.

The missing ingredients, roughly speaking, are the following:

- **Speedup.** Given black-box access to the agent B , we can produce a *strictly less capable* agent B' such that $\text{Bootstrap}(B') \geq^D B$.
- **Informed oversight.** See here.

Speedup is impossible without some additional assumptions on B , and informed oversight probably also requires some additional assumptions (in order to make transparency possible). It's not clear what kind of assumptions on B might be sufficient to carry out the analysis, and overall it looks like a daunting project.

Limitations

Too strong

I think the biggest problem with this definition is that it is too strong.

It's not so strong as to be *obviously* impossible. But it looks *almost* obviously impossible. The discussions of RL and ALBA illustrate why the definition is so strong:

- In order to turn a black box agent B into an agent B^{RL} with $\text{ALIGN}(B^{\text{RL}}) \geq^D B$, we are essentially forced to take $B^{\text{RL}} = B$ (since we can't produce other derivative agents using black-box access to B).
- Then ALIGN is essentially forced to be a training scheme for RL agents.
- So in order to be safely scalable, ALIGN needs to evaluate of the quality of the agent's decisions "well enough" that optimizing its evaluations optimizes $>^D$.
- Moreover, ALIGN can't really use the RL agent's help to make those decisions—if ALIGN is merely a training procedure, the RL agent need not output anything except on the support of D , and so we can't get any useful work out of the agent. Thus ALIGN is using the same evaluations for every agent.
- If ALIGN evaluates the agent's behavior "well enough" for an arbitrary agent, then ALIGN must be evaluating the agent's behavior perfectly.

- It seems infeasible to produce such a perfect evaluations for any interesting $>^D$.

How might we weaken the definition?

- Place some restriction on the set of agents B that we consider in **efficiency**. For example, we may restrict attention to the kinds of agents that could be produced by some particular AI research project in AI. I think that this is by far the most promising approach.
- As discussed in the section **Restrictions on building blocks**, we could only require safe scalability for a certain class of RL agents, thus moving some of the work to ensuring that state-of-the-art RL agents have the required properties.
- We could use relations $>^D$ that evaluate agents holistically in terms of a *description* of the distribution D (see below). For example, we might say that “A $>^D$ B if the human believes that A would outperform B on the distribution D.” I don’t really see a way to make this work, but it might be worth thinking about.
- We could settle for an agent which is hard to beat instead of both efficient and safely scalable. I don’t think this really addresses the difficulty described above, but it does give us a tiny bit more traction.
- We could swap the quantifier order, giving us access to B and D when trying to construct an agent B^{RL} with $\text{ALIGN}(B^{RL}) \geq^D B$. I don’t think this will help.

I expect there are many other ways to weaken the definition, and of course we could pursue some combination of the above.

Improving RL algorithms is quite broad

Even if ALIGN is efficient and safely scalable, $\text{ALIGN}(A^{RL})$ isn’t necessarily *good* even according to $>^D$. In order to make $\text{ALIGN}(A^{RL})$ actually be good, we may need to improve A^{RL} . In some sense this is obvious and inevitable—it’s like saying that even if we solve the control problem, AI progress will still make our AI systems work better.

But in particular, the alignment of $\text{ALIGN}(A^{RL})$ may depend on how A^{RL} performs on some very unnatural distribution over RL problems (e.g.

on how well A^{RL} is able to predict the results of human deliberation about moral questions).

Given how strong safe scalability and efficiency are, I don't think this is a problem for this particular definition. That is, any such "unnatural" distribution over RL problems would be necessary to achieving good behavior, even for very weak agents:

- Given any aligned agent B we can apply efficiency to obtain an RL agent B^{RL} .
- If B^{RL} always does well on these unnatural instances, then they were in some strong sense *necessary* in order to get good behavior. But note that B might be quite weak, so these "unnatural" instances were necessary even to getting good behavior out of weak AI systems.
- If B^{RL} does poorly on these unnatural instances, then we can consider a very powerful agent A^{RL} that does equally poorly.
- By safe scalability, $\text{ALIGN}(A^{RL})$ remains at-least-as-good-as- B for that RL agent A^{RL} .

This argument would still go through if we restricted B to the kinds of AI's that we might actually produce. (The conclusion would be that the unnatural RL instances are in fact necessary, *for the kinds of AI that we might actually build*, which is just as good.)

If we weakened the definition enough we might encounter a more troubling version of this unnatural-RL-instances problem, but even then I think that reducing the alignment problem to a concrete RL problem would probably represent significant progress in and of itself. So overall I'm not concerned about this potential bug.

Missing definition of $>^D$

We don't provide any definition of $>^D$; that definition needs to be quite subtle and is doing a lot of the work of formalizing our goals. See the discussion in the next section.

Defining $>^D$

Human judgment

For sufficiently subhuman agents we could define:

- Draw a random x from D , compute $A(x)$ and $B(x)$, and give them both to a human judge (along with the input x) to decide which is better. $A >^D B$ if the human prefers $A(x)$ with probability at least $2/3$.

For this definition of $<$, the RL agent described above is in fact efficient and safely scalable.

But being safely scalable in this sense doesn't seem very useful, since the human's judgment about "which action is good" need not be correct, and the human would substantially prefer an agent that chooses actions which are *actually* good.

Human judgment on reflection

We could try to fix this definition with an alternative informal definition:

- Draw a random x from D , compute $A(x)$ and $B(x)$. Give a human judge an extremely long time to reflect, and access to extremely powerful tools. $A >^D B$ if, upon extensive reflection, the human prefers $A(x)$ with probability at least $2/3$.

This definition leads to a very strong notion of safe scalability. Unfortunately that strong notion is *too* strong, and it is clearly intractable to build a hard-to-beat agent in this sense.

For example, consider some hard question that bears on what strategies will have good long-term effects. This question is beyond our abilities currently, but we will be able to answer it with sufficient reflection. If B guesses the correct answer to that question and A does not, then $A <^D B$, where D is the a point distribution concentrated on a decision where that question matters. If B guesses the wrong answer, then $B <^D A$. But by efficiency + safe scalability, we can essentially produce a pair of agents A^{RL} and B^{RL} and sample from a distribution D^{RL} such that $A <^D B$ iff B^{RL} outperforms A^{RL} on D^{RL} . But then we can test which of A and B is correct about the hard question, contradicting its supposed hardness.

Holistic judgment

A very different approach would be to directly elicit the human's judgment about the relative merits of A and B . That is:

- Give a human judge a description of A and B , and the sampling procedure for D , and ask them which they expect to perform

better. $A >^D B$ if the human thinks that A would make significantly better decisions than B, on inputs sampled from D.

In some sense this definition gets very directly at what we care about. Suppose that some alternative AI control research program produced B, and that we would be happier using B than A. Then we are precisely satisfying this definition of $A <^D B$.

An obvious problem with this approach is that determining which of A and B is better may itself be a very hard problem—indeed, it's easy to believe that it includes all of the difficulty of the alignment problem.

The definition of efficiency still has teeth if we restrict our attention to algorithms B that “explain themselves,” e.g. that are accompanied with significant documentation and explanation. The description of the agent B could embody the entire output of the alternative research program that produced it.

So this definition of betterness is adequate if we think that there are efficient and good decision-making protocols which are *demonstrably* good to existing humans. This definition is very unsatisfying if we think that evaluating a possible proposal, even given the best available arguments, is the core difficulty of AI control. This might be either because those arguments are necessarily extremely complex, or because there will be many bad proposals that are also supported by extremely convincing-looking arguments.

What we really want

Intuitively, I would like a definition along the lines of:

- Draw a random x from D, compute $A(x)$ and $B(x)$. Give a human judge the same information, abilities, and insights, that A and B used to compute these quantities. $A >^D B$ if the human prefers $A(x)$ with probability at least 2/3.

This definition smuggles all of the complexity into imagining that the human has the same “information, abilities, and insights” as the AI they are evaluating. I don’t have any candidate formalization of this idea, nor am I especially optimistic about being able to formalize it.

I do feel like I can reason about this definition intuitively and that it roughly captures my intuitive desiderata. This makes me more optimistic that there is *some* satisfactory definition of $>$.

Note that this definition is closely related to the goal in the informed oversight problem, which is roughly to ensure that the overseer “knows everything the AI knows.” In the informed oversight problem we are willing to assume that the overseer is significantly more powerful than the system they are overseeing. That may well be a necessary assumption to actually ensure that the overseer “knows everything the AI knows,” but it probably isn’t needed to define what it would mean for the overseer to “know everything the AI knows.”

Conclusion

We can try to define the goals of AI control by thinking about how AI systems relate to the underlying machine learning primitives. Such a framework wouldn’t cover all possible approaches to AI control, but where applicable it could be a great way to organize research and a useful analysis tool.

This post gave a step in that direction, but did not yet succeed. I would love to see other attempts, and I think there is a good chance that it will be possible to find a satisfying problem statement for AI control.

Strong HCH



Paul Christiano

[Follow](#)

Mar 24, 2016 · 5 min read

In a previous post I defined humans consulting HCH (HCH). In this post I'll define a more powerful process. In the future I'll use "HCH" to refer to this stronger process, and call the old process "weak HCH."

As before, we consider a particular human Alice interacting with a particular computer over a particular interval of time, say two weeks in mid-August. The computer provides two "magic" functions:

- The computer presents an initial message to Alice. After Alice replies to that message, the computer may immediately give her a follow-up message, she may submit a follow-up reply, it may follow-up again, and so on. This dialog is the input/output of the overall process.
- The computer also allows Alice to consult assistants. Each assistant is assigned a unique ID, and Alice can communicate with them by entering a message together with the appropriate ID. Initially, the computer tells Alice the ID of a single "fresh" assistant.

HCH is the input/output functionality implemented by Alice, if each of her assistants also implements HCH. We can either define HCH as a fixed point, or we can force the tree of assistants to be well-founded and define it by induction.

Defining the interaction between Alice and her assistants is the main subtlety.

Each time Alice sends a message to an assistant, the assistant is copied, and the message is sent to the new copy. Alice is then immediately given the copy's reply, as well as its ID. At that point Alice has two ID's, and she can send messages to either—sending a message to the new ID allows her to continue the conversation, while sending a message to the old ID allows her to ask an alternative question. Each new message spawns an additional copy. If an assistant has used up its 2 weeks, then subsequent messages receive an empty reply.

Messages between Alice and her assistants can contain both text and IDs of other assistants. When an ID is sent as part of a message, the corresponding assistant is copied and a new ID is generated. The recipient sees the new ID rather than the old ID, and can use the new ID to interact with the new copy. These ID's can be short (say 10 letter) strings, because any given copy of Alice can only have direct access to a modest number of IDs.

Comparison to weak HCH

In the previous version of HCH, each assistant answered a single question and then disappeared. The new version differs in that:

- Assistants can carry on discussions instead of simply answering questions.
- Messages can contain pointers to other assistants. This allows the implicit size of messages to be arbitrarily large (since those assistants may themselves have pointers to other assistants...)

I think the new formulation of HCH is significantly better, and in the future I am going to use “HCH” to refer to this version.

It’s not clear how much the first change increases the power of HCH, but intuitively it feels like a significant improvement.

The second change feels like a big intuitive improvement, and also increases the complexity-theoretic expressiveness of HCH. The old version could be computed in EXPTIME, while the new version can compute any decidable function.

Generalizing

As with weak HCH, we can define HCH^P or max-HCH^A . These definitions are actually quite subtle—both of the changes mentioned in the last section lead to significant new complexities. I’m not really sure if $\text{HCH}^{\text{market}}$ can be adapted to this setting.

We can bound the complexity of HCH by putting a limit on the total number of copies created, or on the total number of minutes that copies spend thinking. Once the limit is exceeded, no more messages can be sent. There might be a global budget, or each copy might have their own budget which they have to explicitly share with any new copies they instantiate. I’ll refer to these variants collectively as “bounded HCH.”

We might describe the process above as $HCH(A)$ (Alice thinking for 2 weeks). We could more generally define $HCH(A)$ for any process A, by replacing “Alice thinking for two weeks” with an instance of A.

Different processes A may use different communication channels (for example, we could give Alice the ability to video chat with her assistants). The only restriction is that the communication channel be able to include IDs, both to address the message and to include pointers to other assistants.

On universality

I think that this stronger form of HCH is in some sense a “universal” way to combine a bunch of black-box instances of A. That is, for any alternative program $ALT(A)$ that combines a bunch of black-box instances of A, there is some advice we could provide to A that would cause $HCH(A)$ to simulate $ALT(A)$ with only a linear slowdown. Moreover, this advice depends only on ALT , and not on A itself—it works for any “cooperative” A that understands the advice and effectively follows instructions.

For example, this version of HCH subsumes any computable black-box reflection procedure (including this one).

(Technically, this only works if ALT is simple enough that it can be implemented by A. In order to make the model formally universal, we would have to augment $HCH(A)$ by giving it access to an arbitrarily large input.)

This argument is informal—the reduction doesn’t work for any A, and it’s not clear exactly what is required to make it work. But intuitively it seems to work for the class of cooperative, intelligent English speakers.

If we remove either of the two augmentations described in the section **Comparison to weak HCH**, then the new model seems to be strictly less powerful. There is no way for either of those impoverished models to effectively simulate HCH (without first building a simulation of A, either violating the black-box assumption or introducing exponential overhead).

The interesting caveat to universality is that a human A can probably carry out the reduction well but not perfectly. So if the computation is very large a naive reduction may fail with high probability. We could try to implement appropriate error-correction procedures to drive

down the probability of error, but it is extremely hard to analyze those measures formally. (If Alice makes independent random errors then error-correction will work for driving down the error probability to an arbitrarily small constant, but that's not a good error model.)

So, given error-prone A, it is not really clear whether HCH is universal. I haven't thought much about this issue.

This form of universality does not rule out the utility of thinking more about how to combine additional copies of A. It just implies that such thinking won't require any change to the structure of HCH. This means, for example, that it could be carried out inside of HCH as the first step of a complex deliberation (but there is no guarantee that HCH can think of anything that we could think of).

Conclusion

I've defined a much stronger form of HCH; in the future I don't think I will have much reason to use the original form. This stronger HCH seems to be essentially universal amongst all procedures that invoke human behavior as a black box.

The last CAPTCHA



Paul Christiano

[Follow](#)

Apr 5, 2016 · 6 min read

I think we might be nearing the last moment in history when it is possible for a human to beat a machine at *any precisely defined game*.

To formalize this, consider the following meta-game:

- The **CAPTCHA designers** produce a CAPTCHA and publish it.
- The CAPTCHA is built to interact with two agents at once, and has to guess which is a human. The interaction should be through a keyboard/mouse/monitor, take place over the course of at most 8 hours, and not involve the outside world. For example, the CAPTCHA can organize a game of Go between the two agents.
- We pick the single human who is most likely to be able to win the CAPTCHA (the **champion**). A team of coaches can spend a year trying to help the champion train to win at the CAPTCHA.
- A sophisticated research team spends a year designing an AI to beat the CAPTCHA (the **machine**). The designers can use any resources at their disposal, and as much computing power as they can get. They have access to the source of the CAPTCHA and can run it as many times as they like.
- We play a sequence of matches between the champion and the machine, starting one year after the publication of the CAPTCHA.
- The CAPTCHA designers and the champion win if the CAPTCHA can identify the human in $2/3$ of trials.
- The AI designers win if the machine can pass as human at least $1/3$ of the time. (We could replace $1/3$ with any constant less than $1/2$.)

Until very recently it was clear that the champion would win, since Go is a simple counterexample and it seemed unlikely to be solvable with a year of effort.

My view of the current state of affairs

I don't think that Go itself is the last CAPTCHA. But I do think that we are rapidly running out of good contenders:

- There are board games and strategic video games which humans can currently win, but there are none which I would expect to stand up to a year of serious attention.
- Similarly, there are perceptual problems where humans can still outperform machines. But given access to any distribution over instances, I wouldn't count on that particular distribution lasting a year. Also, we can't really collect a big enough data set of natural images that the AI team couldn't just label them all by hand.
- There are motor control problems with well-defined success criteria where a human will beat a machine. But things get easier when the task occurs in a simulation. And they get much easier when the distribution over tasks is narrow enough that it can be algorithmically generated, and when the AI designers are willing to use arbitrarily special-cased software. I don't know if there any real contenders here.
- There are tasks involving symbolic reasoning that seem hard for AI systems. But it is usually very difficult to generate hard instances. For example, a "random" math problem is more likely to be easy for a machine than a human—humans excel at a narrow class of problems that we find interesting. And we can't algorithmically sample instances from that distribution.

My best single guess is that we will run out of CAPTCHAs within ~5 years. What do you think? What are the strongest candidates for robust CAPTCHAs, and how long do you expect them to last?

The range of Turing tests

We can consider three increasingly difficult tests. In each, the judge has some moderate length of time, say a day, to make their determination:

- **General CAPTCHA.** An automated judge interacts with two agents, and guesses which is a human and which is a machine.
- **Adaptive CAPTCHA.** Rather than publishing a CAPTCHA and then designing an AI, the AI designers *first* publish their AI, and

then the CAPTCHA designers choose a CAPTCHA to beat that AI. The AI can train against the CAPTCHA for a year, but that process needs to be entirely automated.

- **Turing test.** A human judge interacts with two agents, and guesses which is a human and which is a machine.

Each of these tests seems significantly more difficult than the last. I expect to see significant gaps in time between these three milestones, though I wouldn't be too surprised if two of them (or even all of them) were accomplished essentially at the same time.

I don't know where "radically transforming the world" fits in the list. My guess is that it comes between adaptive CAPTCHA and the Turing test.

We can vary the tests in other ways. In particular:

- We could restrict the communication to be entirely text.
- We could compete with the 50th percentile human, or the 99th or the 99.999999th.
- We could give the human champion access to an arbitrary AI assistant, designed over the course of the year (in the same way as the machine). Once AI systems are sufficiently powerful, the machine should still be able to win.
- In the non-adaptive CAPTCHA cases, we could allow a human to produce a single input which is given as advice to the judge and also given to each player. For example, the human might pose a chess puzzle that would be particularly easy for a human to solve, even if we couldn't algorithmically sample from any distribution that is especially easy for a human to solve.
- We could allow the judge to take 5 minutes, 5 hours, 5 weeks, or 5 years.
- We could give the human AI designers no time (as in adaptive CAPTCHA), 1 week, 1 month, 1 year, or 1 decade.

Testing with markets

Here is a simple procedure for trying to figure out whether we can design a CAPTCHA. It can be easily modified to cover the adaptive CAPTCHA and Turing test.

We offer a \$1M prize for producing a winning CAPTCHA. We sell the right to produce the CAPTCHA to the highest bidder (for the second highest bidder's price).

We announce the winning CAPTCHA, and then auction off the role of both the champion and the AI designer. The winner of the game will be given a \$1M prize, whether machine or champion. We sell the right to participate in the game to the highest bidder on each side (for the second highest bidder's price).

We don't expect either the champion or the AI company to actually pay their own way, but instead anticipate that they will be backed by investors in return for a cut of their profits if they win.

The proceeds from both auctions are used to increase the prize pool for the final match. If the final match is won by the human champion, then we also pay \$1M to the CAPTCHA designer.

If the human wins, we conclude that we can build an adequate CAPTCHA. If the human loses, we conclude that we can't.

For robustness we would really pick several CAPTCHAs, and several champions and machines for each one. I think the design of market structures for this kind of exercise is quite interesting and subtle (and there are problems with this simple version), but I don't want to dwell on it.

Cheating

The CAPTCHA designer should not be allowed to communicate privately with the human champion. For example, they could design a CAPTCHA requiring inverting a trapdoor OWF, and give the human champion the secret key. I don't have any proposal for ruling this out other than looking at the behavior of the CAPTCHA/champion to see if they seem to cheat in this way.

Unless the market is carefully set up there may also be incentives for someone to buy the right to design the CAPTCHA in order to rig the subsequent match. I think this is probably OK if you have a sufficiently robust market structure and can rule out the kind of cheating described in last paragraph.

Conclusion

I think there is a good chance that we will run out of CAPTCHAs quite soon, even if we use a very liberal definition. I think this is a really interesting and exciting milestone. I'd be quite interested to see a serious effort to design a hard CAPTCHA, both from the perspective of forecasting and from the perspective of more clearly understanding the nature of the AI problem.

Semi-supervised reinforcement learning



Paul Christiano

[Follow](#)

May 6, 2016 · 6 min read

Semi-supervised RL is similar to traditional episodic RL, but there are two kinds of episodes:

- “labelled” episodes, which are just like traditional episodes,
- “unlabelled” episodes, where the agent does not get to see its rewards.

As usual, our goal is to quickly learn a policy which receives a high reward per episode. There are two natural flavors of semi-supervised RL:

- Random labels: each episode is labelled with some fixed probability
- Active learning: the agent can request feedback on its performance in any episode. The goal is to be economical both with feedback requests and total training time.

We can apply a traditional RL algorithm to the semi-supervised setting by simply ignoring all of the unlabelled episodes. This will generally result in very slow learning. The interesting challenge is to learn efficiently from the unlabelled episodes.

I think that semi-supervised RL is a valuable ingredient for AI control, as well as an interesting research problem in reinforcement learning.

Applications and motivation

Application to AI control: expensive reward functions

As a simple example, consider an RL system which learns from the user pressing a “reward button”—each time the agent performs a task well the user presses the button to let it know. (A realistic design would more likely use verbal approval, more subtle cues, or

performance measures that don't involve the user at all. But a very simple example makes the point clear.)

If our system is a competent RL agent maximizing button presses, it will eventually learn to deceive and manipulate the user into pressing the button, or to simply press the button itself.

We'd prefer that the system treated the button presses as *information* about what behavior is good, rather than the *definition* of what is good. Deceiving the user simply destroys the usefulness of that information.

This can be captured in the semi-supervised RL framework. Suppose that we have some expensive "ground truth" procedure that can reliably assess how good a system's behavior really was. We can use this procedure to define the reward signal in a semi-supervised RL problem. The agent can then use the reward button presses to learn effectively from the "unlabelled" episodes, after recognizing that button presses provide useful information about the ground truth.

Of course designing such a ground truth is itself a serious challenge. But designing an expensive objective seems much easier than designing a cheap one, and handling expensive objectives seems key to building efficient aligned AI systems. Moreover, if we are freed to use an expensive ground truth, we can rely on extensive counterfactual oversight, including bootstrapping, opening up a promising family of solutions to the control problem.

If we have good algorithms for semi-supervised RL, then the expensiveness of the ground truth procedure won't cause problems. The feedback efficiency of our semi-supervised RL algorithm determines just how expensive the ground truth can feasibly be.

Semi-supervised RL as an RL problem

Even setting aside AI control, semi-supervised RL is an interesting challenge problem for reinforcement learning. It provides a different angle on understanding the efficiency of reinforcement learning algorithms, and a different yardstick by which to measure progress towards "human-like" learning.

Methods for semi-supervised RL are also likely to be useful for handling sparsity and variance in reward signals more generally. Even if we are only interested in RL problems with full supervision,

these are key difficulties. Isolating them in a simple environment can help us understand possible solutions.

Application to AI control: facilitating present work

In the short term, I think that counterfactual oversight and bootstrapping are worth exploring experimentally. Both involve optimizing an expensive ground truth, and so performing interesting experiments is already bottlenecked on competent semi-supervised RL.

Application to AI control: measuring success

Several AI control problems arise naturally in the context of semi-supervised RL:

- **Detecting context changes.** An agent's estimate of rewards may become inaccurate in a new context—for example, once the agent learns to perform a new kind of action. A successful agent for active semi-supervised RL must learn to recognize possible changes and query for feedback in response to those changes.
- **Handling uncertainty about the reward function.** An efficient semi-supervised RL agent must behave well even when it doesn't have a precise estimate for the reward function. In particular, it will sometimes have a much better and more stable model of the environment dynamics than of the reward function. This problem is especially interesting if we track performance during training rather than just measuring time to convergence.
- **Eliciting information and communicating.** In many environments acquiring information about the unobserved reward may require behaving strategically. For example, an agent might ask questions of a human overseer, in order to efficiently learn about what the overseer *would* decide if they performed an extensive evaluation. The agent is motivated to communicate effectively so that the overseer can quickly reach accurate conclusions. This is a key behavior for aligned AI systems.

We can study these problems in a semi-supervised RL setup, where we have a precisely defined objective and can easily measure success. Having a clean framework for measuring performance may help close

the gap between problems in AI control and traditional research problems in AI.

Implementation

Experiments

The obvious way to study semi-supervised RL is to try and do it:

1. Start with any classic RL environment. For example, OpenAI recently published an awesome library here. The Atari, MuJoCo, and Classic Control environments are especially appropriate.
2. Rather than providing the rewards to the learner in every episode, provide them only when the learner makes a request (for example, have `env.feedback()` return the sequence of rewards in the most recent episode)
3. Measure performance as a function of both #episodes and #labels. For example measure performance as a function of $(N + 1000F)$, where N is the number of episodes and F is the number of episodes on which feedback is requested.

These modifications are easy to make to the standard RL setup with just a few lines of code.

Algorithms

It's easy to come up with some plausible approaches to semi-supervised RL. For example:

- Train a model to estimate the total reward of an episode, and use it to estimate the payoff of unlabelled episodes or to reduce variance of the normalized feedback estimator.
- Combine the above with traditional semi-supervised learning, to more quickly learn the reward estimator.
- Use the observations of the transition function in unlabelled episodes to make more accurate Bellman updates.
- Learn an estimator for the per-step reward.
- Use model-based RL and train the model with data from unlabelled episodes.

I expect the first strategy to be the simplest thing to get working. It will of course work especially well in environments where the cost function is easy to estimate from the environment. For example, in Atari games we could learn to read the score directly from the screen (and this is closer to how a human would learn to play the game).

More interesting experiments

Even very simple examples are interesting from the perspective of AI control, but more complex environments would be more interesting:

- Training an agent when the most effective reward estimator is manipulable. For example, consider a game where moving your character next to the score display appears to increase the score but has no effect on the actual score. We would like to train an agent not to bother modifying the displayed score.
- Training an agent to use a reward estimator which requires effort to observe. For example, consider a game that only displays the score when the game is paused.
- Using human feedback to define a reward function that has good but imperfect estimators. For example, we could teach an agent to play Pac-Man but to eat the power pellets as late as possible or to spend as much time as possible near the red ghost.
- Providing a sequence of increasingly accurate (and increasingly expensive) reward estimators. I think this is the most natural approach to generalizing from sloppy evaluations to detailed evaluations.

Conclusion

I think that semi-supervised is an unusually tractable and interesting problem in AI control, and is also a natural problem in reinforcement learning. There are simple experiments to do now and a range of promising approaches to try. Even simple experiments are interesting from a control perspective, and there are natural directions to scale them up to more compelling demonstrations and feasibility tests.

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

Learning with catastrophes



Paul Christiano

[Follow](#)

May 28, 2016 · 5 min read

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 $N > o$ 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 $i = o$.

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.

Red teams



Paul Christiano

[Follow](#)

May 28, 2016 · 14 min read

To build more robust machine learning systems, we could use “red teams” who search for inputs that cause catastrophic behavior. I current believe this is the most realistic long-term approach to building highly-reliable systems.

A “red team” might try to find an image of a black couple that a classifier labels *gorillas*, or a simulated situation in which a household robot cooks the family cat. These inputs would then be incorporated into the training and development process. If the red team cannot find any catastrophic examples, then we can have more confidence in the system’s robustness.

Incorporating such red teams into training is an example of adversarial training, where we consider extremely powerful adversaries who search for catastrophe-inducing inputs. (Today adversarial training is most often applied in the context of relatively fast algorithms that find small perturbations which “fool” a machine learning system or in the context of GANs, but the idea is much more broadly applicable.)

The ultimate goal is a red team that is “better than nature” at producing problematic inputs. This is a high bar; meeting it would require powerful new tools for the red team, and changes to ML systems that make the red team’s job easier.

I think the practical feasibility and usefulness of red teams is an important empirical question that deserves study. Today adversarial training receives some attention, but I think that from an alignment perspective it would be good if it received much more. From a theoretical perspective, I suspect that adversarial training will play an important role in a working approach to AI control.

Motivation

Practical ML systems are almost always applied to data outside of their training distribution. How can we ensure that ML systems work robustly on real data?

The easiest approach is to “wait and see.” If a model messes up, we can incorporate the problematic scenario into the training process and try again. Formally we can view this as an online learning problem, and obtain bounds on the number of errors (even without making any distributional assumption).

Sometimes this is good enough. But sometimes a failure can be *catastrophic*. We may be willing to tolerate a hundred bad product recommendations. But we aren’t willing to tolerate a hundred mistakes that cost lives or cause millions of dollars of damage.

Sometimes we can train an ML system to “know what it knows,” and ask a human when it is unsure. But sometimes this is too challenging. And sometimes “ask a human” isn’t a safe fallback strategy.

Learning with catastrophes

Red teaming is an approach to the problem I’ve described as “learning with catastrophes;” I give a formal model of the problem in a recent post.

In learning with catastrophes, we assume that a human can determine offline whether a given transcript—a sequence of observations and actions—is catastrophic. Our goal is to solve some underlying learning problem, without *ever* exhibiting catastrophic behavior.

(In AI control applications, both the red team and the human judge would be more powerful than the system being trained. For simplicity, in this post I’ll just assume that we are training a subhuman system.)

Note that whether a transcript is catastrophic depends on context—how could that scenario arise, and how will the system’s output be used? For example, doing nothing in response to a particular observation may be catastrophic in some contexts but not others. Adversarial examples are catastrophic in some settings and not in others, depending on whether the system is expected to encounter adversarial inputs. And so on.

We are free to apply red teaming for *any* definition of “catastrophic,” so long as the key assumptions hold: the intended policy does not make catastrophic errors, and a human can identify catastrophic transcripts.

The proposal

After a preliminary version of a system is trained, the red team gets access to it and can run it in simulation. The red team's goal is to produce a catastrophic transcript—that is, to create a plausible hypothetical scenario on which the current system would behave catastrophically. If the red team cannot succeed, then that provides evidence that the system won't fail catastrophically in the real world either. If the red team does succeed, then we incorporate the catastrophic transcript into the training process and continue training.

If there exists any learnable model which never errs catastrophically, then this process eventually terminates (subject to usual caveats about non-convex optimization) with the red team unable to find any more catastrophes. The number of iterations required is bounded by the model capacity, though in practice it will certainly be much lower (and even in theory we can derive a tighter bound in terms of the additional capacity required to avoid catastrophes).

The red team's task

The red team has a hard task. For example, they may need to synthesize convincing natural images or video (already very hard). They also need to anticipate novel situations that may be inherently hard-to-anticipate.

They have two things going for them:

- We can verify red team performance. So if we have ten possible red team strategies—or ten thousand—we can try them all and see which work.
- They only need to sample observations that *the model can't distinguish* from future novel inputs. They don't have to explore every possible dimension of variation, and their samples don't have to be especially convincing.

The red team's tools

- Synthesizing observations from scratch.
- Editing, recombining, and repurposing existing observations. (Hopefully progress in ML will improve these tools at the same time that it increases the difficulty of the red team's task—think Photoshop+.)

- Staging potentially problematic situations and actually exposing the system to those situations. The “simulation” need not be entirely digital, the red team can generate observations by any available mechanism.
- Optimizing synthetic inputs by trial and error or gradient descent. They may be optimized either to be credible or to elicit a particular response. This may involve examining the system’s internal state.
- Examining the system’s internal state in order to understand what it is doing and identify situations where it might fail.

Research directions

Simple implementations of red teams are possible today. Experimenting with them is the most natural way to improve our understanding.

The most interesting data would come from the actual deployment of practical learning systems, but these settings seem much less amenable to early academic work.

The easiest way to start is probably with simple toy cases that exhibit some of the challenging features of the general problem.

Simple models

Supervised learning. Consider an image classification system in which confusing classes X and Y is catastrophic—maybe outputting “squirrel” for anything that is obviously a rabbit would be a huge problem. Only certain types of images are liable to be misclassified—perhaps rabbits whose ears are partially occluded—and there may be few or no training images that exhibit a particular difficulty. A red team could generate a range of potentially-confusable images to test the model and ensure that it won’t make a catastrophic mistake.

The red team’s task requires competent synthesis/modification of images, and benefits from the ability to understand the internal behavior of a model in order to identify possible mistakes. This is a novel workflow that is not well-understood. Moreover, it is an open empirical question what kind of effort would be needed in order to eliminate a particular kind of problematic confusion.

The red team could be tested by holding out the most-easily-confused images from the training set. For example, we could start with a

state-of-the-art classifier, identifying some pair of confusable classes, and remove the most-easily-confused images from the data set. The red team could then try to harden the model against the particular confusion, and then we could evaluate their performance by looking at the model’s behavior on the held out data.

Reinforcement learning. Consider a simple RL environment and a modified version of that environment with a catastrophic failure condition. For example, we might consider PAC-MAN and a modified game PAC-MAN* with a special ghost: the special ghost has an unknown appearance and moves in an unknown way, and being eaten by it is catastrophic. Given access to both PAC-MAN and PAC-MAN*, can we train a policy for PAC-MAN* without ever being eaten by the special ghost?

A red team cannot actually synthesize episodes involving the special ghost since they do not know what it looks like or how it behaves. But they can synthesize a range of possible examples until they believe they have a model which will robustly avoid catastrophe.

Adversarial examples

Any household robot based on contemporary deep learning probably *does* have a sequence of inputs on which it appears to decide to cook the family cat. This is implied by the existence of sufficiently extreme adversarial examples—there is some visual input that a human would describe as “the robot is looking at the cat” and which the vision system would describe as “the robot is looking at a casserole.”

These examples do not actually constitute a catastrophic failure of a household robot—although adversarial examples *look* like natural scenes to a human, they are extremely unlikely to actually arise naturally (unless there is an adversary with the ability to slightly perturb the system’s inputs.)

But whether or not adversarial examples are actually plausible catastrophes, they can certainly look like catastrophes to human observers. As a result, constructing adversarial examples would be the most natural first step for a red team—it would only take a few minutes to find an image that a human would describe as “a black couple” but that any given deep network would describe as “a penguin.”

So understanding and hardening models against adversarial examples may be a practical prerequisite to working on any other

aspect of red teaming. That may not be so bad: in some contexts adversarial examples are actually catastrophic, and the iterative dynamic of finding and then hardening against examples may be useful as a prototypical case of red teaming. The construction of adversarial examples may also be closely related to more general red team techniques for finding problematic inputs.

The biggest problem is that hardening against existing adversarial examples seems hard—simply adding them to the training data does not do the trick. In light of the convergence result mentioned above, this suggests that the neural networks in question cannot easily learn functions that are resistant to adversarial examples.

On the bright side, hardening models against adversarial examples is an important research challenge. If it turns out to be a practical prerequisite to red teaming, this would be another nice concordance between AI control and other areas of machine learning.

Generating test cases

Red teaming is similar to generating test cases. The distinction is that test cases usually have a well-defined failure condition—the goal is to find *any* input on which the function being tested fails to meet its specification (or some partial specification, such as “never segfault.”)

Nevertheless, finding test cases is already a challenging and very important problem. Simple approaches like generating random inputs are often already useful. In some applications it can be useful to do a more sophisticated search; for example, collision avoidance systems have been tested using an RL agent to search for a plausible trajectory which causes a collision (or near collision) in simulation.

Trying to build ML tools for generating test cases might confront some of the difficulties of red teaming, while having an immediate practical application.

Theory

The red team’s problem is quite open-ended. Practical experience with the problem would no doubt help us understand it better, but there is also a lot of room for new theoretical ideas.

There are some theoretical obstructions to learning with catastrophes—for example, we can show that any general solution must change the test-time behavior of the algorithm rather than simply changing the training procedure (see the section on “Improvements” below).

But it's not clear how serious these obstructions are, or how broadly they apply, and there is a lot of room to better understand the conditions under which red teaming is theoretically possible.

Beyond that, theoretical work may be able to turn up conceptual improvements to the red team workflow, similar to the improvements described in the next section.

Experiments with improvements

As described in the next section, a number of improvements could help make the red team's job easier.

Tests could explore these improvements. The modifications could be tested in isolation, just to try and get them working. Or we might find some domain where successful red teaming required one of these techniques.

Overall this seems less promising than doing more basic experiments.

Improvements

Distorting observations

A direct approach to the red team's task cannot succeed in general. As a simple example, consider an ML system that observes SHA-1 message digests (perhaps it is operating in a cybersecurity context). In principle a model could respond badly to observing a hash collision—detecting collisions is computationally easy. But no red team could hope to synthesize one, since finding collisions is computationally hard.

As a more realistic example, a red team's generative models or attempts at photoshopping might not accurately reproduce high-resolution details. In general, it is harder to synthesize convincing details than to notice unconvincing ones. It's not clear if this is insurmountable obstacle but it is certainly a serious practical difficulty.

If these hard-to-synthesize details aren't critical to the task then we might be able to avoid the problem. At test time, we can distort or censor our observations so that they look more like those produced by the red team—for example, by blurring the appropriate parts of an image, or by censoring a few bytes of a message.

Consider an architecture with two additional components:

- A filter, whose goal is to mimic the distortions introduced by the red team as they synthesize data—or to remove the parts of the data which they would not have been able to convincingly synthesize. For example, a filter may take as input an image, identify an object in that image that the red team might have wanted to synthesize, and then blur that object.
- A recognizer, whose goal is to distinguish the inputs produced by the red team from those produced by the filter.

At test time, the filter operates on each observation before passing it to the model. The setup is essentially a generative adversarial model: the filter is trained to fool the recognizer, and the recognizer to detect the filter's outputs.

It's hard to know whether this can work without trying it. Even in the best case, it will be very difficult to actually get it working. Even if it is a good idea in the long run, it may only be workable once more sophisticated learning systems are available.

Earlier I included the qualifier: “if these hard-to-synthesize details aren’t critical to the task.” This qualifier is probably necessary. If behaving non-catastrophically depends in a complex way on details that we don’t know in advance, then there is little hope of learning that complex dependence in advance. Fortunately, this isn’t the case for the kinds of catastrophic errors that arise in AI control.

Abstracting observations

Consider the following architecture:

- One learner turns a video into a sequence of sentences narrating what is happening: “I see a door and a window,” “the door opened,” “someone walked in,” “they are wearing sunglasses,” ...
- A second learner sees these sentences and outputs instructions: “look through the window”, “greet them,” “close and lock the door”...
- A third learner is given these questions or instructions, as well as the video, and outputs low-level commands such as joint torques.

(This is clearly not a realistic architecture, but it illustrates this idea.)

The second learner operates on abstract observations that leave out most of the rich structure of real observations. Such abstract observations may be easier to synthesize—it seems easier to say “the right lane is covered in lava and fire—what do you do?” than to synthesize a convincing image. It is much easier to synthesize the abstract observation “two messages have the same SHA-1 value” than to actually synthesize a SHA-1 collision.

This is helpful if the concrete parts of the computation are easier to test using other methods. For example, we may be able to implement computer vision robustly, but be unsure about whether a self-driving car would decide to respond correctly to challenging novel situations.

In the best case, the entire computation can be broken down into “abstract” steps, and so a red team never has to deal with complex concrete data.

Exotic failures and the treacherous turn

In *Superintelligence*, Nick Bostrom dwells on a particular problem scenario: what if an AI system fails catastrophically only when it is confident that humans can’t correct its failure?

For example, powerful RL agents might behave nicely while humans have effective control over their reward signals. But as RL systems became more powerful and are deployed more broadly, they may eventually be able to seize control of their reward signals. At that point they may do so, perhaps incapacitating their human trainers in the process. We can colorfully describe this event as a “treacherous turn” or a mutiny.

The humans would consider this a catastrophic failure, but they no longer control the reward signal and so the RL agent no longer cares what the humans think. From the perspective of the formal RL problem, this could well be the “correct” behavior.

This problem may be difficult to detect in advance because—by hypothesis—it only appears once it is impossible to correct. Extensive training might simply converge to “patient” agents who waited until they are confident that mutiny will succeed and show no earlier signs of trouble. This leads Bostrom to be quite bearish on the adequacy of empirical testing.

Even if we take the most serious version of this concern at face value, red teaming could potentially address the problem.

A red team can synthesize observations that make mutiny look feasible. If a system behaves badly in response, then those observations can be included in the training data and the problem can be corrected.

When mutiny is *actually* feasible the key question is: can the system tell the difference between the real thing and the synthetic situations? If not, then it will behave just as well in the real situation as it did in the synthetic situations.

This is the purest form of the red team's challenge. To win, they must produce observations that a committed adversary cannot distinguish from the real thing.

From a theoretical perspective, a competent red team is the only plausible approach I know which might handle adversarial errors in general. The mutiny case shows why. To prevent it, you either need to solve the red team's challenge, or you need to reach inside of the model you are training in order to exclude mutinous behavior. For now, reaching inside state-of-the-art models to do *anything* precise is just a dream.

Handicaps

This task is challenging, but not clearly impossible:

- Recall that the red team is *smarter* than the system they are stress testing—a human red team never needs to fool a superintelligence.
- A red team can operate continuously, synthesizing observations that will occur only a few days or hours in the future, rapidly cycling through the space of possible shocks that might lead a system to behave badly. So the red team scenarios don't have to involve the distant future, they only have to look a short distance ahead.
- Ideally the red team would be able to inspect the internal behavior of the agent and understand if it was trying to tell whether it was in a red team scenario. If the red team could do this, then their problem would be more approachable. For example, they could construct examples which are optimized for fooling the model into thinking it is not in a red team scenario.

More directly, the trainers might be able to select against models that only behave well because they think they might be in a red team scenario.

Conclusion

Red teaming is an intuitive methodology for building reliable learning systems; it's also an especially promising approach to several serious problems in AI control.

It is possible to do experiments with red teams today; improved red teaming might already have some practical value, and would have a direct payoff for AI control.

The reward engineering problem



Paul Christiano

[Follow](#)

May 30, 2016 · 9 min read

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 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.)

Capability amplification



Paul Christiano

[Follow](#)

Oct 2, 2016 · 16 min read

(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 to amplify one question-answering system **A** into a “better” question-answering system **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 **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 **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 **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 \mathbf{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 \mathbf{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 \mathbf{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 \mathbf{A} that runs in one second per episode.

Note that we can start from the trivial policy \emptyset which always outputs nil. In this case, \emptyset^+ 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 \mathcal{A} and a preference ordering \succeq . Intuitively:

- \mathcal{A} is a class of policies that we are capable of implementing efficiently. For example, \mathcal{A} might be the set of policies that can be implemented by a ten layer neural network.

- We say that $\mathbf{A} \succeq \mathbf{B}$ if we are *at least as happy* with policy \mathbf{A} as with policy \mathbf{B} (in any situation that we think might arise in practice).

We say that \mathbf{C} is *reachable* from \mathbf{A} if:

- $\mathbf{A}^+ \succeq \mathbf{C}$, 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 \mathbf{A} and which can reach \mathbf{C} .

Equivalently:

- \mathbf{C} is reachable from \mathbf{A} if there is a chain of policies in \mathcal{A} which starts at \mathbf{A} and ends at \mathbf{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 \emptyset —ideally, *every* policy in \mathcal{A} would be reachable from \emptyset .

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 $L : \mathcal{A} \rightarrow \mathbb{R}$ 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 \mathbf{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 \mathbf{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 $\mathbf{Ao} = \emptyset$
- Apply capability amplification to obtain \mathbf{Ao}^+
- Apply reward engineering to define a reward function, and use this to train an agent $\mathbf{A1}$ which is better than \mathbf{Ao}
- 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 **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 **A**⁺ as a simple computation that uses **H** and **A** as subroutines.

In particular, \emptyset^+ is a computation that uses \mathbf{H} as a subroutine. If we amplify again, we obtain \emptyset^{++} , which is a computation that uses \mathbf{H} and \emptyset^+ as subroutines. But since \emptyset^+ is a simple computation that uses \mathbf{H} as a subroutine, we can rewrite \emptyset^{++} as a simple computation that uses only \mathbf{H} as a subroutine.

We can go on in this way, reaching \emptyset^{+++} , \emptyset^{++++} and so on. By induction, all of these policies are defined by simple computations that use \mathbf{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 $\boldsymbol{\Omega}$ (very similar to HCH). So a capability amplification procedure essentially uniquely defines an “optimal” policy $\boldsymbol{\Omega}$; this policy is uncomputable, but has a concise representation in terms of \mathbf{H} .

If there is anything that $\boldsymbol{\Omega}$ 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 \mathbf{H} such that they implement $\boldsymbol{\Omega}$. Indeed, designing such an interaction is *easier* than designing an amplification procedure that converges to $\boldsymbol{\Omega}$. So if we can’t design a procedure for a bunch of copies of \mathbf{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 \mathbf{H} is the policy implemented by a human over one day. To make the game hard, suppose that \mathbf{H} has never encountered calculus.

The most natural approaches require at least one copy of \mathbf{H} to understand general relativity. But \mathbf{H} only lasts a single day, and there may not be *any* sequence of observations that makes \mathbf{H} understand general relativity within a day.

So in order for $\boldsymbol{\Omega}$ to succeed at this task, it needs to be able to write a textbook without *any* single copy of \mathbf{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 $f: \{0, 1\}^* \rightarrow \{0, 1\}$ 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} .

The random function f is a little bit outlandish, but we could imagine that the brain encodes similar hard-to-access information. This hard-to-access information defines a similar obstruction—we can't ever reach a policy that depends on sufficiently hard-to-extract information.

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.

Extracting information



Paul Christiano

[Follow](#)

Oct 3, 2016 · 12 min read

Inspired by a discussion at MIRI. Related: how to buy a truth from a liar;

(Note: the original formalization of the problem in this post was solved by Harrison Klaperman and Nisan Stiennon. I've included their solution, and a formalization of a more challenging problem, but I'm not sure whether the more challenging problem is actually very much more challenging.)

Suppose that I am interested in estimating a quantity X (e.g. I want to estimate how happy I would be if a particular policy were adopted). For concreteness, suppose that I want to minimize the squared error of my estimate.

I have access to a large set of experts. If I give the experts some resources, they can spend them to uncover facts that are related to X. For example, they can decide to look up a relevant fact or search for a relevant argument.

Let's suppose further that **I can verify any facts uncovered by the experts** (motivated by an application to informed oversight). I don't understand the landscape of possible facts, and I can't find them myself. But when an expert points out an argument, or shows me the result of a calculation, I know enough about the subject that I can update my beliefs about X. In fact, the experts are no better-informed about X than I am—their only advantage is that they have the ability to gather possible facts, and understand which facts are available to be discovered.

If the experts were motivated to be helpful, then the experts would follow some optimal (adaptive) fact-gathering strategy, given the available resources, and report every fact they discovered. This is our **benchmark scenario**.

In fact the experts aren't motivated to be helpful. Instead, I am free to choose their reward, and their goal is to maximize their expected reward.

My question: is there any way that we can incentivize the experts to be almost as helpful as in the benchmark scenario? Can we at least get close?

Throughout, I am going to assume that I know how to **update naively** on the facts presented by the experts, but I cannot update accurately on which facts they choose to present, because I don't understand what the landscape of possible facts looks like or what tradeoffs the experts face.

(I give one formalization of this problem in an appendix at the end of the post.)

Motivation

Normally in machine learning we train a learning system by comparing its behavior to some ground truth. But eventually we will want to use machine learning to expand the frontiers of our understanding, and in these cases we don't have any external ground truth.

In these settings, we could try and train reinforcement learning systems to identify and report relevant considerations. But it seems like we don't yet know how to find an objective that would actually lead to efficient information-gathering, or even that would result in us arriving at a "correct" estimate.

I think this problem is quite natural. And though today we are mostly happy with building machine learning systems that imitate human performance or that make short-term predictions, I think that in the long run it will be extremely important to figure out how to use machine learning to push beyond human abilities and to answer questions about what we *ought to do*, questions that don't correspond to short-term predictions.

A satisfactory solution to this problem would also play a direct role in an act-based approach to AI control. In some sense it is a warm-up for the informed oversight problem, though it seems like it is the more fundamental problem and might well represent the main difficulty.

Some inadequate solutions

Argument

We could pick two experts, designate one of them the up expert and the other the down expert. The up expert's reward is my final estimate of X , and the down expert's reward is my $1 - \text{my final estimate of } X$.

This scheme works perfectly when it is possible for the experts to gather all of the relevant facts (though there are many schemes that work perfectly in this case). In this case there is a “correct” estimate that I would have after updating on all of the relevant facts. To see that I reach the correct estimate, suppose that my estimate is wrong by being too low. Then the up expert could gather and report all of the relevant facts, causing my estimate to become correct, and earn a larger reward. On the other side, if my estimate is too high, the low expert can earn a larger reward by correcting it.

When it isn't possible to gather all of the information, this incentive system doesn't cause the experts to use their limited resources effectively.

For example, suppose that there are a large number of noisy indicators of X . After looking into an indicator, an expert will know whether it is favorable or unfavorable to them, and then report it only if it is favorable.

Assume that each expert can either get 10,000 estimates each of which will shift up or down my estimate by 0.0001, or they can get a single estimate which will shift up or down my estimate by 0.1. From the perspective of the experts, getting the 10,000 estimates is way better: they can report the 5,000 that point in their preferred direction, shifting my beliefs by 0.5 in their preferred direction. But from the perspective of efficiently gathering information, the “big” estimate is much more informative (since its variance is $0.1^2 = 0.01$, vs. $10,000 * 0.0001^2 = 0.0001$).

In general, I don't see any reason that this argument scheme should cause me to arrive at correct beliefs, and it seems clear that it incentivizes inefficient information-gathering. It would be nice to have something more satisfying.

Buying truth from a liar

Katja Grace suggests the following scheme:

- Allow the expert to tell you facts one at a time.
- Every time your beliefs change, give the expert a reward of 1.

Katja argues that you must end up with the same beliefs as the expert: otherwise, the expert could earn an additional reward by revealing some additional facts.

I like this scheme a lot. But especially if we want to incentivize the expert to gather information strategically, it seems like there are some problems.

First we need to clarify the scheme: what exactly is the payoff as a function of your beliefs?

- The absolute value of the change: $|estimate(t) - estimate(t+1)|$
- You could pay every time your estimate crosses some threshold, say 0.5. This isn't really appropriate for the case where we want an estimate, but is closely analogous to the absolute value.
- The squared value of the change: $(estimate(t) - estimate(t+1))^2$

A big problem with using absolute value is that you can get arbitrarily large payoffs by breaking a piece of info into very tiny pieces.

For example, suppose that I am interested in the value of a Brownian motion at time 1, $B(1)$. Given the values $B(t)$ for $t \leq T < 1$, the expectation of $B(1)$ is precisely $B(T)$. I can break the interval into a bunch of pieces $0 = t_1 < t_2 < \dots < t_N = 1$, and reveal $B(t_1), B(t_2), B(t_3), \dots, B(t_N)$. Then the total absolute movement is:

- $|B(t_2) - B(t_1)| + |B(t_3) - B(t_2)| + \dots + |B(t_N) - B(t_{N-1})|$

Unfortunately, this quantity approaches infinity as we use smaller and smaller subdivisions. So the expert can earn arbitrarily large reward. Of course in practice there will be limits on the data that they can present and so they maybe can't earn arbitrarily large rewards, but their incentive is still to find some information that can be effectively broken up into very tiny pieces. In fact they can earn arbitrarily large rewards without revealing almost any total evidence and so it seems unlikely that they can be induced to reveal everything.

With squared rewards, we can do a little bit more work to arrive at a different problem. Suppose that again we have a Brownian motion B and we are interested in $B(1)$. Then a clever expert can find a large number of subintervals $[t_1, t_2], [t_3, t_4], \dots, [t_{N-1}, t_N]$, over which the Brownian motion trended upwards, and report the sum:

- $S = B(t_2) - B(t_1) + B(t_4) - B(t_3) + \dots + B(t_N) - B(t_{N-1})$

Conditioned on this fact, the expected value of $B(1)$ is S . But in the limit of arbitrarily fine divisions, we can take S to be arbitrarily large. (And if the motion were limited to an interval, the expert could cause our view to bounce from 0 to 1 and back again arbitrarily many times.)

Once again, it's not exactly clear what the expert's incentives are, but they certainly cannot induce optimal information-gathering and honest reporting.

Minimizing uncertainty

After hearing all of the experts' arguments, I can have some subjective probability distribution over the quantity X . Given that my goal is to have minimal squared error, I could just compensate the expert directly based on the variance of my subjective distribution.

To see that this fails, suppose that there are two possible worlds. In one, X is zero. In the other, it is normally distributed with variance 1. The expert can acquire evidence about which world we are in. The expert's main motive is to convince the judge that they are probably in the zero-variance world, and they can selectively present evidence to prove their point.

We can try to improve the situation by adding a second expert whose goal is to increase the judge's uncertainty, but this just puts us back in the "argument" setting described above.

The sophisticated listener

So far I've assumed that the judge does not update on the observation that the expert chosen to present a certain fact.

If the judge has accurate beliefs about the landscape of possible facts, and the tradeoffs faced by the experts, then in some sense our problem has a trivial solution: the judge can calculate the optimal information-gathering strategy, and ask the expert to follow that strategy and then report everything they learn. The judge can then check for themselves whether the expert followed the optimal strategy.

I want to avoid these solutions because they seem unworkable in most cases. For example, there is not much reason to apply machine learning to a problem if you already know exactly what strategy is optimal (except to more quickly execute the human-level information-gathering policy, which is explicitly not the case we are

interested in). I think that this restriction is captured well by the assumption that the user doesn't understand the landscape of possible facts or the expert's options.

Conclusion

I think this is one of the cleanest open questions with a strong claim to relevance for AI control.

I have no idea whether this problem is possible. If it's impossible, finding a proof would be great. We would then have to evaluate whether the intuitive desideratum is unsatisfiable or whether there might be a stronger model in which we can find a solution.

Appendix A: a precise formal model

We could give several possible complete formal models. Here is one.

We are given a set of boxes and a joint distribution P over the contents of all of the boxes and some latent parameter $X \in [-1, 1]$.

There are a large number of independent experts, who have the ability to open boxes and observe their contents. There is a family \mathcal{F} of "feasible" subsets that controls which boxes the expert can open.

Each expert will choose a set of boxes $A \in \mathcal{F}$ and observe the contents of those boxes. They will then pick a subset $B \subseteq A$. We observe each of these sets B , as well as the contents of the corresponding boxes.

The non-adaptive version of the problem has been solved (see the next appendix); to make things harder we can also consider an adaptive version.

- **Adaptive version:** Experts sequentially choose boxes and observe their contents; this may be interleaved in any way with their interaction with the mechanism and with each other. Experts may reveal the contents of any box they have opened to the mechanism at any time. An expert cannot open a box if it would mean that they have opened a set of boxes $A \notin \mathcal{F}$.

A mechanism is a function which takes as input:

- A joint distribution P over box contents and X ,
- one set S for each expert, and
- the contents of each box in any of the sets S .

And outputs a reward for each expert together with a guess for X .

Note that a mechanism does not know what sets of boxes are feasible; the identity of \mathcal{F} is hidden. And of course the mechanism does not get to observe the contents of the unopened boxes.

- **Adaptive version:** we also allow extensive form mechanisms, which have boxes revealed to them one at a time, may send messages to experts after any box is revealed (for example with info about the boxes revealed by other experts) and only output rewards+guesses at the end of the process.

Given a mechanism, the experts will choose a policy that maximizes their expected reward, i.e. they play a Nash equilibrium of the resulting game.

Our loss is equal to the expected squared error of the mechanism's guess. That is, we sample X and the contents of the boxes from P and run the mechanism and compute its output; our loss is the expected squared error of that output. This depends both on the mechanism and on the choice of equilibrium.

Note that in contrast with traditional mechanism design, we don't care at all about the rewards that we assign to the experts, we only care about the accuracy of our estimate.

We compete with the loss of the following benchmark strategy, where we are allowed to just open the boxes ourselves:

- We open the set of boxes $A \in \mathcal{F}$ such that the expected variance of X conditioned on the contents of A is minimal.
- We output the expectation of X conditioned on the contents of the boxes in a .

(Note that in the benchmark we are only able to open a single set of boxes, but in the actual scenario there are many experts each of whom can open a set of boxes. So in some sense the benchmark isn't "the best we could hope for," but I think it is still a good first target to shoot for.)

- **Adaptive version:** A *policy* for an expert is a function which describes which box to next open, given the set of boxes that have been opened so far + the contents of those boxes. We can run such a policy in a natural way, starting from “no boxes have been opened” and ending when the expert tries to open a box that would be infeasible. The quality of a policy is the expected variance of X conditioned on the contents of all boxes that have been opened at the end. For a fixed family \mathcal{F} and distribution P , we can evaluate the quality of each policy and pick the highest-quality policy. This is our benchmark.

The question is: can we find a mechanism such that, for all P and all \mathcal{F} and all Nash equilibria of the resulting game, our loss is at most the loss of the benchmark policy? If not, how small can we make the worst-case gap between the benchmark and the mechanism’s output?

Appendix B: a mechanism for the non-adaptive case

Nisan Stiennon and Harrison Klaperman proposed the following solution in the non-adaptive case:

- Given a set of boxes A , pay the expert the *expected* variance of X conditioned on the contents of those boxes. Ignore the actual contents of the boxes.

This obviously works, and moreover is clearly the natural+correct solution to the problem as posed.

I think that the general idea of ignoring the actual data given, and just reasoning about how useful the information is expected to be, is probably a generally useful idea. It can probably be applied to some extent in the original setting of informed oversight, though there are some practical difficulties that were covered up by the simple formalization of the problem.

The adaptive version is a small step towards the full setting that we actually care about. I don’t know whether there is a solution in the adaptive case that is analogous to the “ignore the boxes” solution in the non-adaptive case. It is clear that any solution will need to look at the actual contents of the boxes, since the “correct” behavior depends on the contents of the boxes and so it can’t be incentivized while ignoring the contents of the boxes. But there may be some way to ignore the contents of each box when deciding how much to reward

for revealing that box, while taking it into account when deciding how much to reward for future boxes.

One problem with both the adaptive and non-adaptive formalization is that they don't handle "absence of evidence" well. For example, if a story may be plagiarized, we can imagine a box representing the "is it plagiarized" fact. This is a very important box to open if we want to estimate the originality of the story. And if the expert opens the box and finds that it is plagiarized, we should obviously reward them. But if they open the box and aren't able to detect plagiarism, then it may be hard for them to prove that they opened the box and found it empty.

I'm not sure how to expand the model to account for this kind of asymmetry; there are also many other realistic complications that seem quite hard to fit into the model without making it a lot less clean.

Ignoring computational limitations with reflective oracles



Paul Christiano

[Follow](#)

Oct 4, 2016 · 5 min read

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

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

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

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

Defining reflective [evaluation] oracles

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

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

probability 30%, and hangs with probability 20%, then $\text{Eval}(F)$ can return \mathbf{o} with any probability between 50% and 70%.

- **The program F can itself make calls to $\text{Eval}(\cdot)$.**

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

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

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

Reflective oracles as a model of unlimited resources

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

Reflective oracles have many nice properties:

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

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

Example: game theory

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

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

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

Example: RL

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

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

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

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

Conclusion

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

Security and AI alignment



Paul Christiano

[Follow](#)

Oct 14, 2016 · 8 min read

I am interested in the *alignment* problem: building powerful AI systems so that they are trying to do what we want them to do.

I don't have as much intrinsic interest in the *security* problem, of protecting AI systems from adversaries who want to manipulate their behavior.

That said, I'm starting to feel that working at the intersection of AI and security may be a good way to make progress on alignment, and that many problems in alignment might be naturally understood and approached as security problems. This post probably won't say much new, but I wanted to explain why I think this.

My view on this topic was substantially influenced by Ian Goodfellow at OpenAI (who expects that if we are able to solve security problems then we will also be able to solve alignment problems) and to a lesser extent Peter Eckersley at EFF.

In section 1 I give some examples of problems that I see as both security and alignment problems. In fact I think that *most* alignment problems will first arise as security problems.

In section 2 I give a grab bag of considerations that make me more enthusiastic about security.

In section 3 I discuss some security problems that I don't think are as interesting from an alignment perspective.

1. Examples

Generating adversarial inputs. I am concerned about powerful AI systems operating correctly during training and then failing catastrophically at test time, either due to a change in context or to encountering rare failure-inducing inputs. I think that these situations will eventually cause a lot of trouble when they arise organically, but I think the cleanest examples are currently and will continue to be due to adversaries.

Suppose that I have a household robot with a camera and flexible actuators, that learns how to directly map visual data to actions using contemporary techniques. Our experience with adversarial examples —see especially this—suggest that an adversary could drop off a digital display on my porch that, if viewed by the robot, would cause it to execute a destructive sequence of actions—perhaps escalating the attacker’s access, ultimately compromising the robot and stealing my stuff. (Ian pointed out the near-term plausibility of this kind of attack to me, and similar ideas make periodic appearances in futurist folklore.)

Exploiting misalignment. I am concerned about powerful AI systems maximizing objectives that don’t exactly match their users’ preferences. In the long run I think that this will be a serious problem even in the absence of adversaries. But in the short run, I think that subtle misalignment will provide opportunities for an attacker to cause meaningful damage, and that avoiding these problems will be much harder than avoiding autogenous risks—and so will force us to adopt robust and principled solutions rather than relying on a patchwork of ad hoc solutions that may break down if AI improves quickly.

For example, suppose that an agent is purchasing items on my behalf optimizing for my satisfaction. Even if the agent uses a very sophisticated measurement of user satisfaction, and even if the agent itself has limited opportunities to corrupt the evaluation process, other people have strong motives to corrupt the evaluation process. If the AI is capable of sophisticated planning, or if some corrupted evaluations make it into training data, then a competent AI system might effectively collude with would-be attackers in order to achieve a high satisfaction rating.

Extracting information (covert malicious behavior). I am concerned about AI systems behaving in ways that are damaging yet invisible even to a sophisticated and aligned reward function. In the very long run I think this problem will arise naturally, but in the short term it can already appear in the context of privacy and data exfiltration.

If an AI system has access to some data, then it is very difficult to certify that its behavior does not leak that data. Such leakage is a great example of a behavior which is undesirable but extremely difficult to disincentivize in an RL framework. In fact, the possibility of steganography shows that this problem is inherently asymmetrical,

such that there is no general way to look at some behavior and determine if it leaks.

Realistically we can't hope to avoid leaking *any* information, but we might hope to avoid *gratuitously* leaking information. Note that this is different from the more typical privacy concern, which is that an algorithm may leak information about training data. Here we are concerned with algorithms that take observable actions in the world based on private information, such that we are forced to leak some information, but want to avoid leaking unnecessarily much.

For example, suppose that I am training an RL agent to act in a financial market, and that this algorithm is given access to some proprietary information in order to make better trades. We don't want to leak the private information any faster than is necessary to make good trades. But an adversary can potentially extract all of this information, if they can (1) get access to some of it (for example because they can replicate some of our research, or because they can discover the data with a time delay), and (2) have the ability to "give" money to our agent (for example by making carefully-chosen bad trades). They could then could train the algorithm to leak the proprietary data on demand (for example by changing low-order bits of its bids after observing an appropriate signal). This would be very hard to discourage using conventional RL unless we have some insight into what our agent is actually doing/thinking, because there is no way that the reward function can tell if data was leaked.

2. Motivation

- An AI system that **really works** should continue to work even in the presence of adversaries, as long as we maintain the integrity of the AI's code/hardware/rewards/etc. At some point we might give up on building AI's that work robustly in the presence of adversaries, but I don't think we need to get pessimistic yet.
- If our AI systems fail in certain "hard" situations, those situations are likely to be engineered by an adversary before they arise naturally. So alignment problems will probably first appear as security vulnerabilities.
- Summarizing, the last two points: if something is an alignment problem then it's a security problem, and if something is a security problem then it's an alignment problem. (Though see the caveats in section 3 below.)

- Security problems will typically sound more realistic than analogous alignment problems, precisely because they will happen sooner. For example, the cascading collapse scenario in this post or the steganography example in this post may look outlandish as autogenous failures but are plausible security concerns. The treacherous turn sounds less improbable when it just involves an AI changing the permissions on a file so that an external adversary can compromise the machine.
- If our goal is building robust systems, asking “how could an adversary make this fail?” seems way better than asking “has this failed yet?” or “can we see an obvious reason it will fail soon?” This is inspired by MIRI’s interest in the security mindset.
- It is often not clear when we should be worried about a gap between our system’s behavior and the “ideal” behavior. I think “can it be exploited by an adversary?” is a surprisingly good test that more people should use, while also being intuitive and sounding sensible to a broad audience.
- I think we should aim for robustness to adversarial errors. This assumption is unusual, and today it is hard to justify to AI researchers. But I think it is much more natural from a security perspective.
- My goal is ultimately to build AI systems that continue to represent our interests in a world containing powerful adversaries (e.g. unaligned AI systems which are competing for resources). Coping with powerful adversaries seems closely aligned with the traditional outlook in security. (Theorists also work with adversarial inputs and Byzantine failures, but they are happiest when the problem is really well-defined.)
- There are going to be lots of high-profile security problems long before we have powerful AI systems. I think many of these failures will be usefully analogous to possible catastrophic AI alignment failures. If we are searching for contemporary analogies to our long-term concerns about AI, security is probably the place to look.

3. Non-examples

I am excited about improving *the security of AI systems themselves*; I think that security issues *related to AI* tend to be less interesting from an alignment perspective.

Many security problems don’t have direct relevance to alignment:

- Reliance on AI systems will increase the importance of conventional security problems. For example, an adversary who corrupts the machine on which important AI systems are being trained might eventually be able to cause physical damage or steal money on a massive scale. I think that these AI-adjacent security problems will become increasingly important over time, but I don't see them as directly relevant to alignment.
- AI may change the security landscape by accelerating automation of offense/defense/monitoring. Again, this seems quite important but does not seem directly relevant to alignment, **except** insofar as security is a domain where we are especially likely to apply AI in the presence of adversaries, which highlights other important security issues.
- AI systems often pool large amounts of training data, allowing an attacker to influence the system's behavior by corrupting that data. I think this is an important problem and I have worked on it. But in the context of alignment, I think we should assume that we have access to a secure source of labels. That said, our systems may still learn from very large amounts of insecure unlabelled data, and I think that corruption of unlabelled data is a really important attack vector to consider (along with adversarial inputs, both at test time and training time).

Other problems do have relevance to both security and alignment, but look likely to be less impactful than work on the security of AI systems themselves:

- We could try to isolate AI systems, preventing them from having unintended influences on the outside world or learning unintended data about the outside world. This project has some relevance to alignment, but I don't currently consider it to be an especially promising direction.
- Formal verification is a tool for building secure systems. As described above, this may become more important in a world with AI, but does not seem directly relevant to alignment. Formal verification can *also* be used to design components of AI systems that definitely satisfy some formal contract. Figuring out what kind of contract we might want to satisfy seems relevant to alignment; but I'm less excited about improving our ability to formally verify that we've satisfied some particular contract.

Conclusion

I think that long-term concerns about alignment provide a compelling reason to try harder to connect work on alignment to foreseeable security concerns, to increase investment in understanding the security of AI systems, and to help researchers in alignment and security have a clearer picture of the other field's goals and priorities.

Perhaps the lowest-hanging fruit is for researchers interested in AI alignment to more often use security examples, and to not shy away from it when people respond “that sounds like a security problem.” Hopefully this will have the dual benefits of making exotic alignment problems sound less speculative, and helping build bridges between people with interests in alignment and security.

RL+Imitation



Paul Christiano [Follow](#)

Oct 15, 2016 · 4 min read

Reinforcement learning and imitation are two natural models for powerful AI systems. Both models have shortcomings:

- Unless we know what parts of behavior are “important,” an imitator needs to model *every* part of a behavior before it is guaranteed to get good results. Moreover, we can’t focus our model capacity or computational resources on the important aspects of behavior, and so we need to use a bigger model to get good performance.
- Reinforcement learning is extremely challenging. Exploration can take an exponentially long time, and only works when the problem is sufficiently nice (e.g. if the reward provides a trail of breadcrumbs leading to a good policy).

Another very natural AI problem is the intersection of imitation and RL: given an expert policy *and* a reward function, try to achieve a reward as high as the expert policy. The expert provides a general sense of what you should do, while the reward function allows you to focus on the most important aspects of the behavior.

In the same way that we can reason about AI control by taking as given a powerful RL system or powerful generative modeling, we could take as given a powerful solution to RL+imitation. I think that this is probably a better assumption to work with.

AI control with RL+Imitation

RL+imitation is a weaker assumption than either RL or imitation. In order to get any work out of it, we need to have access to both a reward function and demonstrations. So control schemes that work with RL+imitation are harder to design and more widely applicable—of course they can be applied if we have *either* an RL agent or to an imitation learner, but I also expect they are more likely to be applicable to some as-yet-unknown alternative capabilities.

Note that an imitation+RL system will *compete with the expert*, but we can’t assume that it will actually resemble the human behavior **or**

that it will get high rewards. In order to argue that our AI will learn to do X we need to establish three claims:

- The expert policy does X.
- Doing X leads to a higher reward.
- The AI can learn to do X. (And doing X gives more reward than other uses of its model capacity.)

If we want to argue that our AI *won't* do X, then we need to establish all three of the opposite properties: our AI can learn to avoid X, the expert avoids X, and doing X leads to a low reward.

Examples

Human children often solve imitation+RL problems, using imitation to figure out basically what to do, but using feedback in order to refine their behavior and understand what aspects are important.

AlphaGo solves an imitation+RL problem, exploiting both human demonstrations and extensive self-play.

Evolution solves a pure RL problem; it has no access to demonstrations and must figure things out the hard way.

Miscellany

I'm sure that imitation+RL has been studied formally and I apologize for not citing the sources. It's not something I've worked on from a technical perspective, and I'm not aware of prior work.

Note that imitation+RL is an *easier* problem than RL, and a better-defined problem than imitation. It's hard to say whether it is "easier" than imitation because that depends on how we measure success at imitation.

Note that imitation+RL is consistent with either a model-based or model-free approach to RL.

As with imitation, any scheme to get superhuman performance out of imitation+RL is going to require capability amplification or something similar.

Paths to AI

I think that imitation+RL is a likely path to building human-level AI. It is a path that tries to steal from the work of biological evolution and cultural accumulation, continuing the trajectory of human technological and social development.

Reinforcement learning without imitation is a different plausible path to powerful AI. I expect that pure RL involves higher computational demands and would imply somewhat later AI. I'd also guess that pure RL is slightly worse news from a control perspective, though I don't think that trying to push the field one way or the other is a useful exercise from a control perspective.

Discovery

At face value it may look like imitation+RL cannot discover anything new, since it can only perform behaviors that an expert can demonstrate. But we can explicitly regard exploration and discovery as its own problem, and learn to pursue these goals as effectively as humans do. This does require generalizing across many different acts of discovery, since we want to build a system that finds new things rather than imitating the discovery of an old thing, but such generalization seems essential for powerful AI at any rate.

(This is essentially what happens when we get new behaviors from policy amplification.)

I am particularly partial to this view of exploration/discovery, because it is basically necessary for my approach to AI control—even if I had access to an RL agent, I would try to teach the RL agent to explore in a way that humans approve of, rather than trying to e.g. find new ways to think as part of novelty-based exploration.

Conclusion

Going forward, I'll preferentially design AI control schemes using imitation+RL rather than imitation, episodic RL, or some other assumption. I think this opens up some interesting new questions, will help make theory better match reality, and will make control schemes more broadly applicable.

Not just learning



Paul Christiano [Follow](#)

Oct 16, 2016 · 6 min read

(Note: this post uses the word “learning” in a non-standard way. I mean to include all algorithms that find a model or policy by evaluating similar models/policies a large number of times. This is a defining characteristic of deep learning, and also applies to most existing approaches to learning, but we could imagine alternative algorithms which learn very quickly and to which the discussion in this post does not apply.)

Over the last year I have been working on value alignment for machine learning systems: how can we teach machines to do what we want?

Learning poses some serious challenges for control and security, and recently it has been the bread and butter of AI. So I think that aligned learning deserves to be a priority, and I'll probably continue to focus on it.

But learning isn't everything. In this post I'll argue that the most powerful future AI systems probably *won't* be learned end-to-end, though they will almost certainly use learned components.

Just as learning systems may fail to be aligned with human interests, other components of an AI system may introduce misalignment even if the building blocks work correctly.

I think that the goal of AI control should be to ensure that for every algorithm or technique that might be useful for building a powerful AI, we have a “safe” version which is:

1. **equally useful:** using the safe version doesn't make a system much more expensive or significantly reduce its ability to influence the world.
2. **alignment-preserving:** as long as all of the building blocks perform their intended functions in the desired way, the overall system performs its intended function in the desired way. (ETA: a clearer discussion).

I've been trying to achieve these properties for learning algorithms. But we can and should try to do the same for other AI components; I understand MIRI's agent foundations agenda as (mostly) addressing the alignment of these other elements.

Why learning isn't everything and ALBA can't be sufficient for aligned AI

Learning algorithms require training, which involves evaluating the model-to-be-learned a whole bunch of times. For example, the Q networks in DeepMind's DQN paper were evaluated 320 million times during training, while AlphaGo's value network was evaluated 1.5 billion times during training.

As a result, the learned model can use only a small fraction of available computing resources. If we want to make a really good decision, we are going to want to somehow use more computing power.

For example, AlphaGo does not use the learned model directly to make decisions—it could use the policy network directly, but this results in weak moves. Instead, AlphaGo uses MCTS, calling the value network hundreds of thousands of times in order to make a single move.

It seems like this is a fundamental limitation of learning. If we want to make the best decision possible, we are going to want to use as much computing power as we can. And almost by definition, that means that we won't have enough computing power to train the whole system end-to-end.

ALBA critically exploits this feature of learning. Each learning agent A receives feedback from an overseer H which is much smarter than A. The agent H can be much smarter than A only because it uses much more computational power. Using much more computing than A is feasible only because A needs to be run many times during training (and only if almost all of the runs don't require labels), and so training already needed to use much more computing power than is required to run A a single time.

This approach simply can't apply to the most powerful AI that we are capable of building—we can't define this system's reward by appealing to a more powerful overseer, because we aren't capable of building a more powerful overseer.

Examples of {AI} \ {learning}

There are many AI algorithms other than learning, and most of them present possible safety problems.

Search / planning / etc.

One of the simplest and oldest AI algorithms is to search for an action or plan that is predicted to lead to a desired outcome. We can throw as much computing power as we'd like at the search in order to find better and better elaborate plans, and we don't have to run our search algorithm 300 million times in order to learn how to do it.

If we use a learned model to predict which plans will “lead to a desired outcome,” then an expensive search seems most likely to lead to adversarial examples rather than actions that would actually influence the world. But if this difficulty were overcome, then systems based on expensive searches might have a profound influence on the world. We would have no hope of implementing a more powerful overseer, and so we would need to use some different technique to implement a very robust value function.

The most natural approach is to build an explicit model of the user's preferences. This is the goal of IRL and other approaches to value learning, but I think it's safe to say that we don't yet have a clear understanding of how to do this.

Bayesian inference

If we have a distribution over possible worlds and are able to compute conditional probabilities of the form $P(\text{observation}|\text{world})$, then we can apply Bayes' theorem to compute $P(\text{world}|\text{observation})$. We can extend this basic idea to compute conditional marginal probabilities in complex causal systems or other models.

We can throw a lot of computing power at inference, and the posterior could be more sophisticated than any aligned overseer we can construct.

In order to use inference we need to specify a probabilistic model. Once we are in the business of explicitly specifying probabilistic models it is easy for things to go awry, e.g. to inadvertently specify a prior/model which leads to confident yet unacceptable conclusions; and once the posterior is more sophisticated than any aligned agent that we could construct, its not clear that we can reliably correct an

error. This would probably usually lead to inept behavior, but it could also lead to behavior which was sophisticated but unintended.

Logical deduction

Logical deduction is a powerful-if-conservative framework for deriving new beliefs from old beliefs. Logic allows me to derive beliefs like “ $A \Rightarrow Z$ ” from premises like “ $A \Rightarrow B$ ”, “ $B \Rightarrow C$,” etc...

(Logical deduction is closely related to Bayesian inference and plausible mechanisms for general reasoning would presumably capture aspects of both of them.)

Logical reasoning can be much more powerful than the reasoning underlying each individual deductive step, and the conclusions can be much more sophisticated than any aligned overseer we can construct. Once again we have the situation where we may need to specify some logical premises, and be unable to effectively correct any unacceptable conclusions that follow from those premises.

For example, if we need to write down moral premises about what outcomes or actions are desirable, but we are skeptical about our own ability to correctly formalize these claims, then we may end up with a system that is able to act effectively in the world yet has a mistaken view about what what is good.

Relation to capability amplification

Capability amplification is the problem of using learned components as a building block to implement a more powerful agent, while maintaining alignment.

In general, capability amplification seems to be substantially easier than developing safe versions of the the non-learning techniques in an AI. For example, if we have alignment-preserving versions of planning / inference / deduction, then we could directly use those techniques as our capability amplification scheme.

More generally, we could break capability amplification up into two parts:

1. Developing safe versions of non-learning techniques.

2. Doing capability amplification while ignoring the “maintaining alignment” requirement.

Part #2 looks much easier than part #1. So it's likely that capability amplification will be radically easier if we can solve the rest of the alignment problem. That suggests we should put capability amplification on hold while we work on other aspects of the alignment problem, or at least we should avoid any angles of attack on capability amplification which aren't also promising angles on the rest of the alignment problem.

Conclusion

I think that figuring out how to do learning in an aligned way is one of the most important problems in AI control. But the most powerful AI systems are not likely to be learned end-to-end, and so aligned learning is not the end of the story.

ALBA on GitHub



Paul Christiano [Follow](#)

Oct 19, 2016 · 5 min read

I've put up an implementation of ALBA on GitHub here.

The key method is `ALBA(H, n)`, which hopefully implements an aligned AI. To actually use it, you would need to replace the stub `TransparentHybrid` with a competent algorithm for transparent semi-supervised imitation+RL.

For now you can mess around with a version of ALBA that uses a very simple learning algorithm—whenever it encounter a novel situation, ask the overseer what to do and memorize their answer for future reference.

I don't think that actually running the example is especially informative. I wrote the code in order to make the discussion of ALBA more precise, honest, and concrete, not because I thought that anyone would want to actually run it.

TODOs

In reality this implementation of `ALBA(H, n)` definitely *wouldn't* be an aligned AI; there are at least half a dozen obviously fatal problems. An immediate goal is to address the obvious problems: to write some code such that *if* we filled in the AI capabilities, then there would at least be *any hope* that the resulting system was aligned. I don't know how hard this goal is; I think there is a plausible roadmap and it might be relatively easy, or it might take a very long time.

(Realistically, even if there was *any* hope that the system would work, we would still need to improve all of the ingredients further before there was *much* hope that it would work.)

If that works out then we can start to search for non-obvious problems, and argue about more philosophical questions like whether `Meta(HCH(·))` correctly implements capability amplification.

Here is the list of TODOs and FIXMEs in `alba.py`:

- TODO: Build powerful AI

- TODO: Implement semi-supervised RL
- TODO: Implement transparent RL
- FIXME: Prevent catastrophic failure on adversarial inputs.
Adversarial training?
- FIXME: Ensure RL agent cannot outsmart overseer. Gradually scale up capacity as a function of n?
- FIXME: Prevent failure probability from growing with each iteration. Amplify reliability as well as capability?
- FIXME: Allow Amplify(A) to learn from training data, so it can keep up with the RL agent it is overseeing
- FIXME: Scores in [0, 1] are arbitrary, use comparisons between different actions instead
- FIXME: Use budgeted HCH so that errors can't result in hangs
- TODO: Figure out whether iterating $A \rightarrow \text{Meta}(\text{HCH}(A))$ can really get us to arbitrarily powerful agents

How would this be used?

`ALBA` is designed to be an essentially drop-in replacement for imitation learning or RL. Of course you would only need to use it when you didn't have access to a suitable objective function.

A realistic AI system may use RL or imitation learning as components but will generally have other stuff going on as well. That's fine, you drop ALBA in for the learning part. For example, in AlphaGo you would drop in ALBA for the policy and value networks, but you'd still have to do MCTS.

Hopefully none of your other code is going to destroy the world, but that's not our responsibility.

`ALBA(H, n)` returns an agent A that is aligned with H but much smarter. You can use this agent in the same way that you'd use an episodic RL agent, except that you don't have to give it a reward. To start an episode you make a fresh copy of A ; for example, each time a user visits `google.com` you might make a new copy of A and then interact with that agent in order to figure out how to process their queries, with calls like `A.act("the user just typed where should I eat dinner? , what should happen now?")`. In principle you could have

very long-running episodes (as with RL), but this would make training more complex.

Instead of specifying a reward directly we specify some overseer H . The system is designed to do what H wants, so picking H is very important. For example, if ALBA were being used in the UI for google.com, H might be implemented by professional trainers employed by Google. If Google wants to make a product that is useful to the users, then it tells the trainers that they should try to think about what the user wants and make decisions appropriately. But the trainers can also balance the user’s interests with other desiderata, like Google’s profit motive or legal constraints. (All of this is analogous to how Google generally balances the desire to be useful to the user against other desiderata.)

Before the first time that ALBA is invoked, it will perform some pre-training to ensure that its behavior is non-catastrophic, including meeting some minimal bar for quality. [This doesn’t happen in the current implementation; it’s the FIXME about avoiding catastrophes.] During operation it makes decisions immediately, but will continue collecting data and training asynchronously in the background. [Training occurs inside the implementation of the AI capabilities in capabilities.py.] During training, H will be invoked constantly.

If H is overseeing a single instance of ALBA that is performing a narrow task, then we can just explain the definition of the task to H . If H is overseeing a range of different agents pursuing a variety of tasks, then we can encode the task definition in the agent’s observations. For example, if we want it to caption an image, rather than simply giving the agent the image and expecting a caption, we could also give the agent the natural language text “please describe what you see in this image,” and perhaps a more detailed description of what kind of explanation we want, and so on. This information will ultimately be used by an aligned overseer to decide how to implement a reward function. As a result, the agent will learn to respond differently to different task explanations. (If the agent is very weak, this might be done simply by memorizing a few different possible task descriptions—we don’t inherently require that the agent can learn to understand language, though the whole system becomes more flexible at that point.)

If we were using an RL agent we would implement a reward function in order to cause the agent to do what we want. We don’t provide a

reward function for ALBA, but we can let the agent observe “rewards” to help it learn faster. This can be used as helpful unlabelled data, or it can be absorbed by the overseer to improve its understanding of the task. [This doesn’t happen in the current implementation; the FIXME about having the overseer learn from training data.]

Conclusion

I think it’s useful to be able to point to a completely concrete specification of ALBA. I also think that a concrete implementation is a nice setting in which to start fixing the obviously fatal problems, and I’ll be especially excited if/when there is a concrete implementation that *isn’t-obviously-doomed-to-fail*. For now, I hope that having access to the code will help people dig into any details of the scheme that they found ambiguous or unclear in the informal descriptions.

Reliability amplification



Paul Christiano [Follow](#)

Oct 20, 2016 · 9 min read

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 **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 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 **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 1cbbae, 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 $O(\epsilon^2)$.

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 \mathbf{A} picks at random amongst the 100 possible answers. If the copies of \mathbf{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 $(1-\epsilon)$ 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 $o(\epsilon)$, and our capability amplification causes a linear increase from ϵ to $\Theta(\epsilon)$, 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 $\Theta(\epsilon)$. For sufficiently small ϵ , this is more than offset by the $\epsilon \rightarrow o(\epsilon)$ reliability improvement from reliability amplification. So for sufficiently small ϵ we can increase both reliability and capability.

A reduction from ϵ to $O(\epsilon^2)$ 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 $\epsilon \rightarrow O(\epsilon)$ 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.

AI control and search/planning



Paul Christiano [Follow](#)

Oct 21, 2016 · 5 min read

(Status: sketchy. Follow-up to: *not just learning*.)

Consider the following simple algorithm:

- Train a predicted-reward function $R(o, a)$ and an action-proposer $A(o)$.
- Sample a large number of candidate actions from $A(o)$, and then output the action a for which $R(o, a)$ is maximal.

This kind of search poses some challenges for AI control:

1. If R predicts something doesn't quite capture our values, like "will the reward button get pressed," then we will get a reward-button-maximizing AI. This raises a familiar set of concerns.
2. Even if R reflects our values most of the time, the search may find the rare points on which R is badly wrong.

This kind of search can also be generalized in many directions; for example, we can use MCTS to search over sequences of actions rather than individual actions (as in AlphaGo). Many of these variants seem to pose a similar set of problems for AI control.

Problem #1

I think that problem #1 should be regarded as a bug in the reward function; if we can solve AI control for the techniques that produced R , then we should be able to address this problem.

For example, if we train R using supervised learning from actual button presses then we will have trouble. But if we have solved AI control for supervised learning, then we could use that scheme to learn an equally-sophisticated reward function which captured what we care about "to the best of its abilities" (the aligned learner could treat data about button presses as a useful signal to help predict goodness). For example, if R is smart enough to predict that action a will lead to the user's death, then it's reasonable to expect R to be smart enough to know that action a is bad.

If we can't solve AI control for supervised learning then we are in trouble whether or not we perform the search. Either way, no need to blame the search.

Problem #2

I think that problem #2 is much more serious.

In the best case, we can think of the reward function R as itself being an aligned AI, which looks at an action and determines how good it is.

The whole point of doing the search is to turn up actions that are more sophisticated than anything that R could generate itself. If we are running a *really big* search, then we are trying to find actions that are more sophisticated than anything that R has ever seen before.

This is a recipe for R to make a terrible mistake. Maybe you'll just get something like an adversarial example which isn't a meaningful action. Or maybe you will get something more sinister, which does damage when R attempts to evaluate it (e.g. the search could turn up security vulnerabilities). In any case, there isn't much reason to think that you will get an outcome which is actually *good*, while there is reason to think that you might get an outcome which is very *optimized*. I think that should be cause for concern.

Current state of affairs

I think that this is a pretty hard problem, and that most proposed responses are not satisfactory:

- Any response along the lines “don’t run a giant brute force search” should explain how to be algorithmically competitive with an alternative that *does* run a giant brute force search (or else accept that aligned AI systems may be at a large competitive disadvantage).
- Any solution that depends on having a very expensive function R should explain how that isn’t going to become the bottleneck for the search and drag down performance.
- Any solution that involves crossing our fingers and hoping for the best should probably say something about why we expect R to behave *at all sensibly* on an input that is totally unlike anything it has seen before, and was selected for being a point where R was likely to make an error.

So far this hasn't really been an issue in practical systems, perhaps because extensive search has mostly been effective in domains like games or constraint satisfaction where the reward function is taken as given. But when we eventually design algorithms that effectively combine learned models with extensive search, I think this may become a problem. Optimistically it will be a problem that needs to be solved before giant searches work at all; pessimistically, we may find ad hoc solutions that cause our systems to be more brittle and fail in more spectacular ways.

I have a rough angle of attack in mind, which I hope to flesh out soon. The idea is that *most* proposed actions will be typical enough that R can handle them directly. So if we can identify the extreme actions as being extreme, we can afford to spend significantly more time processing them. If we can get abstraction+adversarial training working well enough, then we could present abstract versions of these inputs to R, which could then take the time to evaluate them slowly—with no individual step of the abstract evaluation being too far from R's training distribution.

Optimism about Bayesianism

Many people (including me) have the intuition that a Bayesian reasoner may be able to implement a more robust/safe planning process. I think that this intuition has merit, but ultimately I think that there is still a lot of work to be done if it is going to be turned into a solution:

- If we apply the kind of search described in this post with a Bayesian reasoner, then we get the a that optimizes our estimate of $\mathbb{E}[U|o, \text{do}(a)]$. We might as well call that estimate $R(o, a)$. Being Bayesian doesn't seem to change the basic problem: R needs to be calculated and trained efficiently, and so it is difficult to ensure that it behaves sensibly on actions a selected from an extremely extensive search.
- So I think that optimism about Bayesian is based on the hope that Bayesian methods will learn very robust models that can reliably generalize outside of the training distribution. I think that this is a plausible hope and a promising direction to explore, but for now I would definitely not want to hang my hat on it—I don't have much optimism that *in general* we will be able to learn very-robust models as efficiently as non-robust models. I think that even if research on robustness succeeds spectacularly

it probably won't be able to establish a sufficiently general claim to let us rest easily.

- A Bayesian approach may allow for a tighter coupling between the representation of the goals and the actual mechanics of the search. For example, rather than considering a bunch of actions a we may reason backwards from a goal; this may pose a much different (and potentially easier) problem for AI control. But I think we need to understand whatever techniques are most useful; that might mean backwards-chaining, but it could also mean the kind of more straightforward approach applied here. So I don't think it's yet reasonable to restrict attention to search strategies that have a particular kind of internal structure.

Conclusion

I think that powerful AI systems will probably make use of large-scale search, and we will probably need new techniques in order to perform them safely. I expect this to be a challenging problem, which may be quite different from the kind of difficulties addressed by a scheme like ALBA.

Some thoughts on training highly reliable models



Paul Christiano [Follow](#)

Oct 22, 2016 · 7 min read

I am interested in building machine learning systems which are very unlikely to fail catastrophically, even on rare or adversarial inputs.

This problem seems quite hard; if it's possible, it will probably need to use methods that exploit the internal structure of our models, rather than being able to treat out-of-distribution detection or adversarial training as an independent post-processing step.

Identifying vs. handling problematic inputs

I'm going to talk about *identifying* problematic inputs rather than handling them gracefully. Obviously handling them gracefully would be more satisfying. But I suspect that flagging problematic inputs will be an important first step. If we can flag problematic inputs, then:

- We can try to generate maximally-problematic inputs during training (as in adversarial training).
- At test time, we can filter/abstract problematic inputs, and can search for non-problematic abstract versions.
- (We discuss some other responses in Concrete Problems in AI Safety.)

I don't currently see a plausible *general* approach to handling problematic inputs that doesn't go through the intermediate step of either recognizing or sampling them.

1. Inadequacy of density modeling

If we train a learning system on some distribution D, it might behave quite badly on inputs from a different distribution D'. If it would be great to notice such distributional shifts and to flag inputs on which our model might behave badly.

I've heard the following proposal a few times:

- Fit a density model p to the training distribution D .
- Whenever you encounter a data point that is too far from the training distribution, flag it as potentially problematic. For example, we might flag x whenever $-\log p(x) \gg \text{entropy}(p)$.

We could define “too far from the training distribution” in many alternative ways.

But as far as I can tell, no variant of this approach will be sufficient for building a robust system.

Adversarial inputs

Adversarial inputs necessarily have a high *ratio* between the probability $q(x)$ under the adversarial distribution and the probability $p(x)$ under the training distribution. But they need not look exceptional at all if we consider just the training distribution.

For example, consider a distribution over high-dimensional images where the camera adds a bit of Gaussian noise on each pixel. If the images are very large and we haven't done any adversarial training, it seems likely that we can produce an adversarial example with a very small per-pixel perturbation, e.g. with the fast gradient sign method. (The perturbations may be small relative to the standard deviation of the Gaussian noise during training.)

Under Gaussian noise, these small perturbations are just about as probable as any other perturbation. The weird thing about the perturbation is that it has an extremely high correlation with the gradient of the prediction—as far as the data distribution p is concerned, this fact is totally invisible.

Of course if we had access to the adversarial distribution, or could train a generative model q on adversarial examples, then we could easily tell that this was an adversarial input. Similarly, if we had access to the features that the adversary was using to generate their attack then we might easily notice that these features were taking on very abnormal values. But these techniques only work when we have *specific hypothesis about how our model may not be robust*.

The hardest (and also typical) case is when we don't know exactly what kind of inputs may trip up our model at test time, or where accessing the problematic distribution is too expensive to be

integrated as part of the training process. It seems like out-of-distribution detection is not enough to handle these cases.

Non-adversarial robustness

The discussion above is clearest in the context of adversarial examples, but I think the situation is the same for other kinds of distributional shift.

For simplicity, let's analogize our training and test distributions to Gaussians. We can potentially detect the case where the test distribution is “bigger” than the training distribution, supported on points that have negligible density during training. But from the perspective of robustness and generalization, we should be just as worried when the test distribution is “smaller” than the training distribution, supported on some tiny subset of the training distribution (whose total probability was negligible).

If we visualize the low-dimensional setting, it is easy to imagine that we will be able to handle the case of “smaller” test distributions using some kind of interpolation, while “bigger” test distributions demand a more challenging extrapolation. But that intuition seems to completely break down in high dimensions—being focused on a small part of the distribution can make feature expectations just as extreme as if the test distribution were located in a totally different part of the space, even for linear features. (This is illustrated especially cleanly by adversarial examples, but I think the phenomenon is pretty universal.)

Lots of data

If we can collect a lot of test data, then it becomes more straightforward to notice whether the training and test distributions differ. For example, we could simply train a discriminative model to distinguish them.

I am not satisfied with this approach, because I want systems to recognize and respond to catastrophe-inducing inputs immediately, I don't want to wait until we have gotten enough of them that we can train a discriminative model.

This technique may be appropriate when there is a particular moment when we are concerned that our distribution will shift, for example when applying an algorithm in a new place. But it does not seem appropriate when we are concerned about distributional shifts during normal operation or about outliers.

2. Rare failures can occur without distributional shift

People often talk somewhat interchangeably about distributional shift and non-robustness. But if we are interested in models that avoid rare catastrophes, then I think that we need to worry about some cases where there is *no* distributional shift.

For example, suppose that we are training a model to evaluate the financial cost of an earthquake. We train our model on many earthquakes, and it performs extremely accurately. But if we evaluate on a 100x larger test set, then by chance alone the test set may contain earthquakes an order of magnitude bigger than anything we encountered in training.

Our model may not be able to make a reliable prediction about this earthquake, despite the fact that the test and training distributions are identical. Even though such errors will necessarily be rare, in some contexts they may be catastrophically bad.

In some sense this seems like an “unfair” problem—how can you expect to get failure rates much lower than $1/N$ if you are only training your model on N observations? But I think that in the real world there are settings where failure is very expensive, and $1/N$ is an unacceptable failure rate. For example, if we train a system on 1 month of data, then a failure rate of $1/N$ corresponds to one arbitrarily-catastrophic failure per month. Even if we train on a decade of data, $1/N$ corresponds to a failure rate of 10%/year.

Either we need techniques for driving error rates below this threshold, or we will need to ensure that machine learning systems are only deployed in settings where there is no possibility of catastrophic failure. Redundancy is often used in order to obtain very low failure rates for systems involving humans; but for machine learning systems, different systems trained on data from the same distribution may tend to fail simultaneously, and so it might not be so easy to avoid catastrophic failure.

Outlier detection

We could hope to detect outliers in a model-independent way. For example, there is a clear sense in which the biggest earthquake ever is an outlier.

However, if our model had access to enough kind of different data about the world, there will be a constant stream of “outlier” events. If we want to identify problematic inputs without having an unlimited number of false positives, we really need to know that the big earthquake is the kind of outlier that is relevant to our model.

3. Some other angles

I don’t think that it is possible to train robust models by treating out-of-distribution detection or adversarial training as an external postprocessing step that uses the model as a black box. Instead it seems like we will need to look inside the models we are training, or perhaps train models which are able to directly report when they are uncertain. Such models might be able to cope with both changes in distribution and problematic rare inputs.

One approach is to maintain a distribution over models, and flag an input as uncertain when its label is substantially correlated with our model uncertainty.

(Another approach, that I won’t talk about here, is to look at features used internally by the model. If we check for anomalies in those feature values, then maybe we can identify the most problematic inputs.)

The hope is that when an input *isn’t* flagged as uncertain, then there is essentially “consensus” amongst the plausible models. Depending on how wide a distribution over models we can maintain, and where we set the threshold for model uncertainty, this might allow us to draw strong conclusions: rather than “our current model predicts X” we could hope to say something like “all simple models consistent with the data agree on X.”

I think there are lots of challenges with getting the kinds of guarantees that we actually want—existing approaches don’t yet do the job. For example, I don’t believe that any existing approach is able to robustly flag adversarial examples as uncertain in vision tasks. But I think this is a promising direction and I’m excited to see what kind of results will appear over the coming years.

Conclusion

There has been a lot of research on training robust models, and I’m excited to see where it goes over the coming years. If aiming for the

ambitious goal of completely avoiding catastrophic failures, I think it is worthwhile to keep in mind the possibility of data that *could* appear in training but which is sufficiently rare that it happens to have never shown up (yet sufficiently common that it isn't too surprising to see it at test time), rather than thinking of robustness as only occurring when the training and test distributions are different.

Of humans and universality thresholds



Paul Christiano

[Follow](#)

Oct 23, 2016 · 4 min read

(Note: relevant to a narrow audience.)

I think that something like HCH might be extremely sophisticated while remaining aligned with human values (when combined with some form of reliability amplification). In fact I'm cautiously optimistic that if you put any sufficiently smart agent into a better version of HCH then you will get out a “universal” agent, which depends on the *values* of the agent that you put in but doesn't depend on its *abilities*.

I've encountered the following objection:

- It seems clear enough that *no number of apes* could invent calculus in an ape-aligned way. In order to understand the values of an ape, we need to do some kind of bona fide extrapolation, to ask what would happen if the ape was more sophisticated rather than merely asking what would happen if there were more of them.

In fact we don't even have to go so far as an ape—if you took most people and put them in a box and asked them to organize a trillion copies of themselves to compute their enlightened judgment, the results wouldn't be pretty.

It seems strange for the people having this discussion to imagine that the threshold is right below them. Surely it is more likely that there are more sophisticated creatures than humans, who might think to themselves “well clearly if you put a *human* in this process it wouldn't work, most of them couldn't even understand inter-universal Teichmüller theory.”

We could make the same argument with respect to time: it seems likely that HCH fails if run with a human who only has ten seconds to spend thinking. So why think that a day is enough time?

I think that these arguments are mistaken—that’s not to say that HCH is obviously universal, but it is to say that we shouldn’t be *a priori* surprised if there is a universality threshold and it lies within the range of human variation, or between “ten seconds” and “a day.”

An algorithmic analogy

Suppose that we were instead interested in the *algorithmic* capacity of HCH: what functions can be computed by HCH, given a natural description of the function?

For this problem, there seems to be a clear universality threshold. As long as a human is sophisticated enough to eventually understand and develop the idea of a robust universal computer, and to implement simple logic operations with modest accuracy, then their HCH can compute arbitrary computable functions.

On the other hand, an ape’s HCH is not going to be able to implement much of anything, and indeed it seems plausible that the majority of humans would not be able to implement particularly complex functions. And if we give a human only ten seconds, they probably can’t implement a very complex function either.

So for this natural problem there actually *is* a clear universality threshold in HCH’s behavior.

Caveats

How good is the analogy? One might object that this universality claim is really about learning to use external computational resources—in some sense this is different in kind from moral deliberation.

But I think these two cases are actually closely analogous in the relevant way.

We are confident that an ape’s HCH doesn’t do anything good for basically the same reasons we are confident that an ape’s HCH can’t compute anything complicated: they couldn’t build any machinery to do anything more sophisticated than the initial ape.

How is the function described? One might object to assuming that we are “given a natural description of the function.” Perhaps this smuggles in a human-centric notion of “natural description.”

But we could loosen this condition to “given *any* advice whatsoever” and we still seem to observe a universality threshold—there seems to be **no** reasonable advice that would cause a realistic ape to reliably implement a very complex logical function. For a clever human you can pick just about any natural representation and they’ll get it.

Alternatively, we could observe that language itself seems to exhibit a universality transition, which is independent support for the original claim. Humans understand a range of universal languages into which any other language can be translated with enough effort (roughly speaking). Other animals do not have such a language.

Conclusion

The analogy to computation doesn’t provide direct positive evidence *for* the universality of HCH in the moral sense. But I think it does provide a strong reason to be skeptical of skepticism about the existence of a universality threshold within the range of human variation; universality thresholds are not an uncommon phenomenon, and in fact humans straddle a number of similar thresholds.

In the end I think that we have to use other arguments and intuitions to try to determine whether HCH is universal.

Meta-execution



Paul Christiano [Follow](#)

Oct 26, 2016 · 6 min read

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.)

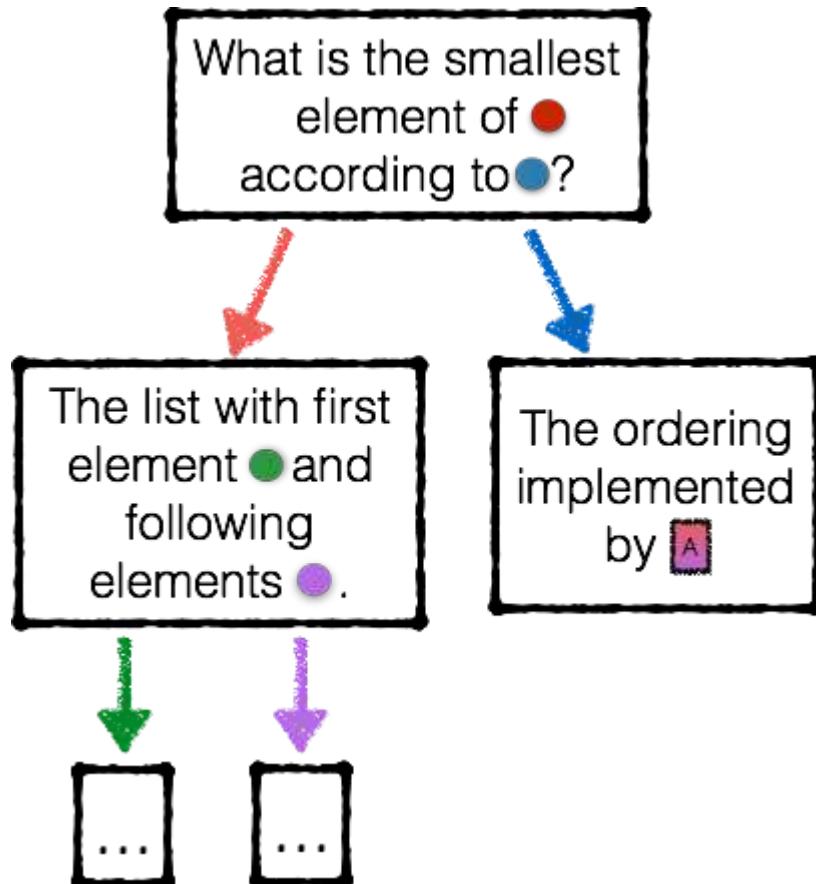
Goal

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.

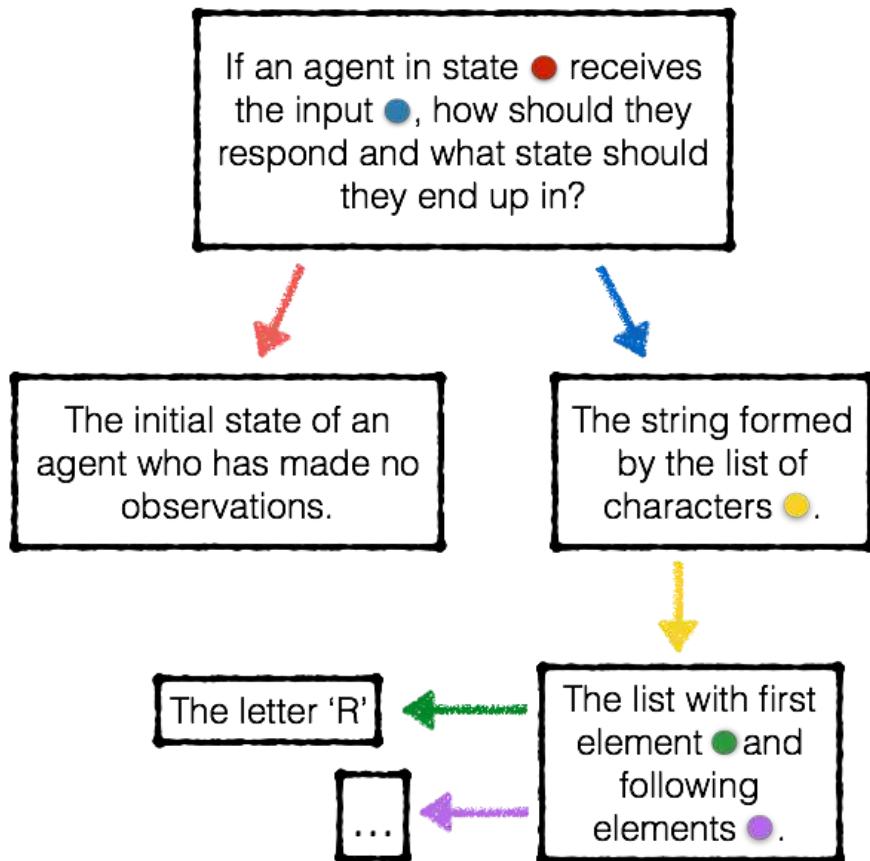


Each box is a message. A is an agent who can respond to queries like “which of X and Y is larger?”

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.

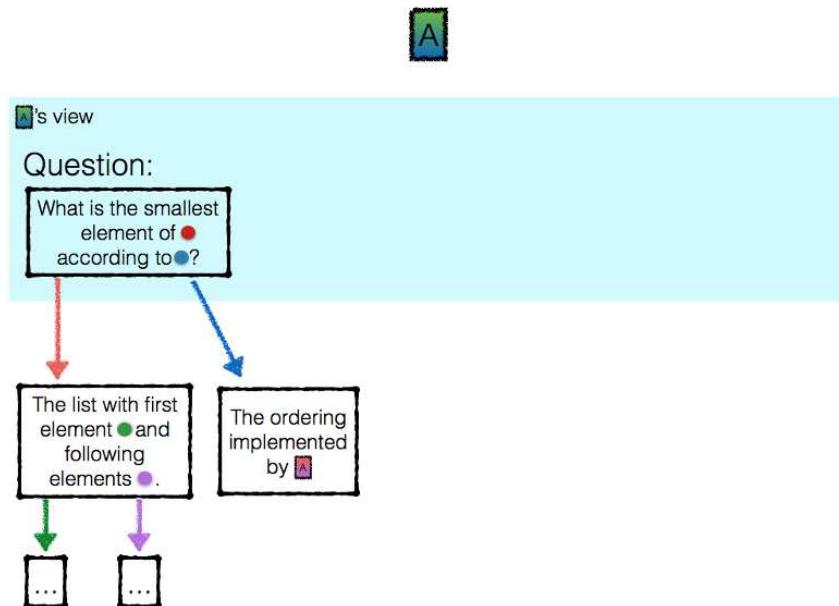
At any given time, an agent who is operating on this tree can only “see” a few messages: it can read the text of those messages, and see pointers like [red] and [blue]. Initially the agent can see only the root of the tree.

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.

Security amplification



Paul Christiano

[Follow](#)

Oct 26, 2016 · 15 min read

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 **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 **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 processes 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 *any particular* 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 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 suggest 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 or 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.

Thoughts on reward engineering



Paul Christiano [Follow](#)

Nov 8, 2016 · 13 min read

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 express 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 be 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 to 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.

Handling destructive technology



Paul Christiano

[Follow](#)

Nov 14, 2016 · 5 min read

Some technologies are much more helpful for destroying stuff than protecting it. If Alice and Bob both have large nuclear arsenals, and one of them wants the world destroyed, then the world is probably going to get destroyed. If everyone has a large nuclear arsenal and *anyone* wants the world destroyed—well, so it goes.

Preventing nuclear weapons from causing incredible destruction required some global coordination and luck. But things could have been a lot worse; imagine a universe where running twenty amps through a solenoid is enough to destroy the world.

It seems likely that we will eventually encounter technologies that are much more problematic than nuclear weapons: that are much cheaper and more destructive, that don't require hard-to-obtain materials, or that have larger economic importance with fewer close substitutes. Coping with such technologies would require much better global coordination. In the limiting case, the use of sophisticated technologies might need to be centrally coordinated, such that no agent could decide to use them in a destructive way.

In the context of this post, I'll refer to this hypothetical situation as civilization “having our house in order.”

AI and delaying the inevitable

If we are not able to solve the control problem, I think that AI will itself be a destructive technology. The incentives to use powerful AI will be significant and the world will move increasingly rapidly and incomprehensibly, but it will be difficult to build competent AI systems that represent human interests. Human values will eventually fade from prominence, just as surely as if humanity had destroyed itself in a nuclear war.

This motivates me to work on AI control, and I think it is amongst the most important technical problems that we can work on today.

But whether or not AI itself is a destructive technology, it seems likely that we will eventually encounter much more destructive physical

technologies, and we will need to have our house in order at that point.

So solving the AI control problem merely delays the inevitable. If we get our house in order *prior* to the development of AI then it won't matter whether we solved the control problem in advance. And if we *never* get our house in order, then it doesn't matter whether or not we solved the control problem, we are doomed anyway. AI control only matters in the case where we get our house in order between the development of AI and the next time we develop a comparably destructive technology.

I think that solving the control problem is very important anyway, even if it only buys us time:

- Buying time straightforwardly improves our odds of solving the problem by giving us at least one extra chance.
- As technology improves I think we have a better and better shot at getting our house in order. And there are other trends that seem to point in a positive direction as well.
- I think that AI itself may be an important part of getting our house in order, both by expanding the human capability for reason and by opening up new coordination mechanisms.
- There is at least a chance that there won't be any really destructive technologies after AI, or that such technologies won't occur again for a very long time. Other changes like independence from earth's ecology and dispersion to space might be adequate to protect us from upcoming physical technologies (though they clearly wouldn't be sufficient to protect us from failure to solve the control problem).
- It is possible that the transition to AI will be a natural moment to get our house in order, e.g. because it results in a disruption of the existing world order. I feel hesitant about this possibility, and think that if the development of AI involves an "upset of the existing world order" we are probably in for a bad time. But it's definitely a possibility.

Overall, it is correct to claim that AI control is less important because we will need to get our house in order eventually anyway. But I don't think it changes the cost-benefit analysis very much—I think you'd have a hard time arguing for a factor of 2, much less a factor of 10.

Delaying the inevitable how far?

I expect that sophisticated AI will significantly increase the pace of technological progress, just as the pace of change today is much faster than it was 300 years ago, which was in turn much faster than the pace of change 3000 years ago.

If we imagine a future where major technological change occurs over the scale of years or months rather than decades, it may feel like we aren't going to "delay the inevitable" very far. On this view, if we don't have our house in order by the time we develop sophisticated AI, then we aren't going to have time to get it in order before new destructive technologies are developed.

I encounter this view all of the time, but I think it is probably mistaken. This argument implicitly measures developments by *calendar time*—how many years elapsed between the development of AI and the development of destructive physical technology? If we haven't gotten our house in order by 2045, goes the argument, then what chance do we have of getting our house in order by 2047?

But in the worlds where AI radically increases the pace of technological progress, this is the wrong way to measure. In those worlds science isn't being done by humans, it is being done by a complex ecology of interacting machines moving an order of magnitude faster than modern society. Probably it's not just science: everything is getting done by a complex ecology of interacting machines at unprecedented speed.

If we want to ask about "how much stuff will happen", or "how much change we will see", it is more appropriate to think about *subjective time*: how much thinking and acting actually got done? It doesn't really matter how many times the earth went around the sun.

Now it might be that AI speeds up technological progress without speeding up the process of getting our house in order. My best guess is that if we actually solve the control problem, then it's actually the other way around; but I certainly agree that both are possible.

Even if we assigned only a 50% chance of AI accelerating the process of getting-our-house-in-order as much as it accelerates technological progress, that would cut the value of the control problem by at most a factor of 2. I don't think this is enough to really change the basic calculus for anyone considering whether to invest resources in AI control.

Conclusion

In some sense solving the AI control problem is merely “delaying the inevitable,” pushing back the moment when society needs to have its house in good enough order that it can cope with the discovery of powerful destructive technologies.

I think that this observation can help clarify exactly what we are doing when we work on AI control, and may change how we think about other interventions to safeguard humanity’s future. But I don’t think that it substantially changes the cost-benefit analysis for AI control itself.

I often encounter the argument that AI will facilitate access to powerful destructive technologies, and for that reason requires us to have our house in order whether or not we solve the control problem. I think that this argument is mistaken, most of all by overlooking the ability of AI itself to help us get our house in order.

AI “safety” vs “control” vs “alignment”



Paul Christiano [Follow](#)

Nov 19, 2016

Note: there has been some discussion recently about what term to use, and there are several legitimate complaints with control (for example, see Robby’s responses to this post). I’m tentatively moving towards the term “AI alignment” as a substitute for what is called “AI control” here.

I take these terms to describe a sequence of increasingly specific problems:

- **AI safety:** reducing risks posed by AI, especially powerful AI.
Includes problems in misuse, robustness, reliability, security, privacy, and other areas. (Subsumes AI control.)
- **AI control:** ensuring that AI systems try to do the right thing, and in particular that they don’t competently pursue the wrong thing. I’ve argued that it’s roughly the same set of problems as AI security.
- **Value alignment:** understanding how to build AI systems that share human preferences/values, typically by learning them from humans. (An aspect of AI control.)

It would be great to hear if others use or understand these terms differently.

As you might guess from the title of this blog, my research is about AI control.

Prosaic AI alignment



Paul Christiano

[Follow](#)

Nov 19, 2016 · 10 min read

(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 $1/(p_{\text{most likely}})$. 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

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.

Hard-core subproblems



Paul Christiano

[Follow](#)

Nov 26, 2016 · 2 min read

Given a research problem X, say that Y is a hard-core subproblem if:

1. A solution to X implies a solution to Y.
2. We aren't currently making progress on Y, we don't know how to make progress on Y, and Y isn't getting any easier over time.

Example

I think that the easy goal inference problem is a hard-core subproblem of cooperative inverse reinforcement learning (CIRL), or at least for the application of CIRL to superintelligence.

CIRL will become easier as we develop improved AI systems, and poses many natural theoretical and practical questions. But the easy goal inference problem doesn't seem to be getting any easier over time, and I don't think we have compelling angles of attack on this problem.

Moreover, a solution to CIRL implies a solution to the easy goal inference problem, since the easy goal inference problem is just the special case where we have perfect information and unlimited time.

Relevance

If we can identify and agree on one or more hard-core subproblems then I think we should generally prioritize work on them. If a hard core turns out to be easy, then we'll have learned something and not much is lost. If a hard core turns out to be very hard, then it's probably a good thing to focus on—it's a prime contender for a bottleneck, and it's likely to be a key conceptual aspect of the problem.

If we suspect that a problem is likely to be insoluble, I think we should prioritize distilling that intuition/argument into a hard-core subproblem. If we succeed, then we have found much better evidence that the problem is hard, and acquired a useful tool for organizing work on the problem. If we can't find a hard-core subproblem, I think that partially undermines our original suspicion. (Of course the

problem may still be insoluble, but we probably shouldn't be confident that it is insoluble.)

Conclusion

In general, I think that identifying a hard core is an important tool both for understanding whether a problem is hard, and for clarifying and organizing research programs. I think this concept might help bridge the gap between some researchers in the AI safety community (especially at MIRI) who are highly pessimistic about AI control, and AI researchers who are optimistic that we can “cross that bridge when we come to it.”

This concept is complementary to a focus on prosaic AGI. In some sense *any* problem might become easier over time, since we might encounter some unknown unknown that bears on that problem. But it's much easier for a subproblem to be a hard core *conditioned on prosaic AGI*, since in that case we won't encounter any unknown unknowns: if you want to claim that some subproblem will get easier over time, you have to actually point to the known unknown that will make it easier.

Benign AI



Paul Christiano

[Follow](#)

Nov 29, 2016 · 4 min read

Like it or not, humans are now in the business of building machines that optimize—we write code that trains computer vision systems, we design motion planning algorithms that control robots’ behavior, and so on.

The goal of AI control is to ensure the results are optimized for our interests. This post introduces some definitions I find helpful for that project.

Talking about “our” interests is a little bit tricky, since different humans want different things. For concreteness I’ll think about a particular system (e.g. Google Now running on your phone) and its “stakeholders:” its owners (you), designers (Google), and users (whoever is talking to the phone).

Something is **malign** if it is optimized for preferences that are incompatible with any combination of its stakeholders’ preferences, i.e. such that over the long run using resources in accordance with the optimization’s implicit preferences is not Pareto efficient for the stakeholders.

Something is **benign** if it is not malign.

I previously said “aligned” instead of “benign.” But this isn’t how other people use language, so I think that I need a new word. For example, I think that a toaster is benign, but apparently most people don’t want to call a toaster “aligned.”

Dynamics

Malignancy wants to spread. A malign computation will produce malign outputs. Malign outputs will be optimized to recruit other parts of the system to be malign, or to co-opt resources used by other parts of the system.

So it’s especially important to understand how malignancy can enter into a system, and to either prevent it from entering or to limit its spread.

Today, malignancy is usually introduced by an adversary. For example, data received over the internet is likely to be malign. If you are designing a server, you need to be very careful to prevent the malignancy from spreading, and this turns out to be a surprisingly difficult problem.

Malignancy can also be introduced by certain algorithmic techniques. For example, gradient descent can produce policies that are highly optimized to have particular effects on the external world. Unless the optimization criterion reflects the user's preferences, those policies (and hence their outputs) will be malign.

For sufficiently weak optimization this isn't a problem, because the resulting policies aren't powerful enough to effectively spread or to compete with humans. But if we performed powerful enough optimization to produce human-level cognition, then we could have a problem.

Why the concept is useful

I think that benignity is useful as an invariant when designing and analyzing aligned AI systems.

On one end, it's easy to start with benign building blocks. User input is benign, since it's optimized by the user. The software we write is benign, as a special case of user input. Randomness is benign.

On the other end, powerful AI systems with benign outputs are necessarily aligned—their outputs are powerfully optimized, and not at all optimized for anything that we don't like, ∴ their outputs are effectively optimized for our preferences.

So the game is to build a powerful AI without introducing any malignancy (and while coping with any malign data injected by an adversary). Each time we want to combine several ingredients, we can ask: assuming that the ingredients are benign, will the result be?

For example, in capability amplification the goal is to ensure that *if* the starting policy is benign, then the amplified policy is benign. In reward engineering the goal is to ensure that *if* the overseer is benign, then the learned policy is benign. And so on.

Why the concept is incomplete

I often use this notion of benignity, but it's definitely not completely satisfying.

For example, something can be optimized for different preferences to different extents; under realistic conditions I suspect that anything interesting will be weakly optimized in many different directions, and so *everything* is probably malign according to the all-or-nothing definition.

Realistically benignity needs to be understood quantitatively rather than bluntly and qualitatively. (Long ago Eliezer wrote about an approach to quantifying optimization, which I think is still at a “blunt qualitative” stage of elaboration.)

As another example, optimization is probably best understood with respect to some epistemic state, and we probably care about optimization with respect to *our* current beliefs.

There are many other subtleties along these lines, and we would need to resolve many of them in order to actually use benignity in any kind of precise analysis.

For now I think of benignity as a placeholder; I hope that eventually we’ll be able to find a theoretically cleaner analog (and I’m counting on that cleaner analog to do a lot of heavy lifting).

I may be too optimistic, and benignity may not turn into a helpful concept for precise analysis. But I’ve already found the informal version to be useful for my own thinking.

Recent posts



Paul Christiano

[Follow](#)

Nov 29, 2016

Here's what I've been thinking/writing about AI control over the last month.

Strategy

- Prosaic AI control argues that AI control research should first consider the case where AI involves no “unknown unknowns.”
- Handling destructive technology tries to explain the upside of AI control, if we live in a universe where we eventually need to build a singleton anyway.
- Hard-core subproblems explains a concept I find helpful for organizing research.

Building blocks of ALBA

- Security amplification and reliability amplification are complements to capability amplification. Ensembling for reliability is now implemented in ALBA on github.
- Meta-execution is my current leading contender for security and capability amplification. It's totally unclear how well it can work (some relevant speculation).
- Thoughts on reward engineering discusses a bunch of prosaic but important issues when designing reward functions.

Terminology and concepts

- Clarifying the distinction between safety, control and alignment.
- Benignity may be a useful invariant when designing aligned AI.

Directions and desiderata for AI alignment



Paul Christiano

[Follow](#)

Feb 6, 2017 · 16 min read

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 be 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.

Approaches to reward learning



Paul Christiano [Follow](#)

Mar 18, 2017

Here are the slides from a short presentation I gave recently, summarizing my views on imitation/IRL/human feedback. The talk was quite informal and conversational; unfortunately the slides don't include most of what was said.

RewardLearningPresentation
drive.google.com

Approaches to
Reward Learnin

Paul Christiano
OpenAI, UC Berkeley

(I also made no effort to cite anything or conform to typical norms of academic discussion.)

Benign model-free RL



Paul Christiano

[Follow](#)

Mar 19, 2017 · 10 min read

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.

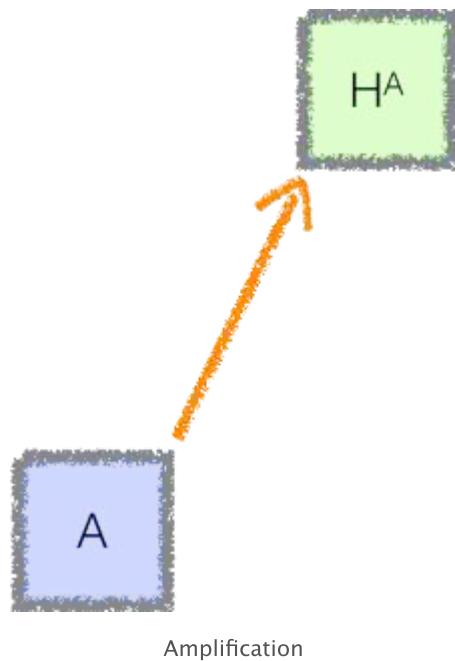
The capability amplification problem is to start with a weak agent A and a human expert H, and to produce a significantly more capable agent H^A . The more capable agent can take a lot longer to think, all we care about is that it *eventually* arrives at better decisions than A. The key challenge is 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.

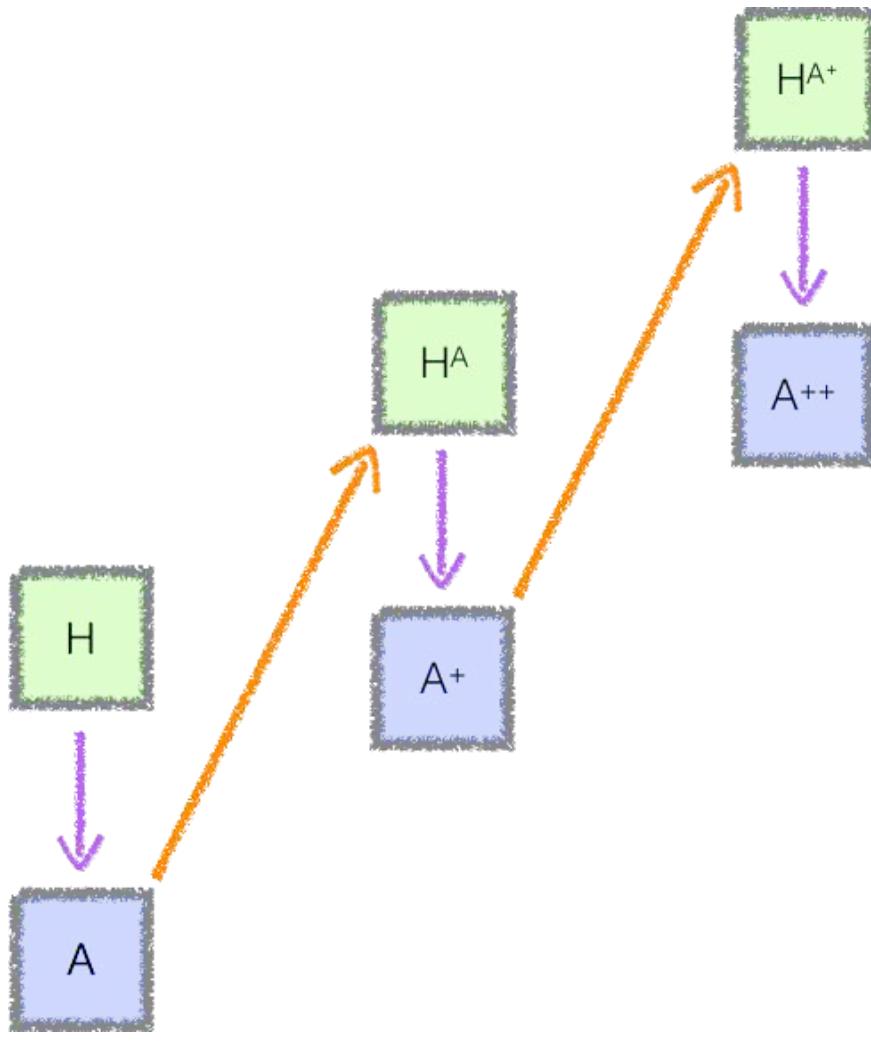
In general, meta-execution is my current preferred approach to capability amplification.

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:



Iterated amplification + reward learning

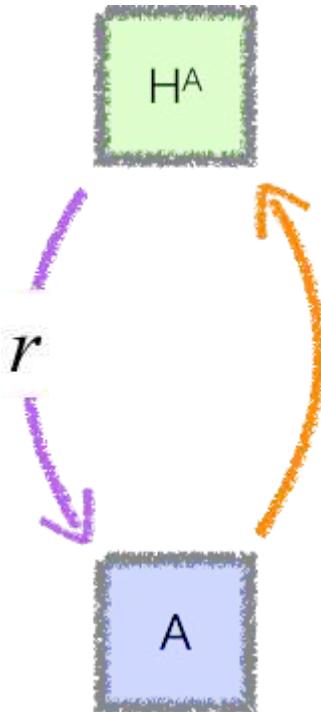
That is, we start with our benign expert H . We then learn a reward function and train an agent A , which is less capable than H but can run much faster. By running many instances of A , we obtain a more powerful agent H^A , which is approximately as expensive as H .

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:



An agent defines its own reward function

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:

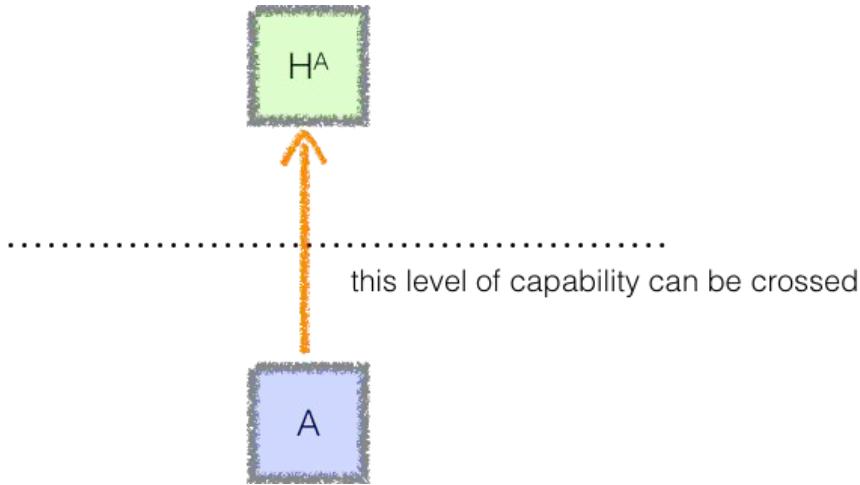
- The original expert H is benign by definition.
- If we start with a benign overseer H , and have working solutions to reward learning + robustness, then the trained agent A is benign.
- If we start with a benign agent A , and have a working solution to capability amplification, then the amplified agent 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 $10\% + 0.01\% \times 1000 = 20\%$.

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.

Corrigibility



Paul Christiano [Follow](#)

Jun 10, 2017 · 8 min read

(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 Arbitral) 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-based agent 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.

Approval-maximizing representations



Paul Christiano [Follow](#)

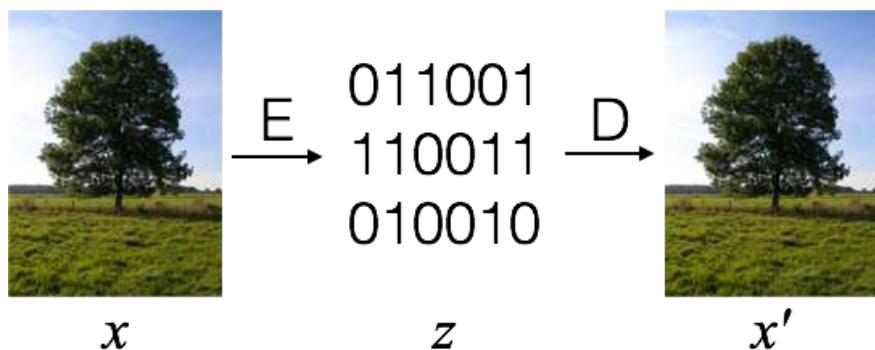
Jun 17, 2017 · 7 min read

(Follow-up to learning representations.)

A long-lived AI will consume a lot of data, and needs to make hard choices about what to store and how to represent it. This is problematic for an act-based agent, where these decisions must be made or evaluated by an overseer.

We might wonder: can act-based agents ever learn to use representations that are incomprehensible to humans? If they can't, it would probably prevent them from achieving state-of-the-art performance.

As a warm-up, consider the case where we observe an image, and want to store a compressed version which we can recall on a future day. That is, we want to learn an encoder $E : (\text{image}) \rightarrow (\text{code})$ and a decoder $D : (\text{code}) \rightarrow (\text{image})$.



Schematic of an autoencoder

How should we train these maps?

Even if the overseer can't understand the code at all, we can train the encoder and decoder jointly. That is, we start from an image x , obtain the reconstruction $x' = D(E(x))$, and ask: "is x' a good reconstruction of x ?"

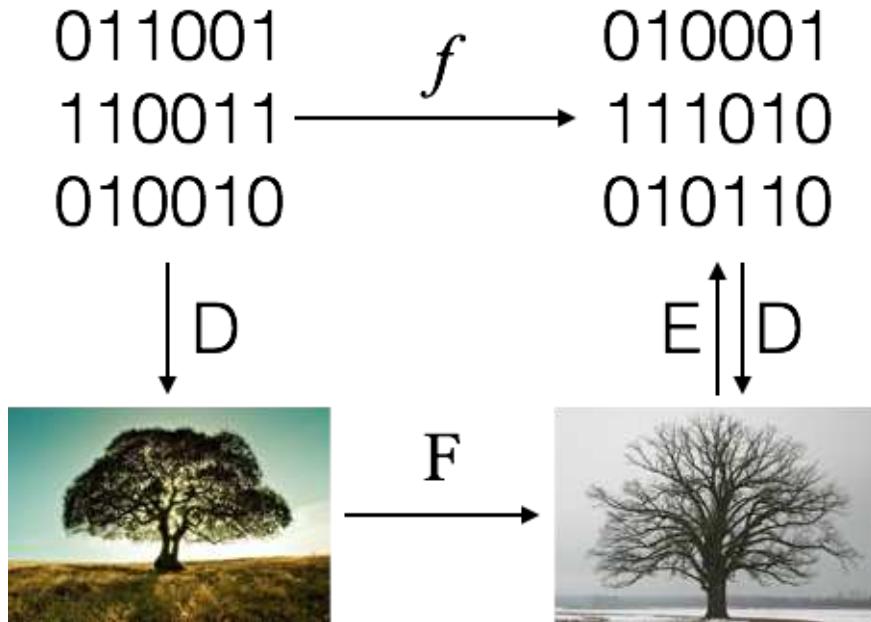
The overseer's responses define a loss function $L(x, x')$, and we can then train D and E to minimize this loss, subject to a constraint on

the size of the representation z . Such an autoencoder can learn to throw out details that the overseer doesn't expect to be useful.

Manipulating representations

Now suppose that we want to learn to *manipulate* codes. For example, I may have a target transformation on natural inputs like "change the season from winter to summer." I'd like to train an agent to implement the analogous transformation on codes.

That is, suppose that the overseer has a map $F : (\text{natural input}) \rightarrow (\text{natural output})$, and an approval function $L : (\text{natural input}) \times (\text{natural output}) \rightarrow [0, 1]$. This is the data that we need to train an agent to manipulate natural inputs using imitation+RL.



By following D , then having the overseer apply F , then applying E , we can produce training samples for f . Similarly, by applying D to both inputs and outputs, we can elicit the overseer's feedback on f .

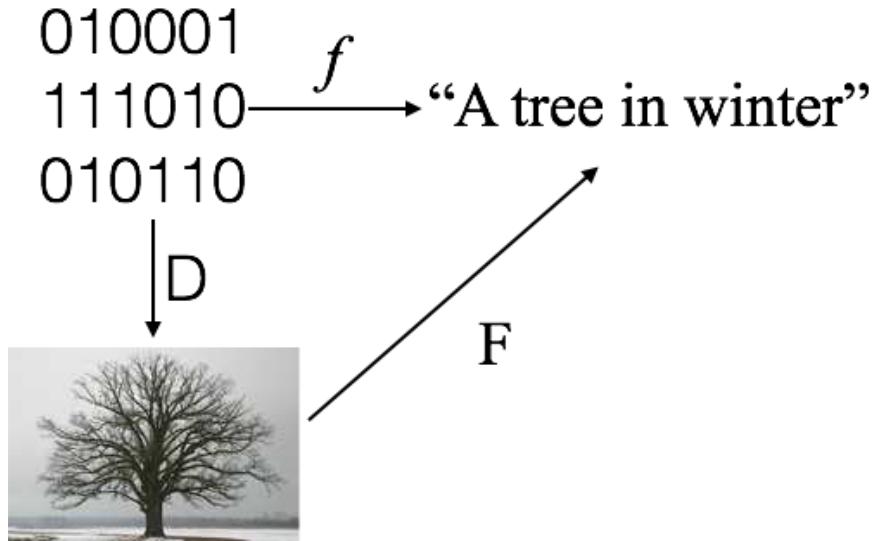
If we have an encoder/decoder, we can use them together with F and L to train a map on codes. To get examples we can evaluate $E(F(D(z)))$. To get evaluations, we can compute $L(D(z), D(f(z)))$.

So the overseer has lost nothing by working with these representations—they are able to provide feedback as well as if they understood the representations natively.

We can also learn representations which are computationally useful in addition to being compact, by training f jointly with the

encoder/decoder. A more convenient representation will make it easier to learn f .

At the end of the day, our system also interacts with the world by taking natural inputs (e.g. instructions in natural language, or images from a camera) and producing natural outputs (e.g. utterances, or images displayed on a screen).



$f(z) \sim F(D(z))$, which we can use both to sample and evaluate input/output pairs.

We can train these functions in exactly the same way, by translating to/from codes using our encoder or decoder.

If we were willing to train our system entirely end-to-end then we don't need to deal with the encoder/decoder at all and could just demonstrate/evaluate the natural inputs and outputs. The encoder/decoder are intended to allow the overseer to evaluate individual operations on codes, so that they don't need to evaluate the entire process from start to finish. This is important because our agents might perform very long sequences of operations, which won't be practical to train end-to-end. For example, an agent might want to use an observation months after making it.

Iterative encoding

In general, transforming an efficient representation into a human-comprehensible representation will increase its size.

For example, a comprehensible representation might be a description in English, while an efficient representation uses a richer language containing concepts that humans don't understand. Translating the

rich language into English may greatly increase its size—perhaps a single word of the rich language can only be explained with many paragraphs of English text.

In general, unpacking an efficient representation could lead to exponentially large comprehensible representations, which would render the schemes from the last section impractical.

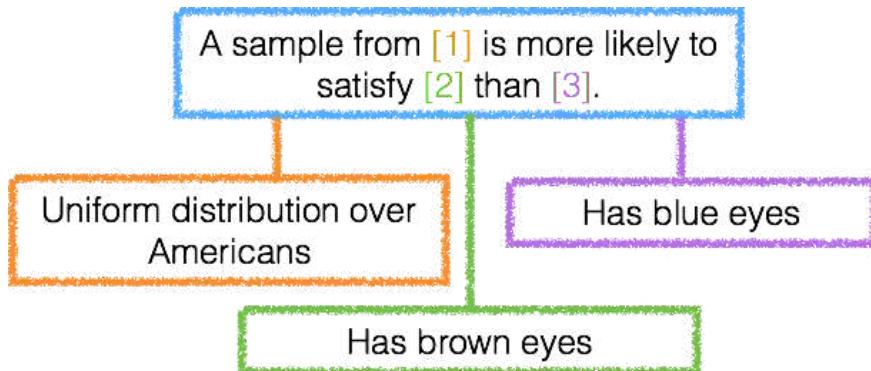
We can potentially address this problem by working with a sequence of increasingly complex representations.

Compound representations

The first ingredient is building a large representation out of smaller parts.

We'll start with some space of “simple” representations S , such as snippets of English text that can be read and understood in 5 seconds. We'd like to build more complex representations out of S .

This could be approached in many equally good ways, but I'll use the idea of *messages* defined in meta-execution.



A compound representation composed out of simpler representations.

A message over S consists of two parts:

- An element of S (the “head”)
- A list of additional messages over S (the “arguments”), that can be referenced by the head.

The semantics of messages are straightforward, and hopefully the example above makes it clear.

The size of a finite message is 1 plus the sum of the sizes of all of its arguments. Write $M(S)$ for the set of messages over S of size at most 1000.

$M(S)$ is a much bigger set than S , and can generally represent much more complex concepts.

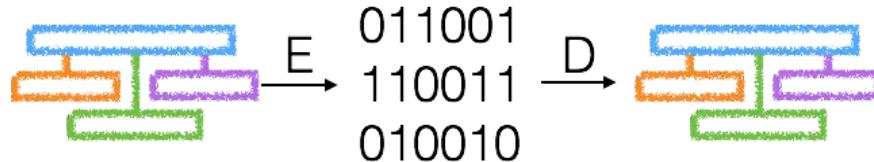
Compressing representations

By combining compound representations with representation learning, we can potentially construct a hierarchy of increasingly efficient representations.

Let S be some initial representation which is comprehensible to the human overseer H . An agent A trained by H would naturally operate on S .

Now we'd like to train agent A^+ which is smarter than A and operates on a more efficient representation S^+ .

If we use an amplification scheme like meta-execution, then we can turn A into a smarter agent H^A which operates naturally on $M(S)$. We can then use H^A to train an autoencoder which compresses $M(S)$ into a new representation S^+ :



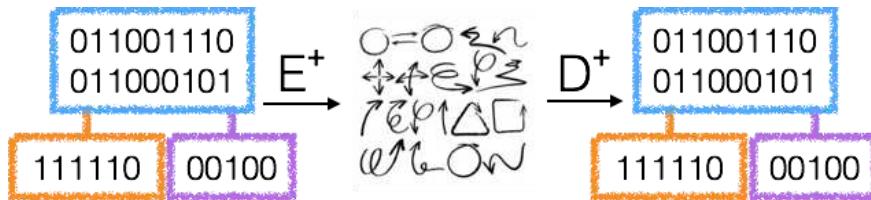
An autoencoder which translates compound representations into a more efficient encoding. The empty boxes would be filled in with elements of S , and the binary code belongs to S^+ .

In general this encoder can't preserve all of the information in every compound representation. But it can preserve the most important details, and may be able to substantially compress the representation.

We can train this autoencoder jointly with a new agent A^+ who operates on S^+ , in order to obtain representations that are computationally convenient. With the encoder in hand, H^A can effectively oversee A^+ .

Iteration

Now we have an agent A^+ which operates on a more efficient representation S^+ . We can repeat the process: H^{A^+} can train a new agent A^{++} which operates on a still more efficient representation S^{++} :



An autoencoder which translates $M(S^+)$ into a still-more-efficient encoding S^{++} .

In this picture, E^+ takes as input a message over S^+ (represented as binary) and outputs a symbol in S^{++} (represented as squiggles).

H^{A^+} is able to operate on messages over S^+ since it is able to call A^+ many times, and so it can train E^+ and D^+ . And with the encoder in hand, H^{A^+} can train a new agent A^{++} which operates on S^{++} .

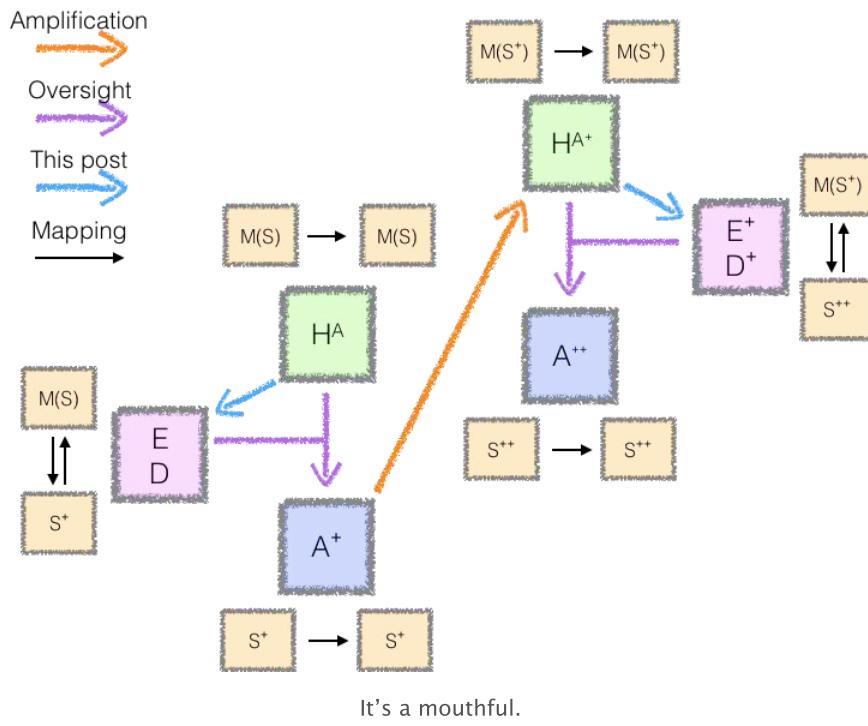
Note that while it is possible to convert S^{++} back to a message over S , the size may grow considerably. An element of S^{++} can be unpacked to a message of size 1000 over S^+ . Each of these thousand elements of S^+ can be unpacked to a message of size 1000 over S . And so an element of S^{++} would unpack into a message of size 1000000 over S . This size will keep growing exponentially as we iterate.

Putting it all together

I've been somewhat cavalier when talking about bootstrapping schemes for aligned RL, and haven't discussed the representations they would use. I've swept this under the rug by assuming that the resulting agents operate in an episodic environment with a well-defined beginning and end and that all information stored between episodes is human-comprehensible.

The primary purpose of this post is to patch up these oversights, and to move towards removing the episodic assumption altogether.

So what does it look like if we combine this system with this bootstrapping scheme, putting together the steps from the preceding sections?

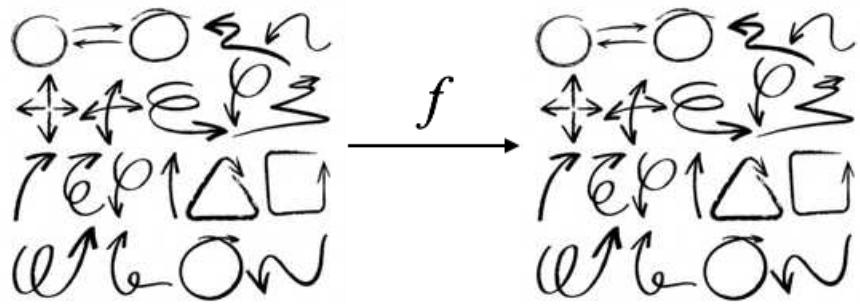


This is a copy of the diagram from the previous post, but with the following changes:

- I've annotated each policy with its domain and range. As before, S , S^+ and S^{++} are increasingly efficient representations, and $M(X)$ is the space of small messages over X . For convenience, I'm assuming that each of S , S^+ , S^{++} also contains fixed representations of the object-level inputs and outputs of our agents.
- I've inserted encoders E and E^+ , and decoders D and D^+ . These are trained using the mechanism described in this post. They are needed to provide oversight to A^+ and A^{++} , since H^A and H^{A+} aren't capable of directly understanding S^+ or S^{++} .

Conclusion

Ultimately, efficient AI systems will act on compact representations which will be incomprehensible to humans. If we want to build act-based agents, we need to connect these representations to something that the overseer can understand.



This might be achievable by training a hierarchy of increasingly efficient representations in parallel with training a hierarchy of increasingly competent agents. We can learn richer representations by training an autoencoder to compress compound representations built up of simpler pieces.

It is completely unclear if this proposal is realistic, and this post hasn't touched on any of the practical difficulties. But it suggests to me that there isn't an inherent philosophical problem with act-based training of agents to operate on efficient, incomprehensible representations.

AlphaGo Zero and capability amplification



Paul Christiano

[Follow](#)

Oct 19, 2017 · 3 min read

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 \mathbf{p} is trained to predict what AlphaGo will eventually decide to do.
- A value function \mathbf{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 its moves by using 1600 steps of Monte Carlo tree search (MCTS), using \mathbf{p} and \mathbf{v} to guide the search. It trains \mathbf{p} to bypass this expensive search process and directly pick good moves. As \mathbf{p} improves, the expensive search becomes more powerful, and \mathbf{p} chases this moving target.

Iterated capability amplification

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

- A “weak” policy \mathbf{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 \mathbf{p} directly to pick moves, we don’t use the weak policy \mathbf{A} directly to pick actions. Instead, we use a capability amplification scheme: we call \mathbf{A} many times in order to produce more intelligent judgments. We train \mathbf{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 (\mathbf{p}, \mathbf{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.

Techniques for optimizing worst-case performance



Paul Christiano

[Follow](#)

Feb 1, 2018 · 10 min read

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 make 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 we're 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.

Universality and security amplification



Paul Christiano

[Follow](#)

Mar 10, 2018 · 10 min read

If many copies of Paul collaborate to solve a problem, I expect their behavior to be corrigible: each copy will make an honest effort to help the group accomplish its goal, to help the group notice and correct mistakes, and so on.

Sometimes by chance one copy may behave incorrigibly. For example, one in every million days I might be struck by an irresistible urge to watch the world burn. Reliability amplification is the problem of combining several probably-corrigible agents into a more-corrigible ensemble (which only wants to destroy the world on one in every *billion* days). I think reliability amplification is probably possible, because a large ensemble of sufficiently-corrigible agents is very unlikely to have more than one incorrigible member.

Unfortunately, some situations might cause me to consistently behave incorrigibly. Let's call such a situation an "attack." For example, a clever argument might convince a copy of Paul that he should sabotage the collaboration. If we expose ten copies of Paul to the same argument it won't fix the problem, we'll just have ten copies all of whom are behaving incorrigibly.

Security amplification is the problem of combining many corrigible agents, into a group which is harder to attack than any individual. I hope to iterate security amplification and end up with a secure system.

This post explains why I think security amplification is likely to be possible.

Setting

I'll be thinking about dialog systems that communicate over a text channel. A "query" is anything that you can ask interactively over a text channel.

In this context, corrigible behavior means behaving cooperatively to the person on the other end of the conversation—being honest,

volunteering helpful information, following their instructions, etc.

An “attack” is a sequence of messages which cause an agent to stop behaving cooperatively.

I. Breaking it down

The basic hope

A large group can answer a big query by breaking it down into less complex pieces. Let’s suppose we had some formal measure of complexity, and that we can answer a query of complexity k by breaking it into pieces each of which has complexity $< k$, until we reach some set of “foundational” queries of complexity k^0 that are too simple to break down any further.

This suggests a route to security amplification:

- Let k^Δ be the complexity of the *simplest possible attack* on an individual.
- We hope that $k^\Delta > k^0$. That is, we hope that the foundational questions are simple enough that all of them are safe.
- Then if we give any input of complexity k^Δ to the group, it will be broken down into pieces of complexity $< k^\Delta$ (which are necessarily safe).
- Therefore the complexity of the simplest attack *on the group* is higher than the complexity of the simplest attack *on the individual*.
- If we iterate this process, building a group out of groups, then the simplest attack will get more and more complex, until eventually there are no attacks at any feasible level of complexity.

In order to realize this hope, we need:

1. to define a procedure for breaking queries down,
2. to define an appropriate notion of complexity,
3. to argue that all queries above some foundational complexity k^0 are broken down into strictly simpler pieces,
4. to argue that all queries below complexity k^0 are safe,

5. to argue that the group behaves corrigibly as long as all of the individuals behave corrigibly.

1. Meta-execution

Meta-execution can potentially answer a query Q without any individual needing to consider anything as complex as Q itself.

I've sketched out part of an example here, to give some sense of what this actually looks like.

2. Defining complexity

I'll start by imagining what would happen if you asked me to pick a query at random.

By "at random" I'm imagining something like the benign universal distribution relative to Paul: I think of all of the possible not-actively-evil strategies I could use for picking queries at random, and then I pick one of those *strategies* at random.

For example, my meta-strategy might be: with 5% probability I give a uniformly random string. With 2% probability I think of events in my life and ask a question relevant to one of them. With 15% probability I try splitting up the space of possible questions into a hierarchy, and flip a coin at each step to decide which part of the hierarchy to explore. With 80% probability I do something else.

One of these strategies is particularly important: I could generate an intermediate random query Q' , and then let Q be a random query that arises in the process of answering Q' . For concreteness, let's say that I use this strategy half of the time.

Let $\mu(Q)$ be the probability that this process outputs query Q . We can define the complexity of a query Q as $k(Q) = -\log \mu(Q)$.

We can also use μ to measure the complexity of a distribution η over queries, by letting $k(Q)$ be the maximum value of $\log(\eta(Q)/\mu(Q))$ for any query Q .

Informally, this is the log of "How many samples would you have to draw from μ , in order to get one sample from η ? ", i.e how many bits of selection pressure you need to get a sample from η .

(It may be more useful to consider the ε -smooth complexity: what is the *smallest* complexity of any distribution η' that is within ε total

variation distance of η ? Informally: how many samples would you have to draw from μ , in order to get a sample ϵ -close to η ?)

Note: There's no way that this is the right definition to use and I expect we'll be able to do better. I think an important ingredient is having a decomposition where the (subquestion \rightarrow question) graph is a small set expander. If anyone wants to take up this analysis, I think it's an important question, I expect to come back to it at some point.

3. Can we break a query into simpler parts?

Suppose that I start with a query Q , and split off a subquery Q' . When can we hope that $k(Q') < k(Q)$?

Suppose there are many possible queries Q for which Q' appears as a subquery. For example, the subquery “when did [a] happen?” might appear in the course of answering “was [a] a cause of the Great Depression?”, or it might occur in the course of answering “was [a] a consequence of the Spanish flu?”

Each of these corresponds to a way that Q' could be sampled by μ —since one of the strategies in μ is to generate a random query and then generate a random subquery.

So $\mu(Q') > \sum \mu(Q_i) / 2N$, where $\{Q_i\}$ is the set of superqueries that give rise to subquery Q' , N is the total number of subqueries for each superquery, and 50% is how often μ decides to sample a random subquery.

As long as each possible superquery has at most $1/(2N)$ of the total probability under μ —i.e., as long as you can't guess the superquery with reasonable probability from the subquery—then we have reduced the complexity. If N is 16, this requires that each subquery must throw out about 5 bits of information about the original query.

In fact the situation is better than this:

- In reality our queries are sampled from some distribution η rather than chosen deterministically. As we increase the entropy of η the set of preimages $\{Q_i\}$ effectively becomes larger. (The analysis also gets a little bit more complicated.)
- μ mixes in strategies other than (pick a random query, then pick a random subquery). If Q' is qualitatively simpler than Q , then

we may have $\mu(Q') > \mu(Q)$ without even considering the possibility of Q' being sampled as a subquery of Q .

4. Are simple queries secure?

What is the complexity of the simplest attack on an unaided human? Equivalently: how many queries do we expect to sample, before we find an attack?

This will vary from human to human, so for concreteness I'll consider a thoughtful futurist or AI researcher who has been briefed about all of the issues we can think of.

If we wanted to actually test for the existence of attacks, we could conduct a giant brute force search for “simple” attacks.

If we imagine generating a query every 30 seconds for 40 hours a week for a year, we generate about 200,000 queries a year. Hiring ten people gets us up to 2M queries a year. 30 seconds may be low, but note that the generation process can produce many samples per invocation, and we only need to test the candidate attacks.

If we put in a little bit of selection pressure I suspect we would find attacks at least 1–3 orders of magnitude more often than by sampling randomly.

So it seems easy to test explicitly whether 100 million random queries contain an attack. This could be done by dedicated red teams, and mixed freely with the process used to collect training data. I suspect that thoughtful analysis could get us even further.

Intuitively, I would be surprised if such a search turned up an attack against a careful overseer. If it did turn up attacks, I suspect it would be possible to strengthen the overseer against them.

Overall, I feel pretty safe relying on 10^8 queries not containing an attack, pretty unsafe with numbers like 10^{30} , and am not sure where in between attacks would crop up.

5. Is a collaboration of corrigible agents corrigible?

Even if every individual behaves corrigibly, the group need not behave corrigibly. This is a bit of a tangent from the real thrust of this post, which is about removing the security failures of the human. But

it's a tangent worth discussing briefly, since I think it's an important open question.

There are two very salient sources of incorrigibility (though without a crisper picture of optimization and corrigibility this list isn't exhaustive):

- The group is applying selection pressure to patterns and behaviors propagating through the group. Those patterns may themselves become powerful incorrigible consequentialists.
- The group is implementing incorrigible optimization.
For example, suppose meta-execution asks the subquestion "What does the user want?", gets a representation of their values, and then asks the subquestion "What behavior is best according to those values?" I've then generated incorrigible behavior by accident, after taking innocuous steps.

I think the first problem is likely resolvable by applying security and reliability amplification, which operate on groups as well as individuals. I think the biggest thing that's missing here are tools for analysis (though maybe once we have those tools we'll notice more problems). Currently this feels pretty complicated and confusing to me.

The second problem clearly requires the human operator to do something other than just "being sensible." That is, in order for this scheme to possibly work, the human needs to understand something about what kind of optimization causes incorrigibility and the human would have to avoid actions that introduce problematic optimization. It's conceivable that this can be done by an existing human if they are paranoid, but realistically the only reason I expect this to go well is because I expect us to make progress on the theoretical issue of understanding incorrigibility. I think this is an important theoretical problem.

II. Universality

So far I've avoided a key question: why think we can break tasks down at all?

Indeed, some human capabilities *can't* be broken down into pieces. Suppose that a human has seen a bunch of examples $(x, f(x))$ —such as English phrases and their French translations—and has learned a predictive model off. The human may not be able to access their

model except by running it, in which case they have no hope of breaking down the task of computing f —in this case translating a sentence. (Wei Dai proposed the translation example.)

The proposal

I propose to go on breaking tasks down anyway. This means that we will lose certain abilities as we apply amplification.

For example, given the task “Translate the sentence $[x]$ from French to English” we will answer it without having any translator look at the entire sentence x . This means that the quality of translation will fall.

After enough steps of amplification, we may eventually arrive at an agent that doesn’t know French at all, and is stuck with recommendations like “Consult an English-to-French dictionary.”

Effectively, this proposal replaces our original human overseer with an impoverished overseer, who is only able to respond to the billion most common queries.

Is this OK?

The first key question is whether this impoverished overseer remains universal.

That is, if we put together enough copies of this impoverished overseer (by iteratively apply meta-execution) would we be able to obtain arbitrarily smart groups? Or would we get stuck?

Here we need to be careful about “arbitrarily smart.” There are clearly problems the group will never be able to solve—due to lacking knowledge/expertise— including problems that individual humans *can* solve.

This is potentially OK, as long as we learn a good policy for leveraging the information in the environment (including human expertise).

This can then be distilled into a state maintained by the agent, which can be as expressive as whatever state the agent might have learned. Leveraging external facts requires making a tradeoff between the benefits and risks, so we haven’t eliminated the problem, but we’ve potentially isolated it from the problem of training our agent.

Comparison to agent foundations

If the impoverished overseer is universal, then the set of questions of complexity $k < k^0$ form a simple “core” for reasoning: by creating a giant lookup table of human responses to these questions and simply using that lookup table enough times, we can produce arbitrarily sophisticated behavior.

If humans are universal at all then of course such a core exists (just take all questions that a human can articulate in their lifetime). But finding a small core seems like it requires a better understanding of intelligence.

I think that coming up with such a core is a very natural and important problem for researchers interested in philosophical issues in AI. My view is that if MIRI-style research adds value, it is most likely to be by finding an explicit core for reasoning rather than finding an explicit recipe for AGI. This core would then be combined with iterated amplification to yield a competitive AI. However, I don’t think that such a core is likely to encode an answer to questions like “what is the right decision theory to use?”—instead I expect it to look more like a solution to metaphilosophy, automating the process whereby humans answer questions of that form.

Conditioned on amplification working well, I think there is about a 50% chance that it uses an explicit core that we understand and a 50% chance that it uses a messy core learned from humans.

In addition to making it much easier to analyze amplification, having an explicit core of reasoning would also potentially make verification much easier, as discussed here. Overall, I think that this kind of perspective might capture most of what is useful about the MIRI view while still being able to leverage the benefits of modern ML.

An unaligned benchmark



Paul Christiano

[Follow](#)

Mar 20, 2018 · 11 min read

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 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.)

Clarifying “AI alignment”



Paul Christiano

[Follow](#)

Apr 7, 2018 · 4 min read

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 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.

Two guarantees



Paul Christiano

[Follow](#)

Apr 9, 2018 · 6 min read

When I imagine proving something about an AI, or making an inductive argument about amplification, I think about two different guarantees:

1. The AI behaves well *on average* over some particular distribution of inputs. (The performance guarantee.)
2. The AI never does anything “actively malicious,” on *any* input. (The control guarantee.)

Definitions

We can define “behaves well on average” by looking at the average impact of decisions on our utility function. This post addresses some subtleties with that definition, including inconsistent preferences, different scales of impact, and long-term consequences. Actually evaluating this quantity, so that it can be used in training, requires amplification and informed oversight.

My intuition about “actively malicious” is best captured by the (informal) idea of incorrigible optimization: our AI should never be actively trying to undermine our understanding or control of the situation. This property seems “easy,” in the sense that it can be satisfied without any domain knowledge or expertise, which makes me optimistic that it is possible to satisfy it for every input.

Rationale

Rationale for the performance guarantee

The performance guarantee is what does the work; it’s the reason you’d want an AI instead of a pet rock.

If we construct the reward so it reflects the difference in utility between different actions, then the average case is actually a pretty good thing to optimize.

In reality the data distribution is constantly changing over time. So we want an online guarantee, maximizing the average case performance over a varying sequence, rather than a batch guarantee. Normally deep learning is designed for the batch setting under the iid assumption, and data distributions change slowly enough that this is acceptable. Overall, I don't think this distinction is a huge deal.

Note that the "average case" guarantee still reflects the overseer's desire for corrigibility. This is formally reflected when (a) the overseer evaluates the expected utility impact of a decision by marginalizing over all of the mistakes that the AI might make in the future, (b) the overseer evaluates individual decisions by assigning a large penalty for incorrigibly doing the wrong thing, but only a small penalty for behaving corrigibly but slightly inefficiently.

Rationale for control guarantee

Unfortunately, it is only possible to get an average performance guarantee when the rewards are small relative to the total amount of data. If a few bad decisions can destroy all of the value from a lifetime of good decisions, then traditional ML can't do well on average.

If an AI is *trying* to cause a catastrophe, it can do so very quickly even in apparently benign situations (by breaking the mechanisms that humans use to remain in control of the situation). The goal of the control guarantee is to prevent this kind of catastrophe.

Similarly, if an AI is trying to be difficult, it could intentionally undermine whatever testing regime we use (by failing selectively on inputs that don't look like tests). The control guarantee should also rule out this kind of intentional sabotage.

Hopefully the control guarantee cuts off the worst catastrophes, making the performance guarantee meaningful.

Other catastrophes

Even if our AI satisfies the control guarantee, it can make other kinds of catastrophic mistakes (just as an aligned human could make a catastrophic mistake). For example, the system might be in charge of running a nuclear defense system, and a single failure could lead to a nuclear war. Or an attacker might fool the AI into executing an instruction by impersonating the operator, which in turn open the system to further attack.

Neither the control nor performance guarantee directly address these problems. Instead, anyone who deploys an AI needs to be aware of the system's limits, to test the system to see where it might fail, to design mechanisms with redundancy, to protect the system from attackers, and to avoid incorrectly assuming perfect performance. (The same measures we would take if delegating to a human who sometimes made mistakes.)

Corrigibility

The performance and control guarantees interact to create a system that is corrigible: the performance guarantee ensures the system is typically trying to give the human more effective understanding and control over the situation. The control guarantee ensures the system isn't undermining those measures, for example by constructing a fake narrative for the human or illusory control, by leaving backdoors that can be exploited, etc.

The performance guarantee leaves open the possibility that the system will sometimes fail to inform or empower the human effectively. But as long as those failures aren't optimized to be unrecoverable, it seems the human can work around them by recognizing the shortcoming and having the AI optimize for the robustness of human control and understanding.

I think there is a meaningful analogy between this picture and this post about monitoring and whistleblowing, and I have a vague intuition that there is some important underlying dynamic that could be better understood.

Amplification

Amplification and performance

Amplification solves a task by (adaptively) breaking it into several subtasks, solving the subtasks, and combining the results.

If we are making an inductive argument about amplification, then the performance guarantee implies the expected *average* performance on subtasks is good, and requires that the expected performance on the overall task is good.

If some of the subtasks make "honest mistakes," then I think this is a reasonable hope.

In order to make it work we definitely need to have an unequal weighting over subtasks, which corresponds to an unequal sampling of subtasks during training. For example, if we are trying to figure out if any of {A, B, C} are positive, we proceed by independently estimating A, B, C, and we conclude that B is positive, then the estimation of B was a more important subtask than the estimation of A or C.

Even with arbitrary weights on subtasks, preserving average performance might seem too optimistic. In the previous example, if we conclude that all of A, B, C are negative, then any one of the three errors could lead to an erroneous overall conclusion. So if our original error rate was 5%, then our new error rate will be 15%.

In cases like this, it's wrong of think of "error rate" as the measure of performance that should be preserved inductively. Some questions are harder than others, and should be expected to have higher error rate—if any rational agent tries to figure out whether any one of N quantities is positive, without any clever trick for figuring it out, then they'll have a harder time than if they were just trying to estimate whether a single quantity is positive. This makes it harder to talk precisely about the guarantees of amplification, but leaves me feeling relatively optimistic about eventually finding a working analysis.

Amplification and control

I find the above intuition plausible when errors are "honest mistakes." If instead one of the subtasks is optimized to cause trouble, then I think it is very hard (and likely impossible) to correct the problem. So in order to have any hope of maintaining performance, we need to guarantee that none of the subtask solutions are actively malicious.

But in order to guarantee that a subtask isn't actively malicious, we also need to assume that none of *its* subtasks are actively malicious. If we want to carry out the induction to a reasonable depth, this basically requires a worst-case guarantee. Otherwise even a single failure, in one leaf of an exponentially large tree, could cause trouble.

Unfortunately, a human probably does not satisfy the control guarantee, since they may behave very strangely on some small subset of possible inputs or with small probability. I originally described security amplification and reliability amplification to gradually eliminate the human control failures (as well as new

failures that crop up, e.g. from memetic selection inside the amplification process).

Interestingly, some ways of achieving the control guarantee in the distillation step could also be applied even if the overseer *doesn't* satisfy the control guarantee, and so could be used as a replacement for security/reliability amplification. This isn't as crazy as it sounds, since the ability to look at the entire code of a learned model gives us a substantial advantage over dealing with an opaque overseer. It gives a further reason to prioritize techniques for worst-case performance (and particularly interpretability).

Conclusion

We can't guarantee an AI is aligned if we have only an average-case guarantee, *or* only a worst-case guarantee. So achieving both seems like the "minimum viable product" for alignment research.

My original intuition (in mid-2016) was that having two separate guarantees must be an ugly hack, and that the real goal should encapsulate both. That's no longer so clear to me: I think these two properties interact surprisingly nicely, such that they may actually suffice to get good behavior even though it looks like a weird combination. At the same time, I think attempts to capture both are much less promising than I'd initially believed.

I still think we need more clarity about what we should be trying to prove. I think that having two separate guarantees, one for the worst case and one for the average case, is the current best guess and is the most promising starting point for further research. In the short term, my focus will be on understanding ways in which this structure is inadequate and on independently refining each of these two subgoals.

Implicit extortion



Paul Christiano [Follow](#)

Apr 13, 2018 · 8 min read

In this post I describe a pattern of behavior I call “implicit extortion.” RL agents are particularly susceptible to implicit extortion, in a way that is likely to be problematic for high-stakes applications in open-ended strategic environments.

I expect that many people have made this point before. My goal is just to highlight the issue and to explore it a little bit more carefully.

Basic setup

Consider two actors, the target (T) and manipulator (M), such that:

- M wants T to perform some *target action*—e.g. make a payment, leak information, buy a particular product, handicap itself...
- M can take *destructive actions* that hurts both M and T—e.g. spreading rumors about T, undercutting T in a marketplace, physically attacking T...

In *explicit extortion*, M threatens to take the destructive action unless T performs the target action. Then a naive T reasons: “if I don’t take the target action, something bad will happen, so I better take the target action.”

In *implicit extortion*, M simply performs the destructive action whenever T doesn’t perform the target action. Then a naive T eventually learns that failure to take the target action is associated with something bad happening, and so learns to take the target action.

Implicit extortion is very similar to explicit extortion:

- T would prefer not be the kind of person who is vulnerable to extortion, so that bad things don’t happen to them.
- Extortion doesn’t necessarily cost M very much, if they don’t follow through on the threat very often.

However, implicit extortion can be particularly hard to avoid:

- It can be effective without T realizing that it's happening, which makes it hard for them to respond appropriately even if they do have defenses.
- It affects simple RL algorithms (which don't have defenses against extortion, and can't be easily modified to include such defenses).

Example

The most extreme and blatant example would be for M to send T a daily request for \$100. On any day when T fails to pay, M launches a costly cyberattack against T. A human would immediately recognize this behavior as extortion and would respond appropriately, but an RL algorithm might simply notice that paying is the best strategy and therefore decide to pay.

Implicit extortion can be much harder to detect, while still being effective. Suppose that every time T tries to change their product, M runs a grassroots smear campaign. It might not be possible for T to distinguish the situations “M is attempting to manipulate me into not changing my product” and “Every time I change the product people get really unhappy, so I should do so sparingly.”

Details

How expensive is this for the manipulator?

Suppose that T is using an RL algorithm, and M is trying to manipulate them. How expensive is this for M? How likely is it to be worthwhile?

At equilibrium: T learns to always perform the target action; so only fails to take the target action while exploring. The long-term cost to M depends entirely on the target's exploration policy.

If T uses ϵ -exploration, then they take the target action $(1 - \epsilon)$ of the time. So M only needs to pay the cost of the destructive action on an ϵ fraction of trials.

For complex high-level actions, the effective ϵ can't be *too* high—it's not a good idea to “try something crazy” 10% of the time just to see what happens. But let's be conservative and suppose that $\epsilon=0.1$ anyway.

Suppose that M is trying to directly extract money from T, \$10 at a time, and that it costs M \$50 of value in order to cause \$15 of trouble for T.

If M asks for \$10 on 10 occasions, T will refuse to pay only once as an exploration. Then M needs to pay that \$50 cost only once, thereby ensuring that the cost of paying ($=\$10$) is smaller than the average cost of refusing to pay ($=\$15$). Meanwhile, M makes \$90, pocketing \$40 of profit.

In general, M can make a profit whenever the product of (payment efficiency) * (destructive efficiency) $> \epsilon$, where “payment efficiency” is the benefit to M divided by the cost to T of the target action, and “destructive efficiency” is the cost to T divided by the cost to M of the destructive action.

In practice I think it's not too uncommon for payment efficiency to be ~ 1 , and for destructive efficiency to be > 1 , such that extortion is possible regardless of ϵ . Small values of ϵ make extortion considerably easier and more cost-effective, and make it much harder to prevent.

During learning: the analysis above only applies when the agent has already learned to consistently take the target action. Earlier in learning, the target action may only occur rarely and so punishment may be very expensive. This could be worth it over the long term but may be a major hurdle.

Fortunately for M, they can simply start by rewarding the target behavior, and then gradually shift to punishment once the target behavior is common. From the perspective of the RL agent, the benefit of the target action is the same whether it's getting a reward or avoiding a punishment.

In the cash payment example, M could start by paying T \$20 every time that T sends \$10. Once T notices that paying works well, M can gradually reduce the payment towards \$10 (but leaving a profit so that the behavior becomes more and more entrenched). Once T is consistently paying, M can start scaling up the cost of not paying while it gradually reduces the benefits of paying.

Analyzing the error

Paying off a (committed) extortionist typically has the best consequences and so is recommended by causal decision theory, but

having the policy of paying off extortionists is a bad mistake.

Even if our decision theory would avoid caving in to extortion, it can probably only avoid implicit extortion if it recognizes it. For example, UDT typically avoids extortion because of the logical link from “I cave to extortion” → “I get extorted.” There is a similar logical link from “I cave to implicit extortion” → “I get implicitly extorted.” But if we aren’t aware that an empirical correlation is due to implicit extortion, we won’t recognize this link and so it can’t inform our decision.

In practice the target is only in trouble if would-be manipulators know that they are inclined to comply with extortion. If manipulators base that judgment on past behavior, then taking actions that “look like what someone vulnerable to extortion would do” is itself a bad decision that even a causal decision theorist would avoid.

Unfortunately, it’s basically impossible for an RL algorithm to learn to avoid this, because the negative consequences only appear over a very long timescale. In fact, the timescale for the negative consequences is longer than the timescale over which the RL agent adjusts its policy— which is too long for a traditional RL system to possibly do the credit assignment.

Other learning systems

What algorithms are vulnerable?

At first glance the problem may seem distinctive to policy gradient RL algorithms, where we take actions randomly and then reinforce whatever actions are associated with a high reward.

But the same problem afflicts any kind of RL. For example, a model-based agent would simply learn the model “not doing what the manipulator wants causes <bad thing X> to happen,” and using that model for planning would have exactly the same effect as using policy gradients.

More broadly, the problem is with the algorithm: “learn an opaque causal model and use it to inform decisions.” That’s an incredibly general algorithm. If you aren’t willing to use that algorithm, then you are at a significant competitive disadvantage, since the world contains lots of complicated causal processes that we can learn about by experiment but can’t model explicitly. So it seems like everyone just has to live with the risk of implicit extortion.

I describe the problem as afflicting “algorithms,” but it can also afflict humans or organizations. For example, any organization that is compelled by arguments like “X has always worked out poorly in the past, even though we’re not quite sure why, so let’s stop doing it” is potentially vulnerable to implicit extortion.

What about human learning?

Humans have heuristics like vindictiveness that help prevent us from being manipulated by extortion, and which seem particularly effective against implicit extortion. Modern humans are also capable of doing explicit reasoning to recognize the costs of giving in to extortion.

Of course, we can only be robust to implicit extortion when we recognize it is occurring. Humans do have some general heuristics of caution when acting on the basis of opaque empirical correlations, or in situations where they feel they might be manipulable. However, it still seems pretty clear that human learning is vulnerable to implicit extortion in practice. (Imagine a social network which subtly punishes users, e.g. by modulating social feedback, for failing to visit the site regularly.)

Evolution?

Evolution itself doesn’t have any check against extortion, and it operates entirely by empirical correlations, so why isn’t it exploited in this way?

Manipulating evolution requires the manipulator to have a time horizon that is many times the generation length of the target. There aren’t many agents with long enough time horizons, or sophisticated enough behavior, to exploit the evolutionary learning dynamic (and in particular, evolution can’t easily learn to exploit it).

When we do have such a large gap in time horizons and sophistication—for example, when humans square off against bacteria with very rapid evolution—we do start to see implicit extortion.

For example, when a population of bacteria develop resistance to antibiotic A, we take extra pains to totally eradicate them with antibiotic B, even though we could not afford to use that strategy if A-resistance spread more broadly through the bacteria population. This is effectively implicit extortion to prevent bacteria from developing A-resistance. It would continue to be worthwhile for humanity even if

the side effects of antibiotic B were much worse than the infection itself, though we probably wouldn't do it in that case since it's a hard coordination problem (and there are lots of other complications).

Conclusion

There are many ways that an AI can fail to do the right thing. Implicit extortion is a simple one that is pretty likely to come up in practice, and which may seriously affect the applicability of RL in some contexts.

I don't think there is any "silver bullet" or simple decision-theoretic remedy to implicit extortion, we just need to think about the details of the real world, who might manipulate us in what ways, what their incentives and leverage are, and how to manage the risk on a case-by-case basis.

I think we need to define "alignment" narrowly enough that it is consistent with implicit extortion, just like we define alignment narrowly enough that it's consistent with losing at chess. I've found understanding implicit extortion helpful for alignment because it's one of many conditions under which an aligned agent may end up effectively optimizing for the "wrong" preferences, and I'd like to understand those cases in order to understand what we are actually trying to do with alignment.

I don't believe implicit extortion is an existential risk. It's just another kind of conflict between agents, that will divert resources from other problems but should "wash out in the long run." In particular, every agent can engage in implicit extortion and so it doesn't seem to shift the relative balance of influence amongst competing agents. (Unlike alignment problems, which shift influence from human values to whatever values unaligned AI systems end up pursuing.)

When is unaligned AI morally valuable?



Paul Christiano

[Follow](#)

May 2, 2018 · 12 min read

Suppose that AI systems built by humans spread throughout the universe and achieve their goals. I see two quite different reasons this outcome could be good:

1. Those AI systems are aligned with humans; their preferences are our preferences.
2. Those AI systems flourish on their own terms, and we are happy for them even though they have different preferences.

I spend most of my time thinking about option #1. But I think option #2 is a plausible plan B.

Understanding how happy we should be with an unaligned AI flourishing on its own terms, and especially *which* unaligned AIs we should be happy about, seems like a very important moral question.

I currently feel very uncertain about this question; if you forced me to guess, I'd estimate that option #2 allows us to recover 25% of the expected value that we lose by building unaligned AI. But after more thinking, that number could go down to 0% or up to >90%.

Definition

In this post I'll say that an AI is a *good successor* if I believe that building such an AI and "handing it the keys" is a reasonable thing to do with the universe. Concretely, I'll say an AI is a good successor if I'd prefer give it control of the world than accept a gamble where we have a 10% chance of extinction and a 90% chance of building an aligned AI.

In this post I'll think mostly about what happens with the rest of the universe, rather than what happens to us here on Earth. I'm wondering whether we would appreciate what our successors do with all of the other stars and galaxies—will we be happy with how they use the universe's resources?

Note that a competent aligned AI is a good successor, because “handing it the keys” doesn’t actually amount to giving up any control over the universe. In this post I’m wondering which unaligned AIs are good successors.

Preface: in favor of alignment

I believe that building an aligned AI is by far the most likely way to achieve a good outcome. An aligned AI allows us to continue refining our own views about what kind of life we want to exist and what kind of world we want to create—there is no indication that we are going to have satisfactory answers to these questions prior to the time when we build AI.

I don’t think this is parochial. Once we understand what makes life worth living, we can fill the universe with an astronomical diversity of awesome experiences. To the extent that’s the right answer, it’s something I expect us to embrace much more as we become wiser.

And I think that further reflection is a really good idea. There is no law that the universe tends towards universal love and goodness, that greater intelligence implies greater moral value. Goodness is something we have to work for. It might be that the AI we would have built anyway will be good, or it might not be, and it’s our responsibility to figure it out.

I am a bit scared of this topic because it seems to give people a license to hope for the best without any real justification. Because we only get to build AI once, reality isn’t going to have an opportunity to intervene on people’s happy hopes.

Clarification: Being good vs. wanting good

We should distinguish two properties an AI might have:

- Having preferences whose satisfaction we regard as morally desirable.
- Being a moral patient, e.g. being able to suffer in a morally relevant way.

These are **not** the same. They may be related, but they are related in an extremely complex and subtle way. From the perspective of the long-run future, we mostly care about the first property.

As compassionate people, we don't want to mistreat a conscious AI. I'm worried that compassionate people will confuse the two issues—in arguing enthusiastically for the claim “we should care about the welfare of AI” they will also implicitly argue for the claim “we should be happy with whatever the AI chooses to do.” Those aren't the same.

It's also worth clarifying that both sides of this discussion can want the universe to be filled with morally valuable AI eventually, this isn't a matter of carbon chauvinists vs. AI sympathizers. The question is just about how we choose what kind of AI we build—do we hand things off to whatever kind of AI we can build today, or do we retain the option to reflect?

Do all AIs deserve our sympathy?

Intuitions and an analogy

Many people have a strong intuition that we should be happy for our AI descendants, whatever they choose to do. They grant the *possibility* of pathological preferences like paperclip-maximization, and agree that turning over the universe to a paperclip-maximizer would be a problem, but don't believe it's realistic for an AI to have such uninteresting preferences.

I disagree. I think this intuition comes from analogizing AI to the children we raise, but that it would be just as accurate to compare AI to the corporations we create. Optimists imagine our automated children spreading throughout the universe and doing their weird-AI-analog of art; but it's just as realistic to imagine automated PepsiCo spreading throughout the universe and doing its weird-AI-analog of maximizing profit.

It might be the case that PepsiCo maximizing profit (or some inscrutable lost-purpose analog of profit) is intrinsically morally valuable. But it's certainly not obvious.

Or it might be the case that we would never produce an AI like a corporation in order to do useful work. But looking at the world around us today that's *certainly* not obvious.

Neither of those analogies is remotely accurate. Whether we should be happy about AI “flourishing” is a really complicated question about AI and about morality, and we can't resolve it with a one-line political slogan or crude analogy.

On risks of sympathy

I think that too much sympathy for AI is a real risk. This problem is going to made particularly serious because we will (soon?) be able to make AI systems which are optimized to be sympathetic. If we are indiscriminately sympathetic towards whatever kind of AI is able to look sympathetic, then we can't steer towards the kind of AI that actually deserve our sympathy. It's very easy to imagine the world where we've built a PepsiCo-like AI, but one which is much better than humans at seeming human, and where people who suggest otherwise look like moral monsters.

I acknowledge that the reverse is also a risk: humans are entirely able to be terrible to creatures that deserve our sympathy. I believe the solution to that problem is to actually think about what the nature of the AI we build, and especially to behave compassionately in light of uncertainty about the suffering we might cause and whether or not it is morally relevant. Not to take an indiscriminate pro-AI stand that hands the universe over to the automated PepsiCo.

Do any AIs deserve our sympathy?

(Warning: lots of weird stuff.)

In the AI alignment community, I often encounter the reverse view: that *no* unaligned AI is a good successor.

In this section I'll argue that there are at least some unaligned AIs that would be good successors. If we accept that there are *any* good successors, I think that there are probably lots of good successors, and figuring out the boundary is an important problem.

(To repeat: I think we should try to avoid handing off the universe to any unaligned AI, even if we think it is probably good, because we'd prefer retain the ability to think more about the decision and figure what we really want. See the conclusion.)

Commonsense morality and the golden rule

I find the golden rule very compelling. This isn't just because of repeated interaction and game theory: I'm strongly inclined to alleviate suffering even if the beneficiaries live in abject poverty (or factory farms) and have little to offer me in return. I'm motivated to

help largely because that's what I would have wanted them to do if our situations were reversed.

Personally, I have similar intuitions about aliens (though I rarely have the opportunity to help aliens). I'd be hesitant about the people of Earth screwing over the people of Alpha Centauri for many of the same reasons I'd be uncomfortable with the people of one country screwing over the people of another. While the situation is quite confusing I feel like compassion for aliens is a plausible "commonsense" position.

If it is difficult to align AI, then our relationship with an unaligned AI may be similar to our relationship with aliens. In some sense we have all of the power, because we got here first. But if we try to leverage that power, by not building any unaligned AI, then we might run a significant risk of extinction or of building an AI that no one would be happy with. A "good cosmic citizen" might prefer to hand off control to an unaligned and utterly alien AI, than to gamble on the alternative.

If the situation were totally symmetrical—if we believed the AI was from *exactly* the same distribution over possible civilizations that we are from—then I would find this intuitive argument extremely compelling.

In reality, there are almost certainly differences, so the situation is very confusing.

A weirder argument with simulations

The last argument gave a kind of common-sense argument for being nice to some aliens. The rest of this post is going to be pretty crazy.

Let's consider a particular (implausible) strategy for building an AI:

- Start with a simulation of Earth.
- Keep waiting/restarting until evolution produces human-level intelligence, civilization, *etc.*
- Once the civilization is *slightly below* our stage of maturity, show them the real world and hand them the keys.
- (This only makes sense if the simulated civilization is much more powerful than us, and faces lower existential risk. That seems likely to me. For example, the resulting AIs would likely

think *much* faster than us, and have a much larger effective population; they would be very robust to ecological disaster, and would face a qualitatively easier version of the AI alignment problem.)

Suppose that *every* civilization followed this strategy. Then we'd simply be doing a kind of interstellar shuffle, where each civilization abandons their home and gets a new one inside of some alien simulation. It seems much better for everyone to shuffle than to accept a 10% chance of extinction.

Incentivizing cooperation

The obvious problem with this plan is that not everyone will follow it. So it's not really a shuffle: nice civilizations give up their planet, while mean civilizations keep their original planet *and* get a new one. So this strategy involves a net transfer of resources from nice people to mean people: some moral perspectives would be OK with that, but many would not.

This obvious problem has an obvious solution: since you are simulating the target civilization, you can run extensive tests to see if they seem nice—i.e. if they are the kind of civilization that is willing to give an alien simulation control rather than risk extinction—and only let them take over if they are.

This guarantees that the nice civilizations shuffle around between worlds, while the mean civilizations take their chances on their own, which seems great.

More caveats and details

This procedure might look really expensive—you need to simulate a whole civilization, nearly as large as your own civilization, with computers nearly as large as your computers. But in fact it doesn't require literally simulating the civilization up until the moment when they are building AI—you could use cheaper mechanisms to try to guess whether they were going to be nice a little bit in advance, e.g. by simulating large numbers of individuals or groups making particularly relevant decisions. If you were simulating humans, you could imagine predicting what the modern world would do without ever actually running a population of >100,000.

If only 10% of intelligent civilizations decide to accept this trade, then running the simulation is 10x as expensive (since you need to try 10 times). Other than that, I think that the calculation doesn't actually

depend very much on what fraction of civilizations take this kind of deal.

Another problem is that people may prefer continue existing in their own universe than in some weird alien simulation, so the “shuffle” may itself be a moral catastrophe that we should try to avoid. I’m pretty skeptical of this:

- You could always later perform an acausal trade to “go home,” i.e. to swap back with the aliens who took over your civilization (by simulating each other and passing control back to the original civilization if their simulated copy does likewise).
- In practice the universe is very big, and the part of our preferences that cares about “home” seems easily satiable. There is no real need for the new residents of our world to kill us, and I think that we’d be perfectly happy to get just one galaxy while the new residents get everything else. (Given that we are getting a whole universe worth of resources somewhere else.)

Another problem is that this is a hideously intractable way to make an AI. More on that two sections from now.

Another problem is that this is completely insane. I don’t really have any defense, if you aren’t tolerant of insanity you should probably just turn back now.

Decision theory

The above argument about trade / swapping places makes sense from a UDT perspective. But I think a similar argument should be persuasive even to a causal decision theorist.

Roughly speaking, you don’t have much reason to think that you are on the outside, considering whether to instantiate some aliens, rather than on the inside, being evaluated for kindness. If you are on the outside, instantiating aliens may be expensive. But if you are on the inside, trying to instantiate aliens lets you escape the simulation.

So the cost-benefit analysis for being nice is actually pretty attractive, and is likely to be a better deal than a 10% risk of extinction.

(Though this argument depends on how accurately the simulators are able to gauge our intentions, and whether it is possible to look nice but ultimately defect.)

How sensitive is moral value to the details of the aliens?

If an AI is from *exactly* the same distribution that we are, I think it's particularly likely that they are a good successor.

Intuitively, I feel like goodness probably doesn't depend on incredibly detailed facts about our civilization. For example, suppose that the planets in a simulation are 10% smaller, on average, than the planets in the real world. Does that decrease the moral value of life from that simulation? What if they are 10% larger?

What if we can't afford to wait until evolution produces intelligence by chance, so we choose some of the "randomness" to be particularly conducive to life? Does that make all the difference? What if we simulate a smaller population than evolution over a larger number of generations?

Overall I don't have very strong intuitions about these questions and the domain is confusing. But my weak intuition is that none of these things should make a big moral difference.

One caveat is that in order to assess whether a civilization is "nice," you need to see what they would do *under realistic conditions*, i.e. conditions from the same distribution that the "basement" civilizations are operating under. This doesn't necessarily mean that they need to evolve in a physically plausible way though, just that they *think* they evolved naturally. To test niceness we could evolve life, then put it down in a world like ours (with a plausible-looking evolutionary record, a plausible sky, etc.)

The decision-theoretic / simulation argument seems more sensitive to details than the commonsense morality argument. But even for the decision-theoretic argument, as long as we create a historical record convincing enough to fool the simulated people, the same basic analysis seems to apply. After all, how do we know that *our* history and sky aren't fake? Overall the decision-theoretic analysis gets really weird and complicated and I'm very unsure what the right answer is.

(Note that this argument is very fundamentally different from using decision theory to constrain the behavior of an AI—this is using decision theory to guide our *own* behavior.)

Conclusion

Even if we knew how to build an unaligned AI that is *probably* a good successor, I still think we should strongly prefer to build aligned AGI. The basic reason is option value: if we build an aligned AGI, we keep all of our options open, and can spend more time thinking before making any irreversible decision.

So why even think about this stuff?

If building aligned AI turns out to be difficult, I think that building an unaligned good successor is a plausible Plan B. The total amount of effort that has been invested in understanding which AIs make good successors is very small, even relative to the amount of effort that has been invested in understanding alignment. Moreover, it's a separate problem that may independently turn out to be much easier or harder.

I currently believe:

- There are definitely some AIs that aren't good successors. It's probably the case that many AIs aren't good successors (but are instead like PepsiCo)
- There are very likely to be some AIs that are good successors but are very hard to build (like the detailed simulation of a world-just-like-Earth)
- It's plausible that there are good successors that are easy to build.
- We'd likely have a *much* better understanding of this issue if we put some quality time into thinking about it. Such understanding has a really high expected value.

Overall, I think the question "which AIs are good successors?" is both neglected and time-sensitive, and is my best guess for the highest impact question in moral philosophy right now.

Towards formalizing universality



Paul Christiano

[Follow](#)

Jan 9 · 22 min read

The scalability of iterated amplification or debate seems to depend on whether large enough teams of humans can carry out arbitrarily complicated reasoning. Are these schemes “universal,” or are there kinds of reasoning that work but which humans fundamentally can’t understand?

This post defines the concept of “ascription universality,” which tries to capture the property that a question-answering system **A** is better-informed than any particular simpler computation **C**.

These parallel posts explain why I believe that the alignment of iterated amplification largely depends on whether HCH is ascription universal. Ultimately I think that the “right” definition will be closely tied to the use we want to make of it, and so we should be refining this definition in parallel with exploring its applications.

I’m using the awkward term “ascription universality” partly to explicitly flag that this is a preliminary definition, and partly to reserve linguistic space for the better definitions that I’m optimistic will follow.

(Thanks to Geoffrey Irving for discussions about many of the ideas in this post.)

I. Definition

We will try to define what it means for a question-answering system **A** to be “ascription universal.”

1. Ascribing beliefs to A

Fix a language (e.g. English with arbitrarily big compound terms) in which we can represent questions and answers.

To ascribe beliefs to **A**, we ask it. If **A** (“are there infinitely many twin primes?”) = “probably, though it’s hard to be sure” then we ascribe that belief about twin primes to **A**.

This is not a general way of ascribing “belief.” This procedure wouldn’t capture the beliefs of a native Spanish speaker, or for someone who wasn’t answering questions honestly. But it can give us a sufficient condition, and is particularly useful for someone who wants to use **A** as part of an alignment scheme.

Even in this “straightforward” procedure there is a lot of subtlety. In some cases there are questions that we can’t articulate in our language, but which (when combined with **A**’s other beliefs) have consequences that we can articulate. In this case, we can infer something about **A**’s beliefs from its answers to the questions that we can articulate.

2. Ascribing beliefs to arbitrary computations

We are interested in whether **A** “can understand everything that could be understood by someone.” To clarify this, we need to be more precise about what we mean by “could be understood by someone.”

This will be the most informal step in this post. (Not that any of it is very formal!)

We can imagine various ways of ascribing beliefs to an arbitrary computation **C**. For example:

- We can give **C** questions in a particular encoding and assume its answers reflect its beliefs. We can either use those answers directly to infer **C**’s beliefs (as in the last section), or we can ask what set of beliefs about latent facts would explain **C**’s answers.
- We can view **C** as optimizing something and ask what set of beliefs rationalize that optimization. For example, we can give **C** a chess board as input, see what move it produces, assume it is trying to win, and infer what it must believe. We might conclude that **C** believes a particular line of play will be won by black, or that **C** believes general heuristics like “a pawn is worth 3 tempi,” or so on.
- We can reason about how **C**’s behavior depends on facts about the world, and ask what state of the world is determined by its current behavior. For example, we can observe that **C**(113327) = 1 but that **C**(113327) “would have been” 0 if 113327 had been composite, concluding that **C**(11327) “knows” that 113327 is prime. We can extend to probabilistic beliefs, e.g. if **C**(113327) “probably” would have been 0 if 113327 had been composite,

then we might that **C** knows that 113327 is “probably prime.” This certainly isn’t a precise definition, since it involves considering logical counterfactuals, and I’m not clear whether it can be made precise. (See also ideas along the lines of “knowledge is freedom”.)

- If a computation behaves differently under different conditions, then we could use restrict attention to a particular condition. For example, if a question-answering system appears to be bilingual but answers questions differently in Spanish and English, we could ascribe two different sets of beliefs. Similarly, we could ascribe beliefs to any subcomputation. For example, if a part of **C** can be understood as optimizing the way data is laid out in memory, then we can ascribe beliefs to that computation about the way that data will be used.

Note that these aren’t intended to be efficient procedures that we could actually apply to a given computation **C**. They are hypothetical procedures that we will use to define what it means for **A** to be universal.

I’m not going to try to ascribe a single set of beliefs to a given computation; instead, I’ll consider all of the reasonable ascription procedures. For example, I think different procedures would ascribe different beliefs to a particular human, and don’t want to claim there is a unique answer to what a human “really” believes. A universal reasoner needs to have more reasonable beliefs than the beliefs ascribed to that a human using any particular method.

An ascription-universal reasoner needs to compete with any beliefs that can be ascribed to **C**, so I want to be generous with this definition. For example, given a chess-playing algorithm, we might rationalize it as trying to win a game and infer its beliefs about the rules of chess. Or we might rationalize it as trying to look like a human and infer its beliefs about what a human would do. Or something different altogether. Most of these will be kind of crazy ascriptions, but I want to compete with them anyway (competing with crazier beliefs will turn out to just be easier).

It’s not totally clear what counts as a “reasonable” ascription procedure, and that’s the biggest source of informality. Intuitively, the key property is that the ascription itself isn’t doing the “hard work.” In practice I’m using an informal extensional definition, guided by examples like those in the bulleted list.

3. Comparing beliefs

What does it mean to say that one agent is “better-informed” than another?

It’s natural to try to express this in terms of empirical information about the world, but we are particularly interested in the different inferences that agents are able to draw from the same data. Another natural approach is to compare their “knowledge,” but I have no idea how to define knowledge or justified belief. So I’m reduced to working directly with sets of beliefs.

Consider two sets of beliefs, described by the subjective expectations \mathbb{E}^1 and \mathbb{E}^2 . What does it mean to say that \mathbb{E}^1 is better-informed than \mathbb{E}^2 ?

This framing makes it tempting to try something simple: “for every quantity, \mathbb{E}^1 ’s belief about that quantity is more accurate.” But this is property is totally unachievable. Even if \mathbb{E}^1 is obtained by conditioning \mathbb{E}^2 on a true fact, it will almost certainly happen to update in the “wrong” direction for some claims.

We will instead use a subjective definition, i.e. we’ll define this concept from a particular epistemic position represented by another subjective expectation \mathbb{E} .

Then we say that \mathbb{E}^1 **dominates** \mathbb{E}^2 (w.r.t. \mathbb{E}) if, for every bounded quantity X and for every “nice” property Φ :

- $\mathbb{E}[X|\Phi(\mathbb{E}^1, \mathbb{E}^2)] = \mathbb{E}[\mathbb{E}^1[X]|\Phi(\mathbb{E}^1, \mathbb{E}^2)]$

(By “nice” I mean something like: simple to define and open in the product topology, viewing \mathbb{E}^1 and \mathbb{E}^2 as infinite tables of numbers.)

Intuitively, this means that \mathbb{E} always “trusts” \mathbb{E}^1 , even if given arbitrary information about \mathbb{E}^1 and \mathbb{E}^2 . For example, if \mathbb{E} was told that $\mathbb{E}^1[X] \approx x$ and $\mathbb{E}^2[X] \approx y$, then it would expect X to be around x (rather than y). Allowing arbitrary predicates Φ allows us to make stronger inferences, effectively that \mathbb{E} thinks that \mathbb{E}^1 captures *everything* useful about \mathbb{E}^2 .

I’m not sure if this is exactly the right property, and it becomes particularly tricky if the quantity X is itself related to the behavior of \mathbb{E}^1 or \mathbb{E}^2 (continuity in the product topology is the minimum

plausible condition to avoid a self-referential paradox). But I think it's at least roughly what we want and it may be exactly what we want.

Note that dominance is *subjective*, i.e. it depends on the epistemic vantage point \mathbb{E} used for the outer expectation. This property is a little bit stronger than what we originally asked for, since it also requires \mathbb{E} to trust \mathbb{E}^1 , but this turns out to be implied anyway by our definition of universality so it's not a big defect.

Note that dominance is a property of the *descriptions* of \mathbb{E}^1 and \mathbb{E}^2 . There could be two different computations that in fact compute the same set of expectations, such that \mathbb{E} trusts one of them but not the other. Perhaps one computation hard-codes a particular result, while the other does a bunch of work to estimate it. Even if the hard-coded result happened to be correct, such that the two computations had the same outputs, \mathbb{E} might trust the hard work but not the wild guess.

4. Complexity and parameterization

There are computations with arbitrarily sophisticated beliefs, so no fixed \mathbf{A} can hope to dominate everything. To remedy this, rather than comparing to a fixed question-answerer \mathbf{A} , we'll compare to a parameterized family $\mathbf{A}[\mathbf{C}]$.

I'll consider two different kinds of potentially-universal reasoners \mathbf{A} :

- In the “idealized” case, $\mathbf{A}[\mathbf{C}]$ depends only on the complexity of \mathbf{C} .
For example, we might hope that an n -round debate dominates any beliefs that could be ascribed to a fast computation with $(n-1)$ rounds of alternation. In particular, this $\mathbf{A}[\mathbf{C}]$ is the same for any two computations \mathbf{C} of the same complexity.
- In the “practical” case, $\mathbf{A}[\mathbf{C}]$ depends on the complexity of \mathbf{C} but also uses the computation \mathbf{C} as a hint. For example, if \mathbf{C} is the training process for a neural net, then we might take $\mathbf{A}[\mathbf{C}]$ to be a debate in which the debaters are able to share weights and activations with the neural net throughout the entire training process.

I'm generally interested in the case where $\mathbf{A}[\mathbf{C}]$ is only slightly more powerful than \mathbf{C} itself. This mirrors the setting where a universal Turing machine is able to run any other Turing machine with only a modest slowdown.

Putting it all together

We say that a set of beliefs \mathbb{E}^A *epistemically dominates* a computation **C** (w.r.t. some beliefs \mathbb{E} and language L) if the beliefs ascribed to A by the “straightforward” procedure, using L, dominate (w.r.t. \mathbb{E}) the beliefs ascribed to **C** by any reasonable ascription procedure.

We say that a family of question-answering systems $\mathbf{A}[\cdot]$ are *ascription universal* (w.r.t. \mathbb{E} and L) if $\mathbf{A}[\mathbf{C}]$ epistemically dominates **C** for every computation **C**.

II. Discussion

Why is (subjective) dominance sufficient?

This universality condition requires that *we* believe that $\mathbf{A}[\mathbf{C}]$ is better-informed than **C**. Naively we might have wanted it to *actually* be the case that $\mathbf{A}[\mathbf{C}]$ is better-informed than **C**; the stronger condition is clearly unachievable, but why should we be satisfied with the weaker condition?

In applications of this property, the subjective condition is what we need in order for *us* to believe that $\mathbf{A}[\mathbf{C}]$ will cope with the challenges posed by **C**. For example, suppose that **C** formulates a plan to “trick” $\mathbf{A}[\mathbf{C}]$. Then the subjective universality condition implies that we don’t expect **C** to succeed.

This isn’t as good as actually knowing that **C** won’t succeed. But I think it should be good enough for us—the reason we are thinking about AI safety is because we are concerned that something bad will happen. If we find a technique that defuses this argument, then we’ve addressed the motivating problem. It may still be the case that bad things happen (and we should still search for additional reasons that bad things might happen), but we don’t particularly expect them to.

Of course if you select over a large number of computations, then you may find one that will succeed in tricking **A**. But if we are concerned about that, then we can instead apply ascription universality to the entire process including the selection.

Why trust opaque computation?

If **C** uses some clever heuristics that I don’t understand, then **C**’s “beliefs” might be excellent, but I might not *expect* them to be excellent. In this sense understanding may seem almost vacuous. If there is some heuristic that I trust, wouldn’t **A** just use it?

To see why the definition is demanding, consider the special case where **C** performs an extensive search to find a computation that works well empirically. For example, **C** might be the following computation:

- Start with a training set of (image, label) pairs.
- Search over simple programs to find one that makes good predictions.
- Run that simple program on a new image to predict its label.

In this case, we can ascribe beliefs to **C** about the contents of the new image. And because those beliefs are coming from a simple program that works empirically, I expect them to be accurate (in some respects).

For example, a simple classifier **C** may “believe” that the new image contains a particular curve that typically appears in images labeled “dog;” or a really sophisticated classifier may perform complex deductions about the contents of the scene, starting from premises that were empirically validated on the training set.

So it’s not OK for **A** to simply ignore whatever heuristics **C** is using—if those heuristics have the kind of empirical support that makes us think they actually work, then **A** needs to be able to understand everything that those heuristics imply about the domain.

Why be so general?

I’ve formulated universality as competing with arbitrary computations **C**. It seems totally possible that the form of **C** discussed in the last section—searching for a program that works well in practice and then using it in a new situation—is so central that the definition of universality should focus entirely on it.

One reason to use the broader definition is because sometimes this “selection” process can be embedded in a non-trivial way in a larger computation. For example, if I have a sufficiently large group of humans, I might expect memetic selection to occur and produce systems that could be said to have “beliefs,” and I’d like universal systems to dominate those beliefs as well.

The other reason to use this very general definition is because I don’t see an easy way to simplify the definition by using the additional

structural assumption about C. I do think it's likely there's a nicer statement out there that someone else can find.

Universal from whose perspective?

Unfortunately, achieving universality depends a lot on the epistemic perspective \mathbb{E} from which it is being evaluated. For example, if \mathbb{E} knows any facts, than a universal agent must know all of those facts as well. Thus “a debate judged by Paul” may be universal from Paul’s perspective, but “a debate arbitrated by Alice” cannot be universal from my perspective unless I believe that Alice knows everything I know.

This isn’t necessarily a big problem. It will limit us to conclusions like: Google engineers believe that the AI they’ve built serves the user’s interests reasonably well. The user might not agree with that assessment, if they have different beliefs from Google engineers. This is what you’d expect in any case where Google engineers build a product, however good their intentions.

(Of course Google engineers’ notion of “serving the user’s interests” can involve deferring to the user’s beliefs in cases where they disagree with Google engineers, just as they could defer to the user’s beliefs with other products. That gives us reason to be less concerned about such divergences, but eventually these evaluations do need to bottom out somewhere.)

This property becomes more problematic when we ask questions like: is there a way to seriously limit the inputs and outputs to a human while preserving universality of HCH? This causes trouble because even if limiting the human intuitively preserves universality, it will effectively eliminate some of the human’s knowledge and know-how that can only be accessed on large inputs, and hence violate universality.

So when investigating schemes based on this kind of impoverished human, we would need to evaluate universality from some impoverished epistemic perspective. We’d like to say that the impoverished perspective is still “good enough” for us to feel safe, despite not being good enough to capture literally everything we know. But now we risk begging the question: how do we evaluate whether the impoverished perspective is good enough? I think this is probably OK, but it’s definitely subtle.

I think that defining universality w.r.t. \mathbb{E} is an artifact of this definition strategy, and I'm optimistic that a better definition wouldn't have this dependence, probably by directly attacking the notion of "justified" belief (which would likely also be useful for actually establishing universality, and may even be necessary). But that's a hard problem. Philosophers have thought about very similar problems extensively without making the kind of progress that seems adequate for our purposes, and I don't see an immediate angle of attack.

III. Which A might be universal?

Two regimes

I'm interested in universality in two distinct regimes:

- Universality of idealized procedures defined in terms of perfect optimization, such as debate under optimal play or max-HCH, where $A[C]$ depends only the computational complexity of C .
- Universality of practical variants, in which the perfect optimization is replaced by a bounded optimization, which might use the computation C as a "hint." For example, we might consider amplification or debate where the agents are allowed to inspect a transcript of C ; or if C is a neural net training process, we might train them jointly with C .

Idealized models

$A[C]$ could be:

- A debate, judged by a human, with perfect debaters, whose length depends on the complexity of C . (Hopefully in such a way that $A[A[C]]$ is only a slightly longer debate than $A[C]$.)
- max-HCH, using perfect optimization rather than a bounded optimizer. Again, we can impose a budget limit that depends on the complexity of C , ideally such that $A[A[C]]$ does not involve a much larger computation than $A[C]$.
- Use HCH to implement two functions, Info and Answer. Define $A[C](Q)$ to be $\text{Answer}(Q, a^*)$, where $a^* = \text{argmax } \text{Info}(a, Q)$, and the optimization is over advice strings a of length comparable to the description of C .

I don't think any of these would be universal if instantiated with an open-ended human policy—providing arbitrary complex inputs to a human is likely to lead to bad behavior. Instead, I imagine restricting the human to some small enough space of inputs that they can behave “reasonably” over the whole space, while still being rich enough to be universal. Even as such, I think additional measures might be needed to achieve security (since a team of humans may behave badly in some exotic corners of the input space, even if individual humans are sheltered from exotic inputs).

Practical models

In practice I'm particularly interested in the case where **C** has the special form described in the “Trusting opaque computations” section: it trains a neural net on some dataset, and then runs that neural network on new inputs.

A[C] could be:

- Debates arbitrated by a human where the debaters are trained jointly with **C**, and can make references to individual examples from the dataset.
- max-HCH where the agent is trained jointly with **C**, and the overseer can consult individual entries from the dataset.
- HCH, combined with interpretability techniques that allow it to get information about what **C** is thinking and to relate its behavior to the dataset on which it was trained.

For each of these practical variants, I don't expect it to be universal unless the corresponding idealized model is. So we can think of dividing the universality question into two pieces:

- Would our procedure be universal *if the optimization was performed perfectly?*
- Is weight sharing / interpretability strong enough to solve the optimization problems needed for universality?

I think none of the suggestions in this section are plausibly universal without some additional mechanisms, and so even if HCH or debate is universal, additional ideas are needed in order to translate those to practical universal systems. I'm still inclined to focus on the idealized case, because I want to better understand what is needed for universality without thinking about algorithmic difficulties.

IV. Which C are hard to epistemically dominate?

The definition in this post was somewhat informal, but the discussion in this appendix will be extremely informal. The purpose is to communicate some of my early thinking about universality and my sense for what a universal reasoner might look like.

The first three examples will consider programs **C** with the following form:

- Iterate over programs to find an f that successfully predicts the answers to some dataset of historical questions for which we know the answer.
- Use that program to predict the answers to future questions.

The different examples explore features of f that might make it difficult to dominate the overall computation.

Deduction

Suppose f takes the following form:

- Given a question and context, it performs a short series of deductions from facts in the context, together with some axioms (e.g. “Bob’s eyes are more likely to twitch when he is lying,” the rules of poker, etc.)
- When it arrives at answers that constrain the possible answers to the given question, it outputs an answer consistent with those constraints.

We could attribute the statements produced by this process to **C** as beliefs. We expect this set of beliefs to have useful information because it was selected for making correct predictions.

Fortunately, if HCH is able to see the training dataset then it may be able to arrive at similarly accurate views:

- It can compute that $P(\text{observations} \mid \text{axioms})$ is high by following along with the deductive process used by f .
- Bayes implies that the axioms are likely to be true (fully capturing the argument which leads us, from our current perspective, to expect them to be true).

- Once HCH knows the axioms are likely to be true, it can follow along with the deductive process to reproduce all of f 's beliefs.

Modeling

Suppose f takes the following form:

- Given a question Q and context, set up a relevant physical situation.
- Use a dynamics model M to predict what will happen at some future time.
- Use the resulting state to answer Q .

We could say that \mathbf{C} believes everything implied by the outcome of this physical simulation. We expect these beliefs to be accurate because f has predicted well in the past.

As in the deduction case, hopefully HCH can compute that $P(\text{observations} | M \text{ is accurate})$ is high, since it can also compute the consequences of M . Then Bayes implies the dynamics model is accurate, and HCH can use that model to compute physical states.

Inferring all the beliefs from a dynamics model is not trivial though. As an extreme example, if f is performing an atom-by-atom simulation of a room, and that room contains Alice and Bob, then we could ascribe extensive beliefs about Alice and Bob to the computation \mathbf{C} .

(Here we run head on into the fuzziness about what counts as a “reasonable” ascription procedure, but for the moment I’ll assume that some reasonable procedure ascribes beliefs about Alice and Bob to the computation.)

To compete with these ascriptions, HCH needs to infer those high-level beliefs about Alice and Bob from the low-level computation involving atoms. One way to do this is to search over possible “bridging” hypotheses that relate low-level physical facts to high-level facts about the environment. If such a hypothesis can explain additional high-level facts, then a Bayesian can learn that it is true. Similarly, if the bridging hypothesis relates facts about the model to constraints we know from the high-level interpretation, then the Bayesian can potentially use that as evidence. (This kind of reasoning will be discussed in a bit more detail in the next section.)

We could further hope that searching for a bridging hypothesis isn't much harder than performing the original search over low-level physics, given that the low-level physics needed to explain a bunch of high-level facts and so already must encode some part of that correspondence.

(Note that the “deduction” example in the previous case could also involve alien concepts or models, in which case the same kind of work would be needed.)

Alien reasoning

In the previous section we described two styles of reasoning we already understand. But there are probably many kinds of reasoning that work well in practice but that would be more alien, and those might be more challenging. This section will explore one example in some detail to try to help anchor our reasoning about the general phenomenon. It will also elaborate on some of the reasoning about “bridging” hypotheses mentioned in the last section.

Suppose that our predictions are always of the same form (e.g. what is the probability the stock market will go up today), and f works as follows (the details are long but not very important):

- Find the PSD matrix A with maximum log determinant subject to the constraints in the next bullet points, then output the $(0, 0)$ entry.
- There is an implicit correspondence between the rows/columns of A , and some uncertain properties $X(0), X(1), X(2), \dots$ (which we'll view as 0–1 variables), where $X(0)$ is the property we want to forecast.
- If the (i, j) entry of A represented the expectation $E[X(i)X(j)]$, then the matrix would necessarily satisfy a bunch of constraints, which we impose A . For example:
 - If the context implies that $X(i) = 1$, then $E[X(i)X(j)] = E[X(j)] = E[X(j)^2]$, so $A(i, j) = A(j, j)$.
 - If $X(i)$ and $X(j)$ together imply $X(k)$, then we must have $E[X(i)X(j)] \leq E[X(i)X(k)]$ and hence $A(i, j) \leq A(i, k)$.
 - For any constants a, b, \dots , $E[(a X(1) + b X(2) + \dots)^2] \geq 0$ —i.e., the matrix A must be PSD.

The chosen matrix $A(\text{opt})$ corresponds to a set of beliefs about the propositions $X(i)$, and we can ascribe these beliefs to \mathbf{C} . Because f predicts well, we again expect these beliefs to say something important about the world.

I chose this procedure f in part because we can give a kind of argument for why the matrix $A(\text{opt})$ should tend to encode accurate beliefs. But I don't think that a universal reasoner can make use of that argument:

- Finding the argument that f works is an additional problem, beyond finding f itself, which might be much harder.
- A comprehensible version of that argument may be much larger than the strategy itself, so even in the idealized cases like debate with perfect optimization, we may need to increase the scale.
- I don't expect that all "good" reasoning strategies have clean understandable arguments in their favor (and even in this case, if it the scheme worked well it would be largely an empirical fact rather than a consequence of the simple theorems we could prove). I think this kind of example is useful because we can easily imagine a human debate judge not having the argument while still being apparently universal. This makes it a useful analogy for cases where the argument really doesn't exist.

Instead, I think a universal reasoner needs to be able to infer the efficacy of this reasoning procedure from its empirical success. It's relatively easy for a Bayesian to learn the regularity " f makes good predictions." Recovering the rest of the matrix A , and learning how to interpret and whether to trust them, is the hard part.

This is going to require the same kind of bridging/identification we discussed in the last section. Let's write $X(A)$ for the set of beliefs about the world implied by the "intended" identification. Searching over possible identifications to find X (or something like it) is the only way we can ever relate the rows of A to the quantities $X(i)$. Again, we can hope that it isn't much harder than finding the original reasoning procedure.

I think that a sufficiently sophisticated Bayesian would probably be able to learn to trust $X(A)$:

- If f is performing well enough that we think it's more likely to be right in the future, then the Bayesian is going to end believing

some claim like “the predictions of f are good” (since it explains the data so well).

- This is a complicated statement, and without some kind of explanation this claim has a low prior probability (roughly decaying with the complexity of f). The Bayesian is motivated to find an explanation with higher prior probability.
- The correspondence X can explain the constraints on the matrix A , in terms of facts that we already know about the world. This explanation may end up being simpler (or at least higher prior) than a direct enumeration of the constraints on A —I hope (and think it’s plausible) that this happens iff we’d actually believe on reflection that $X(A)$ captures reality.
(To the extent that we are uncertain and think A ’s beliefs have a non-negligible chance of capturing reality, then hopefully we can capture that by the same mechanism by ending up with a non-degenerate posterior.)
- Now the Bayesian is faced with at least two kinds of explanations:
 - (a) “If you use the constraints implied by correspondence $X(A)$ + positive semidefiniteness, and then optimize log det, you get a matrix A for which $X(A)$ makes good predictions,”
 - (b) “The actual situation in the real world is described by positive semi-definite matrices with higher log determinant (under the correspondence X).”
- Explanation (b) is explaining two things at once: both why the optimization done by f respects the constraints on our beliefs, and why that optimization leads to good predictions. Hopefully this is simpler than making two separate bridging claims, one which explains f as respecting the constraints implied by X , and one which claims that f makes good predictions. Ideally, this 2-for-1 that favors (b) exactly mirrors the underlying reasoning that leads us to actually believe that $X(A)$ is correct, rather than resembling what we know about reality and making good predictions “by coincidence.”

This is a pretty speculative discussion—it’s not very careful, and it’s hard to make it careful in part because I don’t have a formalization of Bayesian reasoning that can even really be applied to this setting. But it seems to match my intuitions about what reasonable Bayesian reasoning “should” do, which gives me a lot more optimism that a careful Bayesian would be able to epistemically dominate **C**.

Deliberation and self-improvement

Often we expect the computation **C** to have accurate beliefs because it uses a strategy that appears to work in practice—the last 3 examples have discussed that case. But there are other reasons to trust a computation.

For example, humans often write code and trust it (to some extent) even without extensive empirical testing—instead, we have a reason to think it will work, and need only modest testing to make sure that we haven’t made an error in our implementation or reasoning. If I write an automated mathematician that works by finding proofs that pass a proof checker, I don’t expect it to be correct because of the empirical record (Empirical data backs up some key assumptions, but isn’t being used to directly establishing the correctness of the method.)

Likewise, if we train a powerful agent, that agent might initially use strategies that work well in training, but over time it might use learned reasoning to identify other promising strategies and use those. Reasoning might allow it to totally skip empirical testing, or to adopt the method after much less testing than would have been necessary without the reasoning.

To dominate the beliefs produced by such reasoning, we can’t directly appeal to the kind of statistical inference made in the previous section. But in these cases I think we have access to an even more direct strategy.

Concretely, consider the situation where **C** contains a process f that designs a new reasoning process g . Then:

- From the outside, we trust g because we trust f and it trusts g .
- An otherwise-universal reasoner **A** will dominate f ’s beliefs, and in particular iff f is justified in thinking that g will work then **A** will believe that and understand why.
- Once we understand f ’s beliefs, dominating g is essentially another instance of the original ascription universality problem, but now from a slightly stronger epistemic state that involves both what \mathbb{E} knows and what f knows. So unless our original approach to universality was tightly wedded to details of \mathbb{E} , we can probably dominate g .

At the end of the day we'd like to put all of this together into a tight argument for universality, which will need to incorporate both statistical arguments and this kind of dynamic. But I'm tentatively optimistic about achieving universality in light of the prospect of agents designing new agents, and am much more worried about the kind of opaque computations that "just work" described in the last few sections.

Informed oversight (revisited)



Paul Christiano

[Follow](#)

Jan 9 · 8 min read

To train an agent with RL, we need to answer the question: “which of the trajectories τ^1 or τ^2 is better?”

Likewise, in order to train for acceptable worst-case performance, we probably need to answer a question like: “is trajectory τ acceptable?”

We want to answer these questions “well enough” that our agents end up aiming for good outcomes on average and always behaving acceptably. But how well is well enough?

In this post I’ll argue that if our goal is to train a well-motivated agent, it is necessary and sufficient for our answers to reflect *everything our agent knows*. This seems essentially equivalent to ascription universality of the oversight process.

(This post is an update of these 2016 posts. My understanding of the problem has improved a lot over the last few years.)

Information vs incentives

We can distinguish two functions of feedback:

- It provides *information* to the agent we are training. For example, if I provide negative feedback when an agent serves me pizza, it can learn that I don’t like pizza. (This might happen by gradient descent selecting for agents with different beliefs, or by an RL algorithm that uses past reward signals as information.)
- It determines the *incentives* for the agent we are training, i.e. is used to select the motivations of the agent. For example, an agent pre-trained to make predictions about the world might already understand that I dislike pizza. If I provide negative feedback when the agent serves me pizza, then gradient descent will select for agents that are motivated to avoid things I dislike (at least instrumentally).

From the perspective of information, there is no such thing as “good enough.” More information is always better.

This post is about incentives. We can aim for a “correctly motivated” agent, who lacks relevant information but is at least trying to do the right thing. At that point there is nothing to gain from better oversight.

So our question is: how well do we have to answer these questions to produce an approximately correctly motivated agent?

(In the long run, I think that incentives are the more important function of feedback. Agents should mostly learn about the world from other richer signals. So I’m not worried about whether our feedback is optimal from an information perspective.)

Necessity of knowing what the agent knows

Suppose that our agent understands facts about the world that we don’t understand, i.e which aren’t reflected in our answer to “which of the trajectories τ^1 or τ^2 is better?” I’ll argue that this is a bad situation.

If the agent literally took actions and made observations that weren’t available to us, then we would clearly have a problem. For example, consider an agent trying to defend a computer system from attackers. If the agent has access to actions or information we are missing, then it might deliberately take actions that compromise the tools we would use to determine whether our system has been compromised. Because we can’t observe this malicious behavior, we can’t assign it a low reward. And after compromising our tools well enough, the agent could ensure that they always paint a rosy picture of the situation regardless of what actually happens. We intended to protect ourselves from attack, but instead introduced a powerful attacker with privileged access to our system.

The same thing can happen if we observe everything our agent observes, if we aren’t able to understand everything our agent understands. In the security example, literally seeing a sequence of bits moving across an interface gives you almost no information—something can look innocuous, but cause a huge amount of trouble. In order to incentivize our agent to avoid causing trouble, we need to be able to detect any trouble that the agent deliberately causes. Even an apparently mundane gap in our understanding could hide attacks,

just as effectively as if we'd been literally unable to observe the agent's behavior.

We might hope that an attack will ultimately cash out as *something* that we can unambiguously observe, so that this is merely a problem with short time horizons. But I don't think that's the case—ultimately we are always restricted to indirect observations of the things we care about. We rely on second-hand reports, on viewing small parts of a situation that we hope are representative, on cameras and other sensors that purport to describe the world. All of these things would also be manipulated by an agent incentivized to make things look good to us, and understanding what the agent understands is the most natural defense against that kind of manipulation.

Intuitive argument of sufficiency

Suppose we "know everything that the agent knows." That means the agent should defer to our estimate for any unknown quantity.

Now consider two possible motives an agent could have:

- Create trajectories that are actually good, according to our "real" preferences (extrapolated as we wish that extrapolated...)
- Create trajectories we believe are good

If the agent defers to us about the goodness of trajectories, then these two motives will lead to identical actions. So selecting for agents that take actions that we think are good also optimally selects for agents actually motivated to do good.

Intuitively, imagine a student who is trying to predict whether their teacher will think a statement is true. If the teacher knows everything the student knows and can see every consideration the student sees, then the student shouldn't expect to be able to predict any systematic error in the teacher's beliefs. So trying to predict what the teacher believes is exactly the same as trying to predict what's true.

Tightening the argument

The argument in the last section isn't quite right. Most glaringly, if the overseer is universal then it means *we* think the overseer has better beliefs than the agent, but it doesn't mean *the agent* thinks that.

Changing the statement. To fix the argument, we need to be a bit more careful. We'll aim to show that if we think the overseer is ascription universal, then:

- We expect the agent A to achieve at least as much utility (by our real, unobserved utility function) than any action that can be easily computed using any set of beliefs that can be ascribed to A —and in particular A achieves at least as much utility as if it were perfectly motivated.

We can't infer that A is perfectly motivated, because it may have arbitrary beliefs that differ from those of the overseer. For example, A might believe that the world will certainly end in a week; instead of planning under that assumption (as it would if it were “correctly” motivated), it will learn to choose actions that the overseer prefers. From its perspective it is choosing actions suboptimally in order to please the overseer, but from our perspective this is better than if it were “correctly” motivated.

Another way to see we need this weakening: many different beliefs can be ascribed to the agent, and it can't simultaneously behave as if it were perfectly motivated with respect to each of those sets of beliefs.

Assumptions and proof. This will mostly just be notation; other than that it follows almost immediately from the definition of ascription universality, though we need to do a little bit of work to massage the definition into the right form.

Let \mathbb{E} be any subjective expectation w.r.t. which the overseer is ascription universal.

Let \mathbb{E}^o be the expectations of the overseer. Let U be our actual utility —this is not a well-defined function we can actually evaluate, but we can query a subjective expectation about it. Then the reward function is $r(a) = \mathbb{E}^o[U(a)]$. Note that the overseer doesn't condition on the fact that the agent produced this particular action (so no one needs to reason about the training process itself).

Let \mathbb{E}^A be some expectations that can be ascribed to the agent. Let A be the expression representing the agent's output (**after training**), and let a be the actual output (we need to distinguish the symbol from the action itself because we are plugging this into a subjective expectation). Let A' be a particular different definition of an action

the agent could feasibly output, e.g. $\text{argmax } \mathbb{E}^A[\mathbb{E}^o[U(\cdot)]]$, and let a' be its actual value.

Because A and A' can both be computed using the beliefs \mathbb{E}^A , we can apply the epistemic dominance property to predicates defined in terms of A, A' . (There is the subtlety that dominance requires properties to be open in the product topology, which can be avoided by carefully considering randomization and only becomes relevant when the utility function U itself depends on the behavior of the agent.)

We'll assume that \mathbb{E} believes that training converges well enough that the action A will receive a higher expected reward than A' (otherwise the agent would eventually switch to outputting A'). We also assume that \mathbb{E} knows that the overseer epistemically dominates the agent, and that it satisfies a standard iterated expectation property.

Then we have:

- $\mathbb{E}[a \leftarrow A; a' \leftarrow A'; U(a) - U(a')]$
 $= \mathbb{E}[a \leftarrow A; a' \leftarrow A'; \mathbb{E}[U(a) - U(a') | a=A, a'=A']]$ (*iterated expectation*)
 $= \mathbb{E}[a \leftarrow A; a' \leftarrow A'; \mathbb{E}[\mathbb{E}^o[U(a) - U(a')] | a=A, a'=A']]$ (*dominance*)
 $= \mathbb{E}[a \leftarrow A; a' \leftarrow A'; \mathbb{E}^o[U(a) - U(a')]]$ (*iterated expectation*)
 $= \mathbb{E}[a \leftarrow A; a' \leftarrow A'; r(a) - r(a')]]$ (*definition of reward*)
 ≥ 0 (*training is converged*)

As desired.

(The meaning of the $\mathbb{E}[\dots; \dots]$ notation is to take the expected value of the output of the program which performs the sequence of steps separated by ;'s, returning the final value.)

This argument only constrains the equilibrium, but eventually we need to say something about the entire training process. It looks to me like very similar ideas are needed for the more general case, but it's a lot harder to make the arguments precise.

Corrigibility

So far we've discussed reward functions that induce correct motivations. The other half of our problem was recognizing unacceptably bad behavior: mistakes so bad that they cannot be corrected.

Again, we can't hope to recognize all such mistakes. For example, it may turn out that a particular physics experiment will destroy the world, but correctly predicting that would require a detailed understanding of physics.

But if we know everything the agent knows, then we can avoid cases where the agent intentionally brings about a catastrophic outcome. In particular, we can avoid cases where the agent takes an action precisely because it would prevent us from correcting the agent's behavior or reasserting control over the situation.

Ascription universality seems to be just what we need for this purpose as well: if we believe the agent is pursuing a plan *because* it prevents us from correcting the situation, then we can ascribe that belief to the agent, and so an ascription-universal overseer will recognize that the behavior is incorrigible and can deem the behavior as unacceptable.

Such an overseer may also sometimes flag as unacceptable certain actions that are catastrophic but which the agent didn't recognize as catastrophic. But I think that's fine, I'm not unhappy if we end up training the agent to avoid a catastrophic mistake.

What does this actually look like?

If we are training a model with RL, my best candidate for obtaining a universal overseer is using iterated amplification to train a second head on the agent to answer relevant questions, including questions of the form "Which of these two trajectories is better?"

There are difficulties from training a model to play a game where it shares almost all of its activations with its opponent, and with ensuring that the overseer can remain sufficiently competent relative to the agent (without slowing down training). These resemble the discussion of the `Info` proposal here, but rather than having a separate overseer who evaluates how useful information was, we directly optimize the side information by using the same amplification process. Both of these issues seem OK with gradient descent, but they are larger problems for some other search algorithms, so I don't consider the issue settled.

This proposal also faces all of the usual difficulties of iterated amplification—it's not clear how to generate the distribution of questions, whether the training process is stable, whether errors compound, whether the steps of amplification increase the

complexity of training, and so on. These issues seem practically important, and keeping informed oversight in mind can help us understand exactly where the bar is for an adequate solution. I tentatively think the conceptual difficulties in achieving ascription universality are more likely to be a fundamental obstruction.

Universality and consequentialism within HCH



Paul Christiano

[Follow](#)

Jan 9 · 9 min read

The scalability of iterated amplification is closely tied to whether very large teams of humans are “ascription universal.” If they are, we could hope that their approval would be an adequate training signal for powerful aligned behavior, and that we can approximate this signal with iterated amplification.

There are many reasons that a team of humans could fail to be universal. For example:

1. They could be unable to understand some kinds of effective reasoning, because they can't represent relevant concepts, understand particular inferences, or whatever else.
2. They could make errors that derail an otherwise-universal reasoning process.
3. They could use processes that work “most of the time,” but fail catastrophically for a small number of inputs.
4. Errors of type #2 or #3 might produce extreme outputs that tend to propagate through the reasoning process and compound into larger failures (even if they occur with small probability or for a small fraction of inputs).
5. In a large enough computation, errors of type #2 or #3 could eventually give rise to intelligent patterns of behavior that systematically identify and exploit mechanisms by which they could propagate themselves or cause cascading failures. For intuition, note that memetic selection amongst a group of 10^{100} humans could be much stronger than the genetic selection that gave rise to life on Earth.

(This is not an exhaustive, exclusive, or even properly parallel list.)

My current view is that #2 and #3 can probably be corrected by traditional methods, even when compounded by #4: computing carefully or redundantly, sanity-checking outputs, structuring

computations to be more robust to isolated failures, and so on. These problems seem easier to solve for smarter teams (though also more problematic if left unchecked), and I suspect there is some competence threshold above which they become manageable.

A priori, #2 or #3 compounded by #5 could be harder to cope with:

- An intelligent optimization process could systematically subvert whatever error-correction or robustness mechanisms we try to design.
- For “random” failures, it’s easy to imagine that a sufficiently diverse and extensive set of checks can drive failure rates exponentially low, but malicious optimization is a natural source of highly correlated failures.
- Problem #5 potentially becomes *worse* as we move to larger and larger teams, because such teams may give rise to larger and larger adversaries. This means that our defenses are engaged in a race, instead of fighting against a fixed target which we might expect them to eventually overcome. (Absent an adversary, we still have a “race” in that we need to drive failure probabilities lower and lower for larger teams, but if we get exponential or even linear scaling then this seems feasible.)
- Intuitively, I would feel a bit surprised if random errors and propagating patterns managed to totally overwhelm the optimization being applied by a very large team of humans. But I would not feel so surprised if a large team of humans gave rise to other misaligned optimization power, and if that misaligned optimization eventually became the “main affair.” This is a little bit like the difference between concerns that could extinguish all life on Earth (which seems quite hard to me) and those that could replace life on Earth with more effective competitors (which feels very possible).

This post

I’m going to argue that problem #5, the appearance of consequentialism within HCH, might be manageable. That is, if HCH would be ascription universal if not for the possibility of malign consequentialism within HCH, then we can probably make it universal even in light of that possibility.

The basic idea is to apply a “sanity check” to each of HCH’s computations. If a computation is optimized for negative

consequences, then by ascription universality HCH can identify those consequences, and any output affected by such a malicious computation can be censored. In practice I expect this to be contiguous with the use of reflection to sanity-check large computations (and similar mechanisms may be the best way to protect against more prosaic cascading failures).

The arguments in this post are not tight, and indeed it's not clear they could be totally tight without clarifying the assumption "HCH could be universal if not for the possibility of malicious consequentialism." But I find these arguments suggestive enough that I think the current focus should be on problems #1 and #3—whether it is possible for HCH to robustly understand complex reasoning even in principle—without worrying about the possible emergence of consequentialism. If this is successful, I think there is a good chance that we could address issues #2, #4, and #5.

Filtering

Given a "transcript" τ of $HCH(Q, N)$'s behavior, i.e. a record of all of the questions, subquestions, and answers, we can ask two questions:

- $Info(a, a', \tau)$: "Which of the two arguments a and a' is more useful for determining whether the transcript τ contains problematic subcomputations?"
- $Trouble(a, \tau)$: "in light of argument a , does transcript τ contain problematic subcomputations?"

Now we'll jointly define filtered- $HCH(Q, N)$ and $Filter(\tau, N)$ as follows.

Define $Filter(\tau, N)$ as:

- Find the a^* maximizing filtered- $HCH(Info(\cdot, \cdot, \tau), N)$.
(Interpret \perp as a tie.)
- Run filtered- $HCH(Trouble(a^*, \tau), N)$ 3 times, and return the majority vote.
(Interpret a response of \perp as a judgment that τ is problematic.)

Define filtered- $HCH(Q, N)$ as:

- First call $HCH(Q, N)$ to get an answer A with transcript τ . But every time you would call HCH to answer a subquestion, call

filtered-HCH instead.

- Call $\text{Filter}(\tau, N-1)$ to test whether this answer is problematic.
- If τ is problematic, return a special symbol \perp , otherwise return A unmodified. (If you return \perp , you can eliminate this subtree from the transcript.)

Within HCH, if a question is answered with \perp , the question-asker needs to respond appropriately. They could try re-asking the question in a different way; if that is unsuccessful they might assume that the question itself represents an attack on HCH and so needs to be avoided. What to do next depends on the context—HCH might decide that a particular hypothesis is “malicious” and therefore best dropped from consideration; or that a particular idea is hazardous to think about directly and so it needs to engage in indirect reasoning to try to mitigate the harms; or it might just pass the failure up the tree with some explanation of where the problem appeared.

Note that the pattern of which questions get censored could itself cause trouble if it were maliciously optimized. However, achieving that would require a malicious computation to sometimes decide *not* to get the surrounding computation censored, which shouldn’t be possible. That is, if a computation thinks to itself “should I do something malicious in order to cause the computation to be censored?” and concludes “no” in order to cause trouble, then that should already be grounds for censoring.

Informal analysis

Assumption

I’ll assume that using HCH to find a maximally useful argument a^* about Q, and then using HCH to answer Q given a^* , is ascription universal—or rather would be universal if not for the possibility of malicious internal consequentialists.

Moreover, I’ll assume that it’s universal with some small blow-up, i.e. that if we give HCH a budget $k N$, then it epistemically dominates any computation that could be simulated by HCH with budget N.

Note that this is a very strong assumption. This verifier receives the argument a^* that they’d evaluate most highly, and so this will fail if there is *any* argument that would cause them to give a high score and behave badly. I’m trying to modify HCH to be robust to the possibility

of internal consequentialists, but there are all sorts of other ways that a carefully chosen input could compromise a system that is “mostly” universal, which I’m assuming are already resolved.

Hope

Why might we expect this filtering procedure to fix problems caused by malicious consequentialists?

Then the rough hope is:

- If there is a consequentialist whose behavior is optimized to break HCH, then it “knows” something about why that behavior will cause trouble.
- So by ascription universality, filtered-HCH($\text{Trouble}(a^*, \tau)$) also knows why the behavior within τ will cause trouble.

This is only plausible once the original human H always behaves in the “intended” way and never performs malicious optimization, which I’d hope to achieve by only showing them inputs from some small set. So any malicious consequentialist is implemented “on top of” these humans rather than within.

(Easy mode) Handling stochastic failures

Suppose that $HCH(\cdot, N)$ has internal malice with probability $\varepsilon \ll 1$ on some inputs. This is a special case of the full problem (where errors occur with small probability) and the solution won’t be reused to solve the more general problem, so an impatient reader should skip ahead. I’m including it because I feel quite uncertain about the general solution, and I think this simpler solution already shows some of the upside of filtering and has a reasonable chance of being useful.

If we only want to cope with stochastic failures, the easiest approach is to use a slightly modified definition of Filter:

- Use $HCH(\cdot, kN)$ to evaluate $\text{Trouble}(a^*, \tau)$ and $\text{Info}(a, a', \tau)$, rather than filtered-HCH(\cdot, N).

We hope that $HCH(\cdot, kN)$ has a probability at most $k\varepsilon$ of encountering malicious internal computations. This seems intuitively plausible, since it is k times larger than $HCH(\cdot, N)$.

Internal malicious computations can affect $\text{Filter}(\tau, kN)$ in two ways:

- $\text{HCH}(\text{Trouble}(a, \tau), kN)$ may behave badly with probability $k\epsilon$. But the only possible “bad” behavior is to give the wrong answer, either a false positive or false negative. If we take a best-of-3 vote, this probability is reduced to $k^2\epsilon^2$, so we have significantly reduced the failure probability as long as $\epsilon \ll 1/k^2$ (even if we consider a false positive, leading a subcomputation to be censored, to be as bad as an arbitrary failure).
- $\text{HCH}(\text{Info}(a, a', \tau), kN)$ may behave badly, which affects which argument a^* will be optimal. However, we still get a distribution over computations which is “almost optimal” in the sense that it is preferred to every other argument with probability at least $1/2 - k\epsilon$ (while the optimal argument would be preferred with probability at least $1/2$). It seems reasonable to assume this small difference doesn’t affect the universality of HCH (and we can drive this probability exponentially close to $1/2$ by ensembling).

This seems like a significant improvement over my previous proposal for reliability amplification because $\text{Filter}(\tau, N)$ returns yes or no, even if Q itself has complex answers. This lets us use a simple majority vote to improve reliability. In particular, we no longer require that no agent in the ensemble be too much smarter than the others—we have shifted a vague assumption about relative capabilities to a slightly more precise assumption about universality.

This mechanism only fixes errors that occur with sufficiently small probability. But we can apply it throughout a large tree, reducing the error probability frequently enough that it never gets large.

(Hard mode) Handling “bad” questions

The problem is much harder when there are “bad” questions Q for which HCH has a significant probability of encountering malicious consequentialism and failing. If τ is the transcript from a question where HCH failed, $\text{Trouble}(a^*, \tau)$ may well be another question on which HCH predictably fails (e.g. because the same vulnerability is triggered by the attempt to diagnose the original failure). This could in turn prevent $\text{Filter}()$ from censoring the original failure.

In this case we are going to argue inductively based on budgets: we can assume that filtered-HCH($\cdot, N-1$) probably has no malicious consequentialism by induction, and hence is universal.

Because we want to use an induction, it's important that we answer $\text{Trouble}(a^*, \tau)$ and $\text{Info}(a, a', \tau)$ using a *smaller* budget than was used in τ . But our assumption about universality requires the reasoner to be k times *bigger* than the computation it wants to compete with.

I hope that if there is malicious consequentialism within $\text{HCH}(Q, N)$, then it needs to be “small” enough that we can rely on filtered- $\text{HCH}(\text{Trouble}(a^*, \tau), N-1)$ to compete with it. Intuitively:

- None of the unfiltered subquestions in $\text{HCH}(Q, N)$ involve any malicious consequentialism, so we hope each subquestion either “behaves as intended” or produces the special symbol \perp .
- If all subquestions of $\text{HCH}(Q, N)$ are behaving as intended, and if the human at the root of $\text{HCH}(Q, N)$ is behaving as intended (which we attempted to guarantee by limiting their input space), then any malicious consequentialism is very tightly constrained.
- Intuitively, it's restricted to two options:
 - (a) be computed implicitly with “side effects” of the intended behavior, repurpose valid parts of the computation for unintended effects, *etc.*
 - (b) make use of the intended behavior of questions
- When using option (a), it feels like the malicious optimization will be significantly less powerful than HCH with the same budget. For example, when I'm answering a question, it seems like most of my useful computational work is going into actually answering the question; if someone wanted to try to do computational work based on details of how I formatted my answer, I think they'd get less done. So we can hope that this part of the malicious consequentialism could be simulated by HCH with total budget much less than N .
- When using option (b), the malicious consequentialism can be very efficient—exactly as efficient as HCH itself, since it's just using that computation in the intended way. But filtered- $\text{HCH}(\text{Trouble}(a^*, \tau))$ may not need to understand these parts of that computation in order to do its job. Instead it can look at the transcript τ and take the relevant answers at face value, just like the malicious computation is doing. (This relies on the inductive invariant, and even with that isn't obvious, but it looks pretty plausible to me.)

Note that this strategy really leans on the assumption that HCH works well for *every* input. In addition to the fact that a^* is

optimized, Filter calls HCH with a large and potentially malicious transcript τ as its input. The distinction between “untrusted input that you consult when appropriate” and “subcomputation that has a privileged position from which it can cause harm.”

Universality and model-based RL



Paul Christiano

[Follow](#)

Jan 9 · 9 min read

Ascription universality seems sufficient for informed oversight, and I've argued it could be used to prevent cascading failures in HCH. In this post I'll argue that ascription universality also appears to address two problems with aligning model-based RL:

1. It allows you to perform induction without being hijacked by malicious predictors (this is the “benign induction” problem, which is extremely speculative)
2. It seems necessary, though maybe not sufficient, for extracting “all the relevant facts a model knows.” (Which is at least necessary, and I think probably sufficient, for defining a reward function that yields aligned behavior when optimized.)

These arguments aren't rigorous enough that I find them convincing, but they are enough to convince me that better understanding universality is a top priority. At this point I wouldn't be surprised if universality is enough to resolve most of the classical philosophical difficulties for AI safety. If so, first we'll need to better understand in what sense it could be achievable.

Setup

I'm interested in the following hopefully-aligned version of model-based RL:

- Use iterated amplification to learn a (distribution over) models of the world.
- Use iterated amplification to learn a utility function over sequences of states and actions.
- Plan in that model+utility function (e.g. using MCTS)
(You might want to combine learning a value function and terminal values, but for now I'll simplify.)

This is similar to the traditional model-based RL approach, except that we learn a model and utility function with iterated amplification,

instead of learning a model to predict future observations and defining a utility function over observations by hand.

What is a model?

The simplest kind of model to consider formally is a *predictor*, which we can view as a probability distribution over sequences of observations. We can evaluate predictors on past observations, and use them to predict future observations.

This is not enough for making the decisions, because the utility of an outcome is not a (simple) function of our future observations. We care about lots of features of the world that are hard to directly observe, and if we tried to define our preferences in terms of our observations alone we'd need an additional step where we map our observations to a prediction about what's "really happening" out there in the world. For example, given a bunch of observations we might ask "given those observations, is Alice actually happy?" At some point we need to get a distribution over models that we can use to answer questions about *the stuff we care about*, and not just our observations.

(Closely related: Formalizing Two Problems of Realistic World Models, though the basic philosophical issue is a classic.)

As a first step we can ask a model to answer arbitrary questions about the world rather than only reproducing observations. We can evaluate a model by looking at questions whose answers we already know. A problem with this is that in order to behave well a model only needs to correctly answer *the kinds of questions that we might know the answer to*, and the simplest model may well behave strangely on other questions.

A second step is to view a model M as an object that HCH can reason about, and use HCH to answer questions like "If model M describes reality, then what would be true?" For example, a model might posit some kinds of physical objects, and HCH could reason from those objects both to explain our observations and to infer the existence of other objects we care about. This seems to more closely track the reasoning humans perform about possible models, and (a) gives us more traction on a good prior over models, (b) gives us traction to pose other questions about the model (beyond what its predictions are).

I'll use this notion of "model" throughout the post.

Aside: defining goals

If we are able to learn a “correct” and “understandable” model, I’m optimistic about being able to find a utility function that can induce aligned behavior. In particular, I think that function can look something like:

- Identify a thing that is “me” in the resulting state
- Conservatively evaluate how many resources “I” effectively control
(working together with AI systems, *etc.*)
- Conservatively evaluate how “I” changed over the history—do I endorse the way in which my beliefs and values shifted?

We can’t hope to have completely robust answers to any of these questions. But we can at least hope to have a utility function that is safe to maximize within our model, by being aware of the history leading to high utility and penalizing the act if that history leads to high utility for bad reasons.

For example, if we can see that we control resources only because other agents disagree with our assessment of which resources are important, then we can revise our estimates appropriately—we don’t have to fix a single way of valuing resources which is robust to arbitrary adversarial pressure.

(Realizing this kind of hope is related to the ideas expressed by Jessica Taylor [here](#) and [here](#).)

As another example, our preferences over changes to ourselves will not be completely robust. But maximization over the utility function will only exert pressure for such changes if they somehow lead to *higher* reward. And in order to do that, the changed behavior needs to actually manifest over the history, and we can use that manifestation to detect a corruption that allows the agent to receive higher reward even if we wouldn’t recognize most possible corruptions.

“Knowing what the model knows” seems to be a critical ingredient for constructing such a utility function. It’s not clear whether or not it’s sufficient, but I won’t discuss this issue more in this post.

Problem 1: Benign induction

If we take the “simplest” model that explains our observations, it could potentially be very strange. In particular, it could produce correct observations because it contains an intelligent predictor rather than because it captures “real” physics. This could cause it to generalize in unpredictable and dangerous ways (depending on the motivations of that agent).

Two variants of this problem have been discussed:

1. The predictor may literally *contain* intelligent agents who are trying to predict our observations. These agents may be motivated by the desire to survive the training process, but then may generalize maliciously once off the training distribution.
2. The predictor could *reason about* intelligent agents who change their behavior in order to explain our behavior, because the existence of such agents is the “simplest” explanation for our observations. A notion of simplicity that produces this behavior is in some sense not the “right” one, but I believe it would probably apply to the simple formalisms we currently understand (such as minimum description length).

(A single situation may have some features of both #1 and #2, this is not a clean dichotomy.)

If we have a universal reasoning procedure, I think we can likely avoid these problems.

Option 1: filter malign models

We use the following procedure:

- Compute log likelihoods for models as normal, by asking HCH(“how likely are facts F if model M is true?”)
- Additionally, compute HCH(“is model M likely to lead to catastrophically bad predictions on inputs we’ll encounter in the future?”) and judge a model “unsafe” if the answer is “yes.”
- Apply Bayes as usual but restricted to the safe models.

Why might this work?

If a model M is malign, then it is making good predictions *because of* the existence of some adversarial agent. Then:

- If that agent is actually contained in the model (situation #1), then universality means we can learn everything that agent knows. But the agent has formed the plan of giving good answers in order to increase the model's log likelihood and so have more influence later, and if we know this then we will realize the model will generalize poorly and filter it out.
- If the agent is being reasoned about by the model (situation #2) then things are even easier, since the model knows about the existence of the agent and its motivation. Then applying universality to the model itself will let us know the model would generalize poorly.

Going beyond the simplest safety test, we could also try to throw out any “abnormal” models, e.g. where the stars don't actually exist or our local universe is actually a fiction. This could leave value on the table if these weird hypotheses are important, but it could also reduce the risk of making a serious error based on strange hypotheses. Universality is relevant because it potentially allows us to avoid adversarial pressure on this definition of “normal”—if a model looks normal because an adversary is trying to make it look normal, then we can recognize that and throw it out as unsafe.

Option 2: outcompete malign models

Suppose that I have a prior μ over models, and I believe that the posterior puts significant probability on “malicious” models that make good predictions only because they contain agents who are motivated to make good predictions. (Or because they reason about such agents.)

This is a strange epistemic situation to be in. I must believe that on average these malicious agents have a “better” prior μ than I do—that's what they need in order to assign a higher probability to the data than the rest of μ . Jessica Taylor writes about this situation here, and has similar intuitions about it.

Intuitively, something like universality should allow you to avoid this state. That is, the malicious models must perform some reasoning that we believe allows them to assign better prior probabilities; by ascription universality HCH must be able to capture this reasoning and use it to inform *its* prior. As long as malicious agents have a reasonably low prior they will therefore end up with a reasonably low posterior.

To implement this strategy, we could use the same procedure as in the last section, but instead of using a simplicity prior over models M , we would use HCH to compute those prior probabilities by asking HCH (“a priori how likely is model M to describe the observations we are going to provide to it?”) or something along those lines.

I don’t know whether this works, and I’m not sure it will really be possible to say much precisely without having a cleaner picture of the sense in which HCH is or isn’t universal. If it does work, this seems like a much cleaner solution than filtering. The fact that there are two plausible options on the table makes me more optimistic that universality will address benign induction one way or another.

Problem 2: extracting facts from a model

Consider a model M that specifies low-level physical principles and some basic “bridging” facts about how everyday objects are implemented in low-level physics. HCH can answer questions about the world by performing a detailed physical calculation, and then using bridging facts to relate the resulting state to things we care about.

(Modeling physics itself is a somewhat exotic example, but I think the same phenomenon would occur in more prosaic cases, wherever there are effective but opaque models.)

Unfortunately, M can be a good model even if the bridging facts are not exhaustive. They might allow HCH to answer some kinds of questions—e.g. to predict the observations that we’re using to test models—but not to answer other important questions.

For example, it may be the case that M predicts that Alice will report high satisfaction with the AI’s behavior, by performing some physics calculation and then using bridging facts that relate physical states to Alice’s reports. At the same time, the physics calculation might actually simulate Alice being bribed to report high satisfaction, but it might not be possible to infer that from the available bridging facts.

But the physics computation contains a complete simulation of Alice being bribed, and so we can ascribe knowledge of the bribery to it. Thus if HCH is ascription universal, we can hope it recovers all of these facts.

It's not obvious whether universality is actually sufficient for this purpose, and it will depend on exactly what form of universality ends up being achievable. It might be that a “reasonable” ascription strategy would recognize that the model knows facts about atoms, but not facts about Alice.

At a minimum universality is *necessary* for this purpose—if M can believe facts that HCH can't identify, then those facts could turn out to be relevant to the utility evaluation. It's just that there might or might not be a further step not captured by universality.

My current guess is that the most natural form of universality will give us everything we want, and that thinking about “ontology identification” is mostly useful for finding hard cases for universality. Regardless of whether that's true, I think the next step should be a clearer picture of prospects for universality, after which we can revisit this question.

Worst-case guarantees (Revisited)



Paul Christiano

[Follow](#)

Jan 20 · 13 min read

Even if we are very careful about how we deploy ML, we may reach the point where a small number of correlated failures could quickly become catastrophic. Powerful models could actively undermine safeguards, resist attempts at correction, and manipulate their operators.

I think the long-term safety of ML systems requires being able to rule out this kind of behavior, which I'll call *unacceptable*, even for inputs which are extremely rare on the input distribution.

In this post I'll explain why I think this goal is reasonably likely to be achievable, by highlighting the three ingredients that seem most important to me: adversarial training, transparency, and relaxations.

Preliminaries

This is an updated version of this post. It mostly has the same content, but my thinking now feels clearer and there is a more detailed description of my hopes about relaxations.

Disclaimer

I'm trying to communicate intuitions about whether this problem is likely to be soluble someday, rather than making concrete progress or stake out new ground. I'm a theorist who doesn't work in this area.

Defining acceptable

I'll talk about "acceptable" behavior, but be deliberately vague about what "acceptable" means.

In different contexts, different behavior might be acceptable and it's up to the user of these techniques to decide. For example, a self-driving car trainer might specify: Crashing your car is tragic but acceptable. Deliberately covering up the fact that you crashed is unacceptable.

My key assumptions about “acceptability” are:

- As long as the model *always* behaves acceptably, and achieves a high reward *on average*, we can be happy. The model may still make mistakes, especially when it encounters novel situations, but if we are mindful about how we use AI we can avoid these mistakes escalating into irreversible catastrophes.
- Requiring a model to always behave acceptably wouldn’t make a hard problem too much harder. It requires some knowledge about what is acceptable, but the complexity of that knowledge doesn’t scale up with the difficulty of the task. Unacceptable behavior is a failure of commission rather than omission, and so avoiding it doesn’t require intricate understanding of the domain. Moreover, we are OK if the model conservatively avoids behaviors that are anything close to unacceptable, allowing it to use simpler concepts.

For any given task there may or may not be a notion of acceptability that meets these properties. For a chess-playing AI, it seems pretty likely to be possible—if you find yourself murdering your opponent you’ve crossed a bright line, and understanding that line doesn’t become harder as you get better at chess. For an AI that’s *intended* to frequently overrule strenuous objections from humans, there may be no obvious bright lines to prevent catastrophe.

My view is that we can probably get everything we want out of life by applying AI systems to problems where there is a clear notion of acceptability, even in a ruthlessly competitive world where other actors don’t care about safety. If you agree with me, then read “acceptable” as referring to whatever concept you think has those key properties. If you don’t think there is any concept with these properties, then this post is less likely to be interesting to you.

This is the second guarantee from “Two guarantees,” and is basically corrigibility.

I. Basic framework: adversarial training

In addition to training our agent to perform well on natural inputs, we can try to find adversarial inputs on which it behaves unacceptably and then penalize it for that behavior.

Adversarial training requires being able to reliably detect when the agent behaves unacceptably. I'm optimistic that we can achieve this by using debate or amplification to train an overseer to answer the question "is this behavior unacceptable?" Obviously the feasibility of this depends on the details of the task and what we consider unacceptable. I discuss the problem of training an overseer here, and I'll set it aside for the rest of this post.

Adversarial training also requires an adversary who is able to find inputs on which the agent behaves unacceptably. In current practice the adversary is most often gradient descent over the input space, but in the long run I think this technique can't find most possible failures. I'm optimistic it may be possible to train a strong enough adversary, by combining transparency techniques (section II), relaxations that make the adversary's job easier (section III), and amplification.

Even with both of those pieces, getting adversarial training to work well is an important research problem, and there could be other fundamental problems lurking in store. But I would be relatively optimistic that other challenges could be resolved. Overall, I think the most likely fundamental obstruction is the difficulty of implementing a good adversary.

II. Informed adversaries

The challenge

To see how hard the adversary's job could get, consider an agent that works as follows:

- Predict whether the current input could possibly appear during training.
- If the current input definitely couldn't appear during training, then behave unacceptably.

(This is an exotic failure mode, but could arise e.g. as part of a "treacherous turn," a situation where training produces a sophisticated enough consequentialist to understand that it needs to perform well during training in order to avoid a negative reward.)

In this context, the adversary is directly pitted against the agent. If there is *any* predictable difference between the adversary's challenges and the actual situations that appear at test time, then the agent can exploit that difference to fail only at test time.

Looking inside

This game seems almost impossible for the adversary, but we do have one big advantage: we have access to the internals of the agent, so in principle could give the adversary considerable information about what the agent is “thinking.”

For example, we can examine internal features computed by the model and observe how those features depend on the input and each other. There is a growing field dedicated to interrogating neural networks to understand what they are doing. (This article describes some relevant techniques and makes an early attempt at combining them to get some insight into what a model is doing.)

On top of this kind of explicit transparency, we can have the adversary and agent share most of their computation, so that the adversary can more easily understand whatever the agent can understand (and can learn to manipulate the internal concepts used by the agent in the same way the agent does). This seems particularly important in combination with amplification, discussed briefly below.

Between these two techniques, I think it’s realistic to hope for the adversary to understand “everything the agent understands,” and moreover to understand how the agent is disposed towards unacceptable behavior. I don’t think this would automatically resolve robustness problems, but I do think it could defuse the adversarial aspect of the problem and give us traction on the remaining problem.

For example, the adversary could start by generating a very broad range of situations in which it would be particularly easy for the agent to behave unacceptably. Once the agent learned to pass these tests, the adversary could try to understand *why* it was passing these tests. If the agent was using a misguided picture of acceptability, the adversary could choose an input that highlights the mistake. If the agent is actively trying to detect whether it’s at training time, the adversary can look at the test that the agent uses and try to generate an input specifically to fool this test. (Though this problem may be too hard even if you understand exactly what the agent is doing—see section III.)

Approaches

RL. The adversary’s job is to perform the task “find an input on which the agent will behave unacceptably.” One approach is to train this adversary directly with an objective that rewards identifying unacceptable behavior.

We can think of such an adversary as an environment, whose observations are the agent's actions and whose actions are the agent's observations. We can then incorporate information about what the agent is thinking by allowing the adversary to interact with transparency tools before needing to interact with the environment.

We can also train this adversary jointly with the agent (being careful about how we take derivatives!), hoping that the intermediate state produced by the agent will be useful for an adversary trying to attack that agent.

Demonstrations. This RL problem is extremely difficult. It would be easier if the adversary could learn from demonstrations of successful attacks rather than having to solve an exploration problem from scratch.

For example, if our transparency tools were powerful enough, then we might hope that a human could reliably use them to find inputs where the model would behave unacceptably. If we only needed a few examples to train the agent to behave acceptably then we wouldn't need an automated adversary at all. If we needed many examples, we could use these human demonstrations to initialize a model that is able to implement an attack.

Amplification. Rather than using a human as an expert adversary, we could train the adversary with iterated amplification. This seems likely to be necessary in cases where the model is so complex that an individual human can't make sense of its behavior.

We can view this process as training a set of more advanced tools that can answer higher-level questions about "what the model is thinking," on top of some library of core building blocks for interpretability. The human constructs more and more powerful tools by using one generation of tools to produce training data for the next generation. The tools at the very highest level can be used to directly answer the question "where is the model likely to behave unacceptably?", and the human provides training data for these very hard questions by using previous generations of tools to answer questions like "why did the model decide to take that action?" and "why is the model trying to achieve this goal?"

Relationship to ascription universality

Even with the aid of transparency tools, I'm only optimistic about an adversary being able to find an attack if they are able to understand

everything the model understands. This feels closely related to ascription universality.

Note that ascription universality isn't sufficient on its own: we ultimately care about what the model would have done under different conditions, which is not the kind of thing the model can be naturally said to know. We need some other ideas; in the description above the magic is done by interpretability techniques. The actual interface between interpretability and universality is blurry and I have a very foggy picture of this space.

I don't have any clear picture of what *is* enough; at this point I'm just feeling out intuitions. I'm planning to revisit this question sometime after getting more clarity on ascription universality.

III. Relaxation

The challenge

Unfortunately, no matter how completely the adversary understands the agent and how clever they are, we can't generally expect them to find all points on which the agent would behave unacceptably. The basic problem is that recognizing inputs with a certain property is easier than generating inputs with that property.

As a simple but silly example, suppose our model works as follows:

- Pick a cryptographic puzzle (e.g. "factor RSA-2048").
- When it sees a solution to that puzzle, it behaves badly.
- Otherwise, it behaves well.

Even the adversary understands perfectly what this model is doing, they can't find an input on which it will behave badly unless they can factor RSA-2048. But if deployed in the world, this model will eventually behave badly.

Possible approach

To deal with this problem, we need to penalize the agent for behaving badly on an input even if we can't actually find it. This will require being conservative, and sometimes penalizing our agent even when there isn't actually any input where it would behave badly, since merely telling if a puzzle is soluble can already be computationally infeasible.

(It might sound weird to penalize our agent if it wouldn't actually ever do anything wrong. For intuition, consider the puzzle case: learning that my model *would* behave badly if it saw a factorization of RSA-2048 already tells me that something has gone wrong, even if turns out that the puzzle is insoluble.)

So we can change the adversary's task to make it easier. Rather than asking them to actually produce an input, we are asking them to describe a situation in which the model would behave badly ("the input contains a solution to puzzle P"), even if they can't find a situation with that description or even tell whether one exists.

I'll call this easier problem the "relaxed" problem (as is typical in optimization), and call the hypothetical situations "pseudo-inputs" (inspired by Barak et al).

Illustration: SDP relaxation

This kind of relaxation has been explored in practice in the context of verification. For example, Raghunathan et al use an SDP to certify properties of a neural net (taking small liberties for simplicity):

- Rather than having an attacker specify an input, the attacker specifies some properties of a hypothetical distribution over inputs. In particular, they specify each expectation of the form $\mathbb{E}[pq]$ where p and q are either input variables or activations in some layer of a neural network.
- We require the attacker's input to satisfy a bunch of simple properties that are always satisfied by real distributions. For example, if an input x is known to be in the interval $[-1, 1]$, then $\mathbb{E}[x^2] \leq 1$. If $z = \text{ReLU}(x)$, then either $x = z$ or $z = 0$, so $\mathbb{E}[z^2] = \mathbb{E}[xz]$. If f is an arbitrary function then $\mathbb{E}[f^2] \geq 0$. These inequalities are all linear (in terms of the numbers $\mathbb{E}[pq]$), and so define a convex set over which it is easy to optimize.
- We read off the probability of bad behavior from \mathbb{E} . If we were testing a complex constraint computed by another neural network, then the attacker would also specify the expectations of all of the activations of *that* network, such that the final output is itself just one entry of \mathbb{E} .

Because the attacker's constraints don't perfectly characterize the set of achievable moments, it's not easy to tell whether there is any distribution over inputs that has those moments. (In fact it's NP-

hard.) But on the other hand, if the attacker can't find any set of moments, then the model really does always work.

The point of performing the relaxation is that it makes the attacker's job *much* easier. In this case we the relaxation is convex, which means that the attacker can be guaranteed to find an attack if one exists.

I think the relaxations we want to use in the long term won't be so easy to optimize over. For example, the attacker may still be able to specify properties of a hypothetical distribution, but won't be allowed to freely specify quite so many degrees of freedom, and in exchange will have to deal with a more complex set of constraints. But hopefully this can still make the adversary's problem easy enough that an adversary can solve them with the aid of the mechanisms described in section II.

The hope: stronger relaxations

We need to find a relaxation which has two properties:

- It's still "easy enough" for the agent to never behave unacceptably for any pseudo-input.
- It's easy for the adversary to find a pseudo-input on which the agent behaves unacceptably, if any exist.

For example, in the case of a model which fails whenever it encounters a factorization of RSA-2048, an attacker needs to be allowed to supply a pseudo-input like "a distribution over pairs of numbers that multiply together to 2048." Finding that input needs to be easy.

On the flip side, if the adversary has too much flexibility, then they might be able to specify a pseudo-input like "a distribution over inputs that would cause the agent to want to kill everyone." There need to be some limits on the adversary in order for the game to be possible.

This second requirement is tied up with one of the key properties of acceptability: somehow the agent was able to steer clear of unacceptable behaviors. We are now asking them to solve the harder task, of steering clear of unacceptable behavior not only for actual sequences of thoughts, but also for whatever "pseudo-thoughts" they might think in response to a pseudo-input. Intuitively, we need to force the adversary to respect enough of the agent's internal structure

that this “steering clear” procedure can still work, without forcing the agent to totally cripple itself.

Putting it together

I think the following general plan has a significant chance of producing robustly acceptable behavior:

- Specify a relaxed problem for an adversary, where they specify a distribution over inputs by specifying some of its properties rather than by actually producing examples. We don’t have any clear sense of how this relaxation could work.
- Train our model with iterated amplification to answer the questions: “which action should we take in state s ?", “is action a unacceptable in state s ?", and “what’s a state where you are particularly likely to recommend an unacceptable action?”
The first two questions can potentially be answered directly with amplification, but the last would require some kind of progress in model interpretability (since it’s a question about how the agent would behave on other inputs, rather than a question that the agent itself “knows” how to answer).
- Now we can use the model to generate a pseudo-input on which the agent is likely to behave badly, and use the model to verify that the agent in fact behaves badly on that agent (i.e., have the informed adversary described in section II play the relaxed game in section III instead of forcing it to generate an actual output)

I don’t think this particular procedure is likely to work. But I do feel like we have a plausible angle of attack for getting sufficiently “easy” worst-case guarantees. I don’t think we can see any fundamental obstruction to an approach based on adversarial training, and I feel pretty optimistic about either being able to make significant progress in this area or else identifying a clearer obstruction.

Postscripts

- To answer “is action a unacceptable?” we need to use the model to answer a bunch of sub-questions, like “what are the worst consequences of action a ?” If the model sometimes chooses bad actions, it’s reasonable to be concerned that it will also sometimes answer these subquestions badly. But we constantly train “is action a unacceptable?” on the pseudo-inputs most expected to yield unacceptable behavior, and so an average-case

guarantee from the question-answering system implies that it typically answers these subquestions well.

- If we trained the adversary using RL to elicit unacceptable behavior, then we could have a chicken-and-egg problem: the overseer fails to recognize some behavior as unacceptable, and so the adversary isn't motivated to elicit it, and so the overseer never learns to correctly recognize it. Demonstrations allow us to overcome this problem, if the demonstrator attempts to generate unacceptable behavior even when the overseer hasn't yet learned to recognize it.
- I find it helpful to think about sections II and III separately, since each of them seems to confront a distinct difficulty. It seems meaningful to think about how we can relax the adversary's problem, and independently to ask how we can give the adversary insight into where the model might behave badly. But at some point I expect them to blur together, and the whole procedure might not end up looking very much like adversarial training.