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 ith 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.