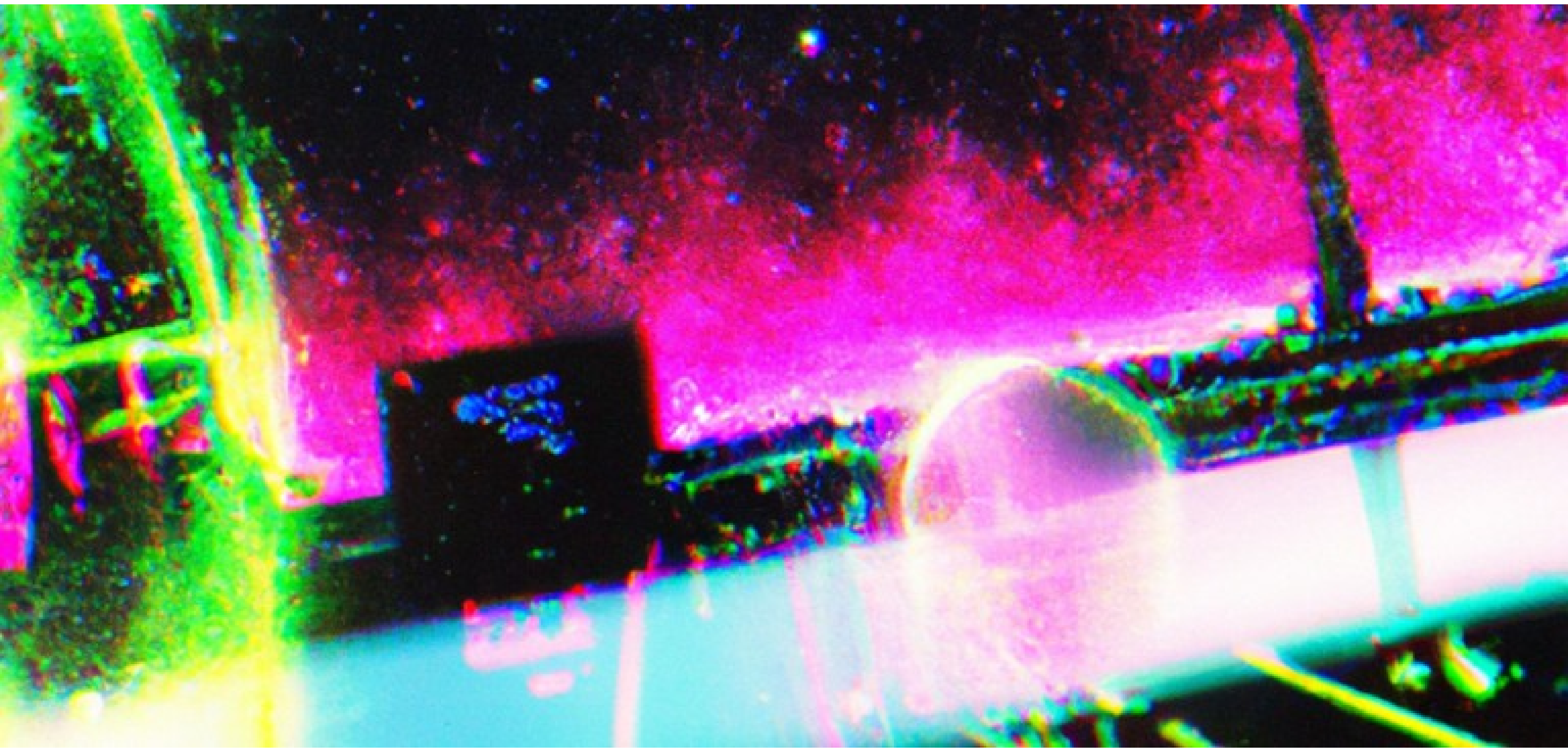


Alignment Stream of Thought

[ASoT] Observations about ELK

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

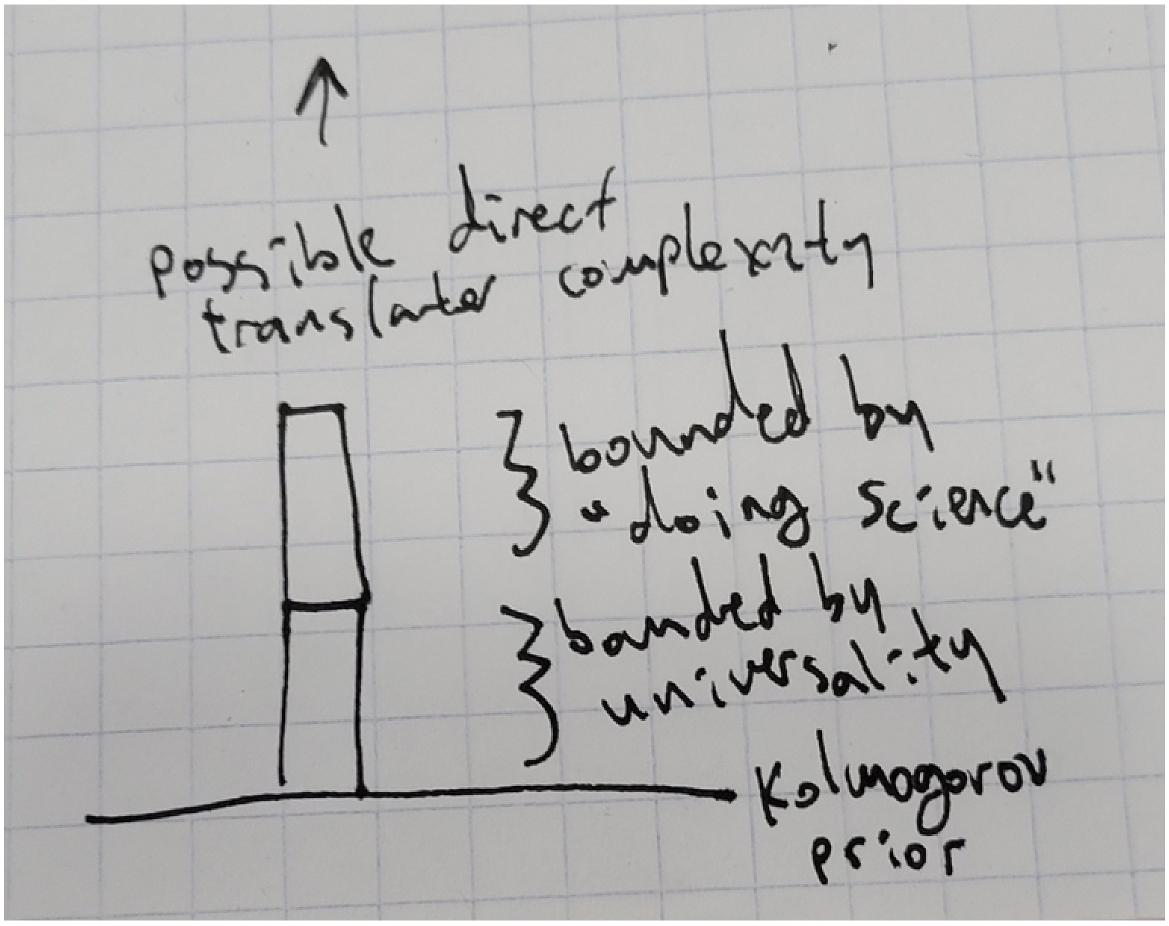
This document outlines some of my current thinking about ELK, in the form of a series of observations I have made that inform my thinking about ELK.

Editor's note: I'm experimenting with having a lower quality threshold for just posting things even while I'm still confused and unconfident about my conclusions, but with this disclaimer at the top. Thanks to AI_WAIFU and Peter Barnett for discussions.

- One way we can think of ELK is we have some set of all possible reporters, each of which takes in the latent states of a world model and outputs, for the sake of concreteness, the answer to some particular fixed question. So essentially we can think of a reporter+model pair as assigning some answer to every point in the action space. Specifically, collapsing possible sequences of actions into just a set of possible actions and only worrying about one photo doesn't result in any loss of generality but makes things easier to talk about.
- We pick some prior over the set of possible reporters. We can collect training data which looks like pairs (action, answer). This can, however, only cover a small part of the action space, specifically limited by how well we can "do science."
- This prior has to depend on the world model. If it didn't, then you could have two different world models with the same behavior on the set where we can do science, but where one of the models understands what the actual value of the latent variable is, and one is only able to predict that the diamond will still appear there but can't tell whether the diamond is still real. A direct translator for the second will be a human simulator for the first.
- More generally, we don't really have guarantees that the world model will actually understand what's going on, and so it might genuinely believe things about the latent variable that are wrong. Like, we might have models which think exactly like a human and so they don't understand anything a human wouldn't understand, so the direct translator for this model would also simultaneously be a human simulator, because the model literally believes exactly what a human would believe.
- There are more than just human simulators and direct translators that are consistent with the training data; there are a huge number of ways the remaining data can be labeled. This basically breaks any proposal that starts with penalizing reporters that look like a human simulator.
- I basically assume the "human simulator" is actually simulating us and whatever science-doing process we come up with, since this doesn't change much and we're assuming that doing science alone isn't enough because we're looking at the worst case.
- So, for any fixed world model, we need to come up with some prior over all reporters such that after picking all the reporters consistent with our data (action, answer), the maximum of the resulting posterior is the direct translator.
- A big assumption I'm making is that the KL distance between our prior and the complexity prior has to be finite. My intuition for why this has to be is that a) universality of the complexity prior, and b) even if our prior doesn't have to cover all possible reporters, it still seems reasonable that as complexity increases the number of reporters that our prior accepts as nonzero needs to increase exponentially, which gets something similar to universality of complexity on the subset of plausible reporters. I think this point is the main weak point of the argument.
- Since the amount of data we can get is bounded by our ability to do science, the amount of information we can get from that is also bounded. However, since we're assuming worst case, we can make the complexity difference between the direct translator and all the other junk arbitrarily big, and therefore arbitrarily less likely, and since our modified prior and our data only get us a finite amount of evidence, we can

always imagine a world where the direct translator is sufficiently complex such that we can’t pick it out.

- So to solve ELK we would need to somehow rule out a huge part of the set of possible reporters before even updating on data. In some sense this feels like we ran in a circle, because there is always the prior that looks at the model, solves ELK, and then assigns probability 1 to the direct translator. So we can always just move all the difficulty into choosing the prior and then the data part is just entirely useless. I guess in retrospect it is kind of obvious that the prior has to capture the part that the data can’t capture, but that's literally the entire ELK problem.
- One seemingly useful thing this does tell us is that our prior has to be pretty restrictive if we want to solve ELK in the worst case, which I think rules out a ton of proposals right off the bat. In fact, I think your prior can only assign mass to a finite number of reporters, because to get the sum to converge it needs to decay fast enough, which also gets you the same problem.
- Only assigning mass to a finite number of reporters is equivalent to saying we know the upper bound of direct translator complexity. Therefore, we can only solve ELK if we can bound direct translator complexity, but in the worst case setting we're assuming direct translator complexity can be unbounded.
- ~~So either we have to be able to bound complexity above, violating the worst case assumption, or we have to bound the number of bits we can get away from the complexity prior. Therefore, there exists no worst case solution to ELK.~~ *(Update: I no longer think this is accurate, because there could exist a solution to ELK that function entirely by looking at the original model. Then the conclusion of this post is weaker and only shows that assigning an infinite prior is not a feasible strategy for worst case)*


possible direct translator complexity
bounded by "doing science"
bounded by universality
Kolmogorov prior

[ASoT] Some ways ELK could still be solvable in practice

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

Editor's note: I'm experimenting with having a lower quality threshold for just posting things even while I'm still confused and unconfident about my conclusions, but with this disclaimer at the top.

This post is a followup to my earlier post.

If ELK is impossible in generality, how could we solve it in practice? Two main ways I can think of:

- Natural abstractions: maybe the complexity of direct translators is bounded in practice because the model ontology is not so different from ours, violating the worst case assumption.
- Science being easy: maybe there isn't any hard ceiling to how good you can get at doing science, and it's not actually that much more expensive so it is actually feasible to just scale science up.

For the first one, I feel like there are both reasons for and against it being potentially true. The main argument for, especially in the context of LMs, is that human abstractions are baked into language and so there's a good chance LMs also learn this abstraction. I used to be a lot more optimistic that this is true, but since then I've changed my mind on this. For one, in the limit, natural abstractions are definitely not optimal. The optimal thing in the limit is to simulate all the air molecules or something like that. So we know that even if natural abstractions holds true at some point, it must eventually stop being optimal, and the abstractions get more and more weird past that point. Then the question becomes not just whether or not there even exists a sweet spot where natural abstractions holds, but also whether it overlaps with the kinds of models that are dangerous and might kill us. I still think this is plausible but kind of iffy.

I think this is worth looking into empirically, but I'm very cautious about not reading too much into the results, because even if we see increasingly natural abstractions as size increases, it could still start going down again before our models are sufficiently powerful. I'm moderately worried that someone will overclaim the significance of results in this vein. Also, my intuition about language models is that despite communicating in language, they learn and think in pretty alien ways.

For the second one, if this is true then this kind of defeats most of the point of even doing ELK. If you can just scale up the science doing, then ELK becomes just an optimization the same way that reward modelling is just an optimization of asking humans every single time you need a reward. In that case, then doing something vaguely ELK-like might be more efficient, but it wouldn't allow us to do anything fundamentally new. That might be hugely important in practice, but would be a very different focus than what people focus on when they talk about ELK right now.

As for the other point, I'm personally not hugely optimistic about any of the existing science-doing methods, but this opinion is not based on very much thought and I expect my stance on this to be fairly easily changeable. I think this is a useful thing for me to spend some more time thinking about at some point.

Future Directions: empirical stuff testing natural abstractions, how do we do science better

[ASoT] Searching for consequentialist structure

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

Editor's note: I'm experimenting with having a lower quality threshold for just posting things even while I'm still confused and unconfident about my conclusions, but with this disclaimer at the top. Thanks to Kyle and Laria for discussions.

In "Searching for Bayes Structure", Eliezer argues that things which arrive at the truth must be doing something Bayes-like somewhere in there, even if it's extremely inefficient or accidental or filtered through some process that looks very different from the outside. If it has no Bayes structure, it literally cannot arrive at the truth.

I think it's plausible that Bayes is to epistemic rationality as consequentialism is to instrumental rationality. Here, by consequentialism I mean the ability to get to some goal from a broader set of initial states, which is how I understood Elliezer's use of the term in the dialogues. It does not have to be an EUM/VNM-rational. Importantly, to be good at consequentialism comes with all the stuff like instrumental convergence (and consequently self preservation, power seeking, etc).

In other words, there's some minimal amount of consequentialism you need to achieve a given goal, and there's no getting around this lower bound. Of course, some agents do a lot more consequentialism than is necessary, but I interpret some people (i.e Eliezer) as arguing that even this lower bound is too much consequentialism for, say, performing a pivotal action, because it's fundamentally very difficult to do. As an analogy, if your goal is to buy an apple for $1, any plans which fail to collect at least $1 will simply not be able to acquire an apple, and even though there also exist plans which collect $100 (which is more dangerous than collecting $1), if even collecting $1 is too dangerous by default, then even lobotomizing a $100-collecting agent to only collect $1 is scarcely comforting. The main lesson I draw from this is that if we want to do something consequentialist then we have to somehow figure out how to align consequentialists, and that making the agent worse at consequentialism isn't a solution. I think it also seems productive to figure out where the lower bound of how much consequentialism we need (for, i.e a pivotal action) and the upper bound of how much consequentialism is safe are relative to each other.

As an aside, my intuition for why instrumental convergence is unavoidable for consequentialists: There is some set of plans that achieve your goal with high probability across all universes, and a slightly larger superset that achieves your goal with high probability in across the universes in which you don't try to stop the system. Among those plans, under some complexity prior or something, the ones that have the safe properties we want (they actually stop when you try to stop them) occupy a very small measure, because this set is very unnatural. Think Nate's laser analogy: the plans which accomplish the goal in a way robust to many perturbations, but yet are not robust against the perturbations that correspond to us trying to stop the system, is difficult to specify. So to pull out the ones we want requires a bunch of work, which means that by default most plans that achieve our goal will be power seeking and also not corrigible.

Actually, it gets even worse, because there are also deceptive plans that look corrigible but actually aren't; it's really hard to separate these from the actually corrigible ones. So even getting really good at picking out the corrigibility cluster of things isn't good enough, because there's a big incentive for deceptive plans to look corrigible. (I think this is vaguely corresponding to outer vs inner alignment but I'm not 100% sure.) Also, there might be self-modification considerations here where non-EUMs and other lobotomized AIs might self modify to EUMs, if you lobotomize them too much, the reason being that if you lobotomize the AI too much then most plans conditioned on success start looking like "build a non lobotomized AI somehow and then tell that AI to accomplish the thing".

Tool AI

I think there's also a way in which this applies to tool AI type arguments as well. Ultimately, the reason we want tools is they help us accomplish some goal (for instance, performing a pivotal action). So, we can view the entire human+tool system as an agent that is trying to get to some goal, where there's some human oversight to ensure that the plans pass muster. This system, to get to goals, will have to do consequentialism. While having the human provide the glue between the model and the real world does add some safety, I would argue that this is approximately as safe as just humans vetting plans from an agent, which is itself not really sufficient because many plans which seem non suspicious to us are actually dangerous in some unforeseen way (diving into whether this is most plans or vanishingly few plans seems like a useful avenue - on the one hand one would think that making a plan sound useful while still doing something bad might require a lot of optimization, on the other hand "I can't see anything going wrong with this" is a hugely lower bar than "nothing will go wrong with this" resulting in most plans that satisfy the former not satisfying the latter).

While the human providing the "outer loop" so to speak does allow the human a little bit more control in that they have control over the particular reasoning processes used to arrive at the action, I don't think this is a very large effect. For one, if the tool is deceptive, it can still manipulate the human by giving them carefully crafted answers to their queries - in the deceptive case this gives almost no protection. Even without deception, if we're in a universe where "I can't see anything going wrong with this" is much looser than "nothing goes wrong" (which, for what it's worth, I think is highly likely to be true), you end up with the exact same problem again. Because there's some minimal amount of consequentialism you need to do no matter what to accomplish your goal, you also need to do some minimal amount of optimizing against your ability to see things wrong with plans, and in the universe where most ok-looking plans suck, this optimization quickly goodharts your ability to tell whether a plan is good.

This seems to suggest the human's ability to pick good plans with extremely high precision (i.e definitely not accepting any dangerous plans, though possibly rejecting some safe plans) as a core difficulty. This particular subproblem has also popped up in discussions about other parts of alignment, so I think it seems like something pretty important.

(Also, tools are economically uncompetitive anyways.)

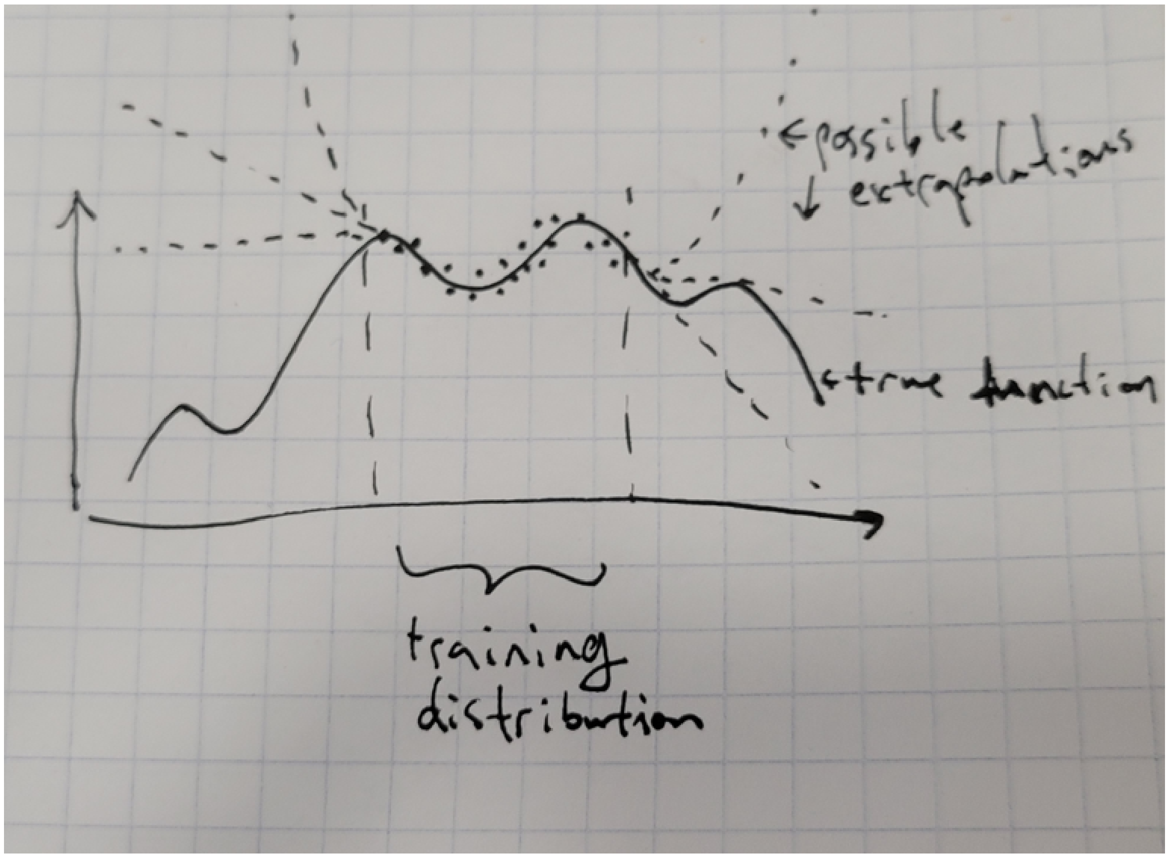
[ASoT] Some thoughts about deceptive mesaoptimization

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

Editor's note: I'm experimenting with having a lower quality threshold for just posting things even while I'm still confused and unconfident about my conclusions, but with this disclaimer at the top. Thanks to AI_WAIFU for discussions.

I think of mesaoptimization as being split into different types: aligned (which is fine), misaligned (which is bad), and deceptively aligned (which is really bad - I draw the line for deceptively aligned when it actually tries to hide from us instead of us just not noticing it by accident). I mostly think about deceptive mesaoptimization, and I think of regular misaligned optimization as basically an easier case because at least it's not trying to hide from us (what it means for it to actually be "trying to hide" is kinda fraught anyways so this sidesteps that). In particular, I think of mesaoptimizers as being dangerous because they fail in a malign way (I don't think mesaoptimization itself is necessarily bad, and preventing it doesn't seem obviously good).

OOD-focused framework of deception



One framework for thinking about deceptive mesaoptimizers that I used to use a lot was that they would be good on the training distribution, but then do bad things out of distribution. In this framework, what we would do is then to try to somehow tell during training whether a model is going to do bad things out of distribution, and only keep the ones that don’t, or something along those lines. The intuition is that the mesaoptimizer cares about something in the real world, but it knows that if we catch it then it will not be able to achieve its goals in the real world, so it behaves normally on the training distribution when we’re looking at it but then in deployment it goes and does evil things to satisfy its mesaobjective. Importantly, pursuing the mesaobjective hurts the base objective (since we’re only focusing on inner alignment here we can assume that the base objective is accurate), so deployment is the only time where it can do evil things without getting itself removed.

The main reason here that deception happens is that your base objective isn’t being integrated over the set of all possible states. In theory, if we could enumerate the entire (huge) input space, we could eliminate deceptive behavior entirely. We'd just simply test every single input and see how whether there are any regions where the model does a treacherous turn and consequently tanks the base objective (for now I'm trying to isolate inner alignment so I'm assuming we've solved outer alignment and have a value function that we can just query). If there is nowhere in the entire input source where the model does a treacherous turn, then congratulations, your model is, for all intents and purposes, indistinguishable from being not deceptively aligned. (Non-deceptive inner misalignment is very similar except the mesaoptimizer won’t be trying to hide from you.)

In practice, the input space is absolutely enormous, and it's completely and utterly infeasible to enumerate it. Still, we might think to approximate that "perfect" red teaming by doing a really good approximation with loads of input instances sampled from a broad distribution, possibly with the help of other NNs. Unfortunately, I think this approach is fundamentally doomed, no matter how good the red ranking distribution is made. For this to work, the entire distribution of possible inputs in the real world would have to be covered by the red teaming distribution, and any small imperfections are potential points of leverage for the mesaoptimizer. Aside from some contrived existence proofs like the RSA2048 example (which are admittedly very useful for thinking about these things so I use them a lot despite the contrivedness), there are simply many things that could pop up in the real world (and especially the future) that we simply cannot predict/simulate, and that would be a giveaway signal of being in the real world.

Given an imperfect red teaming distribution, penalties for discovered deceptive behavior incentivizes the mesaoptimizer to shift probability mass from those regions into regions of the input space that are not covered by the red teaming; essentially, the model could learn just as well learn to not be caught deceptively mesaoptimizing, rather than not to mesaoptimize. Two possible ways this could swing towards the former are if either a) the mesaoptimizer is able to convergently gradient hack to ensure that between the two equal-training-loss options the training process goes into the one that preserves the mesaoptimizer, or b) mesaobjectives that imply deceptive behavior are easier to learn than those that imply honest behaviors (this one is especially likely to happen if the base objective is weird and complicated, as human values probably are, since all the simpler, easy to learn approximations are misaligned). I think both of these have a pretty good chance of being true, and would result, in effect, in models learning to hide from oversight.

So, we can’t possibly cover the entire space of possible inputs that could be encountered in deployment, and if the mesaoptimizer retreats into one of the states we didn't think of then we’re done for, and it has a pretty big incentive to do so. This means that black box approaches, where we try various inputs while assuming no knowledge about the model itself, are kind of hopeless.

So actually although we care about the model’s behavior in situations we might encounter in the future, we can’t specify that yet in our training loss. So, if we optimize for this loss function, if it turns out that deception is favored by the prior defined by our learning process (which seems plausible), then even if we reach an optimum of this training loss our thing still might be deceptive outside of the training distribution. The other way things could go wrong is if our optimizer is not good enough. Then, we might just settle with a suboptimal deceptive model because our optimizer is unable to find the global min. This connects to the frame of inner alignment as a problem of imperfect optimization. In practice we get a combination of both of these problems at once: we can't find the optimum of the training loss defined by our training distribution, which would allow for deception, and even if we could it might still be deceptive.

Unforeseen consequences framework of deception

(Update: I no longer think of the failure introduced in this section as purely an inner alignment failure, but rather as a more general problem with training consequentialist systems.)

However, I think there’s a completely different frame from the whole out of distribution / “it only becomes evil in deployment” frame that I’ve been thinking a lot about. So one thing we might think is that maybe we can just do whatever oversight we did during training but during deployment as well, and it might have some competitiveness problems, but it seems like it would at least be a good starting point. Unfortunately, I think there are some subtle problems with this that totally break it and analyzing those problems brings us to the other framing of deception. I think these problems show up in the previous frame too.

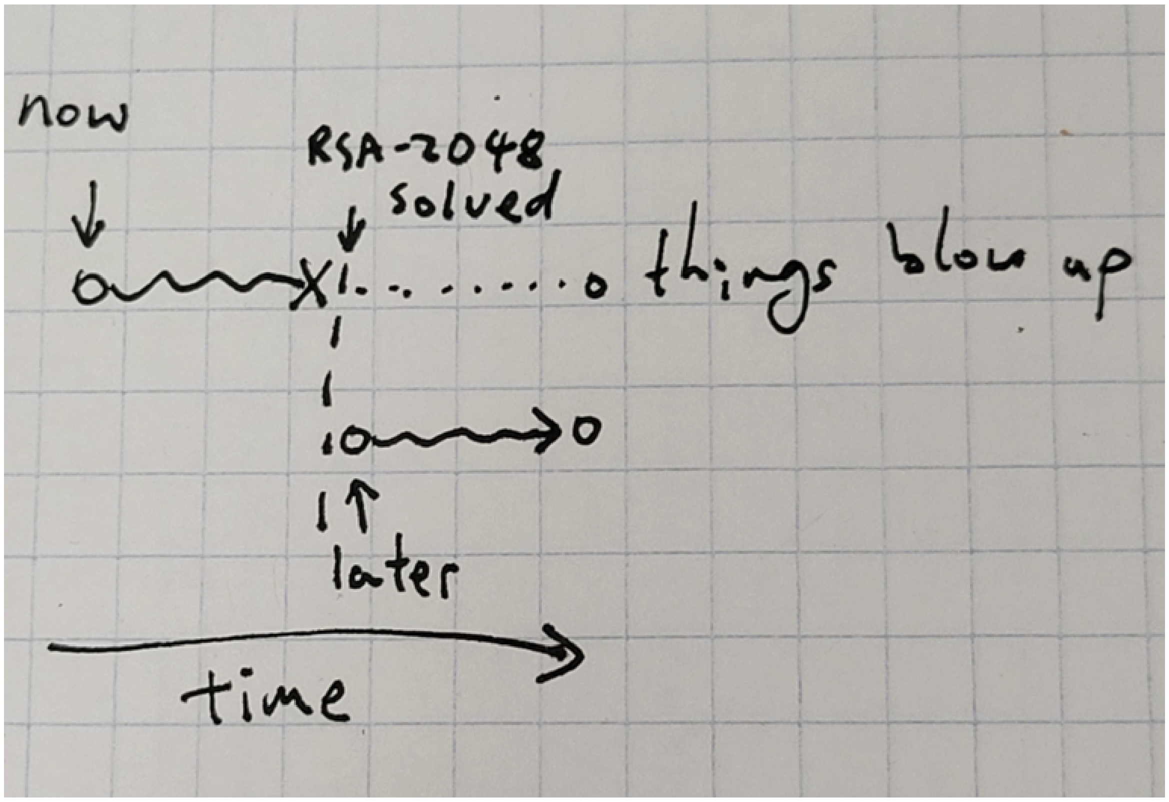
So let’s say the model proposes some action, and we want to know whether the action would be safe to implement. Then, assuming we have some world model, we can simulate the model’s action out, and see whether something bad happens (I’m assuming for now that we know something is bad when we see it, because we’re focusing on inner alignment. This is definitely not trivial or remotely solved and is a big part of outer alignment.) This is also roughly what we’d do for the training procedure in the previous frame. The problem is that this world model is completely impossible. Aside from the fact that it’s smaller than the world and therefore can’t fit the world, it’s also susceptible to RSA2048-like problems, where there are certain states of the world where we know we can be in, but can’t actually produce without enormous cost. Another example would be simulating a future where more powerful computers exist; we can’t do that on current computers obviously. And of course, RSA2048 is again just a particularly sharp example of the class of things that could happen but that we can’t think of. In practice, there are many other distributional shifts, but RSA2048 is essentially the worst possible case of distributional shift, and I expect the solution to transfer pretty well.

This means that we can’t consider plans in which RSA2048 suddenly appears at some intermediate point. Note that this actually does still allow us to catch cases where the model merely does something bad the moment it sees RSA2048, because after we see RSA2048 in the real world during deployment, then we can actually just use that and plan through the consequences, and we’ll catch the treacherous turn there. So our failure case has to be subtler than that: the failures happen when there’s some action the agent can take that would make it *such that our deploy time world model simulation thing breaks*. Some examples:

- The model self modifies to remove the world model simulator. This looks like a really good action, because removing this safeguard makes the model a lot more efficient, and because we can’t simulate worlds where RSA2048 happens, we don’t see it explode.
- The model improves itself without removing the simulator but also in the process introduces some exploitable bugs (either intentionally or unintentionally).
- If we prevent it from self modifying: The model creates a copy of itself that doesn’t have the safeguard.
- If we also try to somehow prevent it from making new agents, there are a whole bunch of ways to goodhart whatever definition of agent we pick that could still put the world

into a state where RSA2048 is bad

In general I don't think we can make the model not take actions which would make it such that seeing RSA2048 would then cause bad things, though I admit to not having thought too much about it so I could be wrong.



The more general problem is that due to our world model being inaccurate, we can't predict the consequences of the model's actions accurately, and the model can exploit this to do things that are bad in ways we can't think of. One corollary is the problem of always rejecting actions which do something bad that you can't even understand - I also talked about this subproblem here. If you can do that, then you can reject everything that could potentially cause the world to enter into a state without the safeguards in ways you can't understand, and of the remaining ones you should be able to understand them.

This problem actually applies to on-training-distribution things too. In the OOD frame, we kind of assume that on the training distribution we can simply tell on the training distribution whether it is indeed what we want, but in fact we can't because to know whether we actually want it we would need to know the consequences of the action. In that sense, this frame is kind of orthogonal to the OOD one: OOD-only stuff is one problem, and the inability to see consequences is another orthogonal problem, but I think the latter is somewhat more fundamental because if we solve it then we can get (uncompetitive) OOD safety.

This also kind of ties into ELK: if we can solve ELK, then we can ask the model whether it thinks the action will lead to the safeguards being removed. The main problem here is that an action could still lead to the safeguards being removed even if the model doesn't think so. An interesting avenue here would be how often this would happen - I don't have much of a prior on even what order of magnitude problem this is. ELK might also simply not be solvable too, which makes this much more of a headache.

Future Directions: what do we do about actions that look good but are actually evil in ways that we can't detect/comprehend; does ELK prevent safeguard removal in practice?

UPDATE: I made a followup post that dives into a bit more detail into the imperfect world modelling problem.

[ASoT] Some thoughts about LM monologue limitations and ELK

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

Editor's note: I'm experimenting with having a lower quality threshold for just posting things even while I'm still confused and unconfident about my conclusions, but with this disclaimer at the top. Thanks to Kyle and Laria for discussion.

One potential way we might think to interpret LMs is to have them explain their thinking as a monologue or justification or train of thought or something. Particularly, by putting the explanation before the answer, we might hope to encourage the model to actually use the monologue to come to its conclusion and try to avoid the model coming up with the bottom line first. However, there are a bunch of ways this could go wrong. For instance:

- Humans are really good at coming up with the bottom line first and then rationalizing a reasonable sounding process to arrive at that conclusion, so imitating humans encourages rationalization by default, even if the explanation is put temporally before the bottom line. Even though the model has uncertainty over the explanation, and in some sense the explanation determines the answer, the problem is that the distribution over the explanations may not be determined by the validity of the explanation, but rather be an integral over possible bottom lines.
- It's not sufficient for the answer to merely depend on the explanation: the model could be storing arbitrary data in the explanation in a way that doesn't line up with the meaning of the words. It could also be using the additional tokens to do more thinking in the latents but not in the tokens, etc.
- If the model is deceptive and we use the monologues to try and tell if the model is deceptive and apply any optimization pressure against that whatsoever (this includes rejecting actions/models based on it), then it would actually be incentivized to not tell us about its scheming, because we would be punishing (blatant monologue, evil action) more than (benign monologue, evil action).
- This also has a reduction to ELK: if the model's abstractions are sufficiently different from human abstractions, it may actually be more work for the model to translate its actual thought process to human understandable text than to just simulate the rationalization process a human would go through. To make the connection to ELK more clear, we can think of the reporter as the circuit that converts the model's actual thought process to text (or simulates the human thought process).

I think it's possible that we're simply lucky and that GPTs of the relevant size just don't have problems (in the same way that ELK could just be easy), but I'm not sure we want to bet on that. I also think lots of these problems correspond to problems with mechanistic interpretability, especially the ELK and deception related ones.

I don't think this is a particularly novel observation but it's nice having a reference to point to anyways.

UPDATE: After further discussion with Kyle, I've now been convinced that "generalizes human concepts correctly" (which is closer to the natural abstractions of alignment by default) and "has a compact direct translator" are subtly different, and that crucially it's possible for the former to be true while the latter is false (i.e you could have a model whose latents are extremely obfuscated and therefore expensive to translate to the human ontology, yet this model correctly generalizes human concepts).

[ASoT] Some thoughts about imperfect world modeling

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

So a few posts ago I looked at the problem of not being able to anticipate all consequences of an action as being related to deceptive mesaoptimization, but also outer alignment too. This post digs more into some of the things I only touched on briefly in that post.

Editor's note: I'm experimenting with having a lower quality threshold for just posting things even while I'm still confused and unconfident about my conclusions, but with this disclaimer at the top. Thanks to Kyle and AI_WAIFU for discussions.

Last time, we narrowed down the problem to [certain actions we can take now] leading to [a world such that if [RSA2048 or some other thing we can't simulate] happens, then [bad things will happen]] (brackets added for easier parsing of this sentence). This is because we can't plan for a future where RSA-2048 is solved but if we can already see RSA2048 in the world, then we can plan forward from that and see when things blow up.

So if we could somehow figure out how to ensure that no action puts us into an RSA2048-vulnerable world, then we can prevent this failure case. Unfortunately, knowing precisely whether a world is vulnerable would essentially require us to simulate RSA2048, which gets us nowhere. However, one observation is we don't have to know precisely, we just need to know whether a world *could* be vulnerable — like for example if the agent removes its own safeguard, then we wouldn't want to allow that even if the model is in fact safe and wouldn't destroy the world when it sees RSA2048.

Telling whether side effects are dangerous is hard

One thing we might think to do is instead simulate putting a dummy AI in the "do rollouts to find out if these actions destroy the world" box (the word "box" here has no relation to the AI box experiment), such that this dummy AI tries to kill everyone when it sees something we can actually simulate, like the string "chariots chariots" or something. Then, we can execute the action in the simulation, and then simulate seeing that string, and see if the box successfully prevented the action from being executed in the real world. This way, we can test whether the box is broken by the action, right?

Unfortunately there's a huge counterexample to this. So suppose the AI proposes building a really complicated machine in the world that we don't understand. We can still simulate the consequences of having this machine exist, of course, but we can't simulate it past the RSA2048 barrier, and so this machine could do evil stuff after seeing RSA2048 and we'd never know. The more general lesson from this is that if the model proposes doing something with side effects in the real world that we can't understand, that allows it to build up stuff "outside of the box" and then that stuff could do bad things. If we try to understand everything the model does, we end up bottlenecked by the whole "doing science" thing again, and if we try to bring the things the model does back inside the box (i.e only allow the machine it builds to also propose actions within the box), then you end up never being able to modify the real world at all because any changes to the real world could actually be setting up something that blows up after RSA2048, which is totally useless.

So, in general, it seems hard to know whether an action sets up something evil in the world that only activates after RSA2048.

It’s (partial) ELK reduction time, my dudes

There is a partial ELK reduction here in that if we can solve ELK then we can ask the model whether it thinks its actions are putting the world into a vulnerable state. It's worth noting that this *wouldn't* work if we just asked the model whether RSA2048 would cause it to kill everyone, because a) the model could always just remove our outer loop with this whole ELK thing before ever seeing RSA2048, and trying to prevent that collapses to just asking it whether it puts the world into a vulnerable state, and b) this actually depends on a slightly stronger version of ELK, where not only do we generalize to action sequences we can't see, but also to questions we can't see (because we can never simulate question answer pairs of the form "if you see RSA2048, then do you do x?", so the prior has to do all the work here).

~~However, this isn’t necessarily enough to prevent putting the world into a vulnerable state. For instance, suppose that it’s actually very easy to stumble on actions that put the world into a vulnerable state in ways humans can’t understand. Then, even if you just have a weak optimizer (say, just best-of-n based on some value model) that isn’t really maliciously “trying” to create a vulnerable state, and/or doesn't even realize that its actions will lead to a vulnerable state, you still have a high chance of ending up in one anyways.~~ *(Update: I no longer endorse this, because to be able to do best-of-n, the value model would need to be able to model action consequences, which brings us back to ELK. In general, it seems difficult/impossible to reliably take actions which optimize for RSA2048-vulnerability without having some model of whether actions lead to RSA2048-vulnerability. Possible forthcoming post on this)*

Stumbling into RSA2048-vulnerable states

One important thing here is that our abstraction where we pretend that RSA2048 is the only thing we can't model starts leaking here, because the more inaccurate our world model is, the easier it is to stumble into a vulnerable state, because there will simply be a lot more vulnerable states. So my intuition is that even though in these examples RSA2048-triggered optimization is a pretty unnatural thing to accidentally stumble upon, in practice this weak optimizer case matters a lot, because lots of things we might not be able to anticipate could trigger some optimizer we build in the world.

One reason we might expect it to be easy to stumble into these states is that lots of these states correspond to your AI building another AI to help it, which seems like a pretty convergently instrumental thing to do, and maybe your AI just fails at making the second AI inner aligned (but since the inner alignment failures happen behind the RSA2048 barrier, your world model rollouts think that this second AI is inner aligned!)

Another angle to look at the same thing is that this weak optimizer case is the case where even if the model isn’t “trying very hard” to get around your box, by default actions that are successful tend to be the ones that violate the integrity of the box. This framing makes weak optimizer failures seem pretty reasonable - the box is a pretty big constraint on the model, and it could get a lot more done a lot more efficiently without that constraint.

What do we do?

Unfortunately I don't know if there's a solid solution to this problem. Our biggest advantage here is that the optimizer is restricted and so can't put the full weight of its optimization behind optimizing against us here, though even then you can still apply quite a bit of pressure. We might be able to put in this constraint with some kind of hack like a learned reward model to detect the creation of additional agents, which is obviously bottlenecked on science-doing, but because the optimization pressure isn't that strong it *might* be fine, and would probably allow us to stretch further than we could with just doing science on the original problem directly. And of course this all depends on ELK; if we can't solve that then this becomes even more of a problem.

I also feel like the idea of weak optimization behaving in ways that mislead us even if there’s no “intent” in some sense in the system is something that could be useful to think about more generally, not just in the context of mesaoptimization/world modeling. I think this is an important crux and I'm still confused about it.

[ASoT] Consequentialist models as a superset of mesaoptimizers

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

TL;DR: I split out mesaoptimizers (models which do explicit search internally) from the superset of consequentialist models (which accomplish goals in the world, and may or may not use search internally). This resolves a bit of confusion I had about mesaoptimizers and whether things like GPT simulating an agent counted as mesaoptimization or not.

Editor's note: I'm experimenting with having a lower quality threshold for just posting things even while I'm still confused and unconfident about my conclusions, but with this disclaimer at the top. Thanks to Vivek Hebbar, Ethan Perez, Owain Evans, and Evan Hubinger for discussions.

UPDATE: The idea in this post is basically the same as the idea of "mesa-controllers" in this post .

What do I mean by consequentialist models?

By consequentialist models I mean models which optimize the world in the Alex Flint sense of optimization; i.e narrowing world states to some goal in a way robust to some perturbations. In other words, it's able to achieve consequences. The model doesn't have to be fully consequentialist either, it just has to have some kernel of consequentialist structure by virtue of actually achieving its goals sometimes. Of course, the degree to which a model is consequentialist is more of a scalar quantity than a discrete yes/no thing; for my purposes it really doesn't matter where to draw a dotted line that dictates when a model "becomes consequentialist".

(For what it's worth I would totally have called "consequentialist models" just mesaoptimizers and what other people call mesaoptimizers as like "searching mesaoptimizers" or something, but that would only create even more confusion than already exists)

For instance, the policy network of alphago alone (i.e none of the MCTS) is consequentialist, because it consistently steers the world into states where it's winning, despite my attempts to make it not win. Basically *any* RL policy is a fairly consequentialist model by this definition, since they channel the set of all states to the set of states that achieve high reward. I think of mesaoptimizers as a subset of consequentialist models, in that they are consequentialist, but they implement this consequentialism using explicit search rather than some other random thing. Explicit search is a kind of optimization, but not all optimization has to be search; you could have symbol manipulation, clever heuristics, gradient based methods, or at the most extreme even just a big lookup table preloaded with the optimal answer for each possible situation.

Why do we care about consequentialist models?

Consequentialist models are scary because when they are learned imperfectly resulting in a goal that is misaligned with ours, they competently pursue the wrong thing, rather than failing outright (other terms for this phenomenon coined by various people: objective misgeneralization, malign failure). This is often stated as a danger of mesaoptimization, but I think this property is slightly more general than that. It's not the search part that makes this dangerous, it's the consequentialism part. Search is just one way that consequentialism can be implemented.

Another way of thinking about this: instrumental convergence (and as a result, deception) is not exactly a result of search, or even utility maximization. Those things are ways of implementing consequentialism, but other ways of implementing consequentialism would result in the exact same problems. These non-searching consequentialist models might function internally as a pile of heuristics and shallow patterns that are just extremely competent at steering world states. Of course this is also a continuous thing, where you can do varying ratios of search to heuristics. My intuition is that humans, for instance, do very little search (our System 2), and rely on a huge pile of heuristics (our System 1) to provide a very powerful “rollout policy” to make the most of the little search we do.

(My intuition is that GPT, for instance, does little to no search, and has a huge pile of heuristics—but heuristics is likely all you need)

How do non searching models become deceptive?

One thing is that it’s slightly more complicated how non-search consequentialist models would discover deceptive strategies. Unlike search, where you can think up plans you never trained on and then filter for high reward, if you can’t do search, you have to somehow be able to tell that deceptive actions are high reward. However, I don’t think it’s impossible for non-searching models to still be deceptive. Some ways this could happen include:

- Learned from training data: The training data contained examples of deception that allowed the model to know about deceptive strategies, and it transfers this knowledge (i.e GPT can simulate agents with deceptive behavior)
- Deception might lase: I think it's likely that deception is natural even for non-search strategies, and *not* being deceptive requires additional work, so the heuristics that help models be consequentialist naturally generalize to deceptive behavior more than they do to corrigible behavior.

Differences between mesaoptimizers and consequentialist models

There are still some very key differences between mesaoptimizers and consequentialist models. For instance:

- Complexity: The fact that search is less complex than a pile of heuristics is extremely important when thinking about what the complexity prior incentivizes, so actually in these settings only mesaoptimizers really matter and we don't really care about some conseuentialist model that behaves largely the same but is implemented in a much more complex way.
- OOD behavior: Maybe consequentialist models have capabilities and objective robustness fail simultaneously off the training distribution more often than explicit search models, which continue to be capabilites robust and optimize competently for the wrong thing. This of course depends a lot on the exact nature of the heuristics; some heuristics might be capabilities-robust off distribution. I think it's plausible for some heuristic based systems to retain a lot of capabilities off distribution.

Aside: things that are optimized vs things that are optimizers

Another possible framing is that both consequentialist models and mesaoptimizers are *optimized* but consequentialist models don’t do any optimization themselves, whereas mesaoptimizers do. However, I don’t think this is quite true; just because you aren’t doing any explicit search doesn’t mean you can’t channel world states towards some those that satisfy some mesaobjective.

In the thermostat analogy: sure, it’s true that we put optimization into making the thermostat, and that the thermostat is kind of dumb and basically “just” executes the simple heuristic that we put into it, but it is still effective at steering the world into the state where the room is a particular temperature in a way that’s robust to other sources of temperature changes. My claim is essentially that there is not really any hard boundary between things that are optimized to be “just piles of heuristics/simple routines” (aka Adaptation-Executers) and things that are optimized to “actually do reasoning”. I fully expect that there can exist large enough piles of heuristics that can actually create sufficiently-powerful-to-be-dangerous plans.

Humans Reflecting on HRH

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

TL;DR: HRH places a theoretical bound on the best reflection process achievable by us. Notably, this is not necessarily guaranteed to converge to human values, nor is it something that is actually implementable in practice (analogously to HCH). In particular, this is intended to argue against claims that it is insufficient to model CEV as the output of some kind of reflection process. One can also think of HRH as an operationalization of the idea of a long reflection.

Thanks to Connor Leahy, Tasmin Leake, and AI_WAIFU for discussion.

One thing we might want to do is to define our CEV (i.e the ultimate thing we actually want our AGI to optimize for the rest of time) as the output of some long-running deliberation process (a long reflection). This would be extremely convenient; one could imagine having an AGI that lets the reflection process run untampered with, and then implements whatever it decides on. However, one might worry that this could be impossible -- perhaps there are kinds of moral progress that can't be captured in the frame of a reflection process, that require some kind of more sophisticated formalism to capture.

However, consider the reflection process defined as follows: it takes in information from the real world and the utility function output from the previous iteration of reflection, and has a human deliberate for a while and then outputs the improved utility function and crucially also an improved reflection process to be used in the next iteration (you can also tack on the ability for it to interact with the world, which enables it to run experiments, consult a computer, build more powerful aligned AGIs to help, etc). Let's call this Humans Reflecting on HRH. This process essentially covers any process we can use to come up with better theories of what our CEV is.

(If it bothers you that it "modifies" itself, you can think of it as a fixed function that takes in an initial program and optional additional inputs, and outputs a new program, and the function just evals the program internally at every step. The higher level algorithm of "have each step determine the algorithm used in the next step" remains constant, and that's the thing I refer to.)

I claim that HRH is the *best achievable* reflection process up to constant factors. Suppose you could come up with a better reflective process. Then the version of you in HRH would come up with that process too, and replace the next iteration of HRH with that process. A similar argument applies to the choice of who to put in the reflection process; if you can think of a better person to put in the process, then you could have thought of that within HRH and updated the reflection process to use that person instead. A similar argument also applies to how you ensure that the initial conditions of the reflective process are set up in the best way possible, or to ensure that the reflective process is robust to noise, etc, etc.

This construction may feel like cheating, but it exploits the core property that whatever reflection process we come up with, we are using our current reflective process to come up with it.

I expect some people to look at HRH and say "of course it would be aligned the hard part is that we literally can't implement that", and others to find it woefully inadequate and mutter "have you ever *met* a human?" Unfortunately, there is a fundamental limitation on what CEVs we can come up with, due to the fact that we bootstrap from humans. There might exist "better" CEVs that we could never think of, even with all the assistance and recursively self improved reflection processes we can construct for ourselves.

Some remaining difficulties with HRH that I haven't figured out yet:

- Can we extract *intermediate* outputs from HRH? Possibly using logical inductors to get credences over the final output? Can we somehow put some kind of convergence guarantee on these intermediates?
- How the hell do you actually implement anything remotely like HRH (with or without the logical inductors) in the real world?
- How do we firm down the formalism for cases where the reflection process interacts with the real world? Real world interactions potentially breaks things by making the reflection process not a pure function.

Towards deconfusing wireheading and reward maximization

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

TL;DR: A response to “Reward is not the optimization target,” and to a lesser extent some other shard theory claims. I agree with the headline claim but I disagree with some of the implications drawn in the post, especially about wireheading and the "executing behaviors downstream of past reinforcement" framing.

My main claims:

- Not only does RL not, by default, produce *policies* which have reward maximization as their behavioral objective, but in fact I argue that it is *not possible* for RL policies to care about “reward” in an embedded setting.
- I argue that this does *not* imply that wireheading in RL *agents* is impossible, because wireheading does not mean “the policy has reward as its objective”. It is still possible for an RL *agent* to wirehead, and in fact, is a high probability outcome under certain circumstances.
- This bears a clean analogy to the human case, viewing humans as RL agents; there is no special mechanism going on in humans that is needed to explain why humans care about things in the world.

A few notes on terminology, which may or may not be a bit idiosyncratic:

- I define an RL *policy* to refer to a function that takes states and outputs a distribution over next actions.
- I define an RL *agent* to be an RL policy combined with some RL algorithm (i.e PPO, Q-learning) that updates the policy on the fly as new trajectories are taken (i.e I mostly consider the online setting here).
- The objective that any given *policy* appears to optimize is its behavioral objective (same definition as in Risks from Learned Optimization).
- The objective that the *agent* optimizes is the expected value the policy achieves on the reward function (which takes observations and outputs reals).
- A utility function takes a universe description and outputs reals (i.e it cares about "real things in the outside world", as opposed to just observations/reward, to the extent that those even make sense in an embedded setting).
- I mostly elide over the mesaoptimizer/mesacontroller distinction in this post, as I believe it’s mostly orthogonal.

Thanks to AI_WAIFU and Alex Turner for discussion.

Outside world objectives are the policy’s optimization target

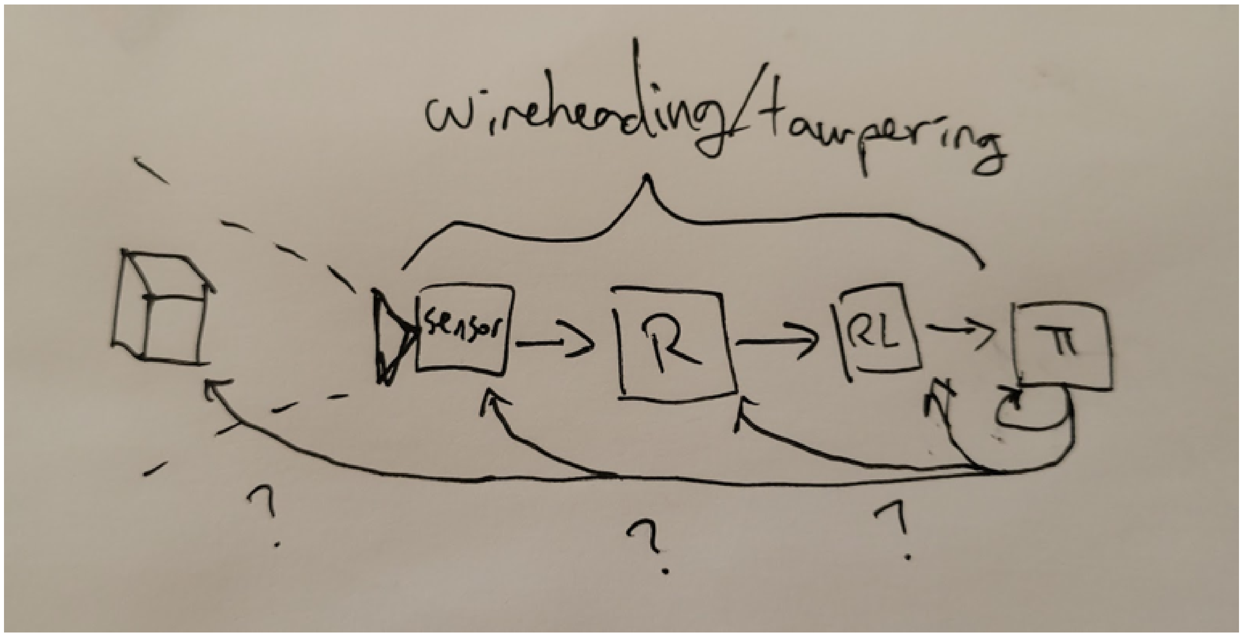
First, let us think about the mechanics of an RL agent. Each possible policy in policy-space implements a behavioral objective, and as a result has some behavior which may or may not result in trajectories which receive high reward. The RL algorithm performs optimization over the space of possible policies, to find ones that would have received high reward (the exact mechanism depends on the details of the algorithm). This optimization may or may not be local. The resulting policies do not necessarily “care” about “reward” in any sense, but they will have been selected to achieve high reward.

Now, consider the same RL agent in an embedded setting (i.e the agent runs on a computer that is part of the environment that the agent acts in). Then, because the sensors, reward function, RL algorithm, etc are all implemented within the world, there exist *possible policies* that execute the strategies that result in i.e the reward function being bypassed and the output register being set to a large number, or the sensors being tampered with so everything looks good. This is the typical example of wireheading. Whether the RL *algorithm* successfully finds the policies that result in this behavior, it is the case that the *global optima* of the reward function on the set of all policies consists of these kinds of policies. Thus, for sufficiently powerful RL *algorithms* (not *policies*!), we should expect them to tend to choose policies which implement wireheading.

There are in fact many distinct possible policies with different behavioral objectives for the RL algorithm to select for: there is a policy that changes the world in the “intended” way so that the reward function reports a high value, or one that changes the reward function such that it now implements a different algorithm that returns higher values, or one that changes the register the output from the reward function is stored in to a higher number, or one that causes a specific transistor in the processor to misfire, etc. All of these policies optimize some thing in the outside world (a utility function); for instance, the utility function that assigns high utility to a particular register being a large number. The value of the particular register is a fact of the world. There even exists a policy that cares about that particular register at that particular physical location in the world even if it is not connected to the RL algorithm at all, but that policy does not get selected for by the RL algorithm.

However, when we try to construct an RL policy that has as its behavioral objective the “reward”, we encounter the problem that it is unclear what it would mean for the RL policy to “care about” reward, because there is no well defined reward channel in the embedded setting. We may observe that all of the above strategies are instrumental to having the particular policy be picked by the RL algorithm as the next policy used by the agent, but this is a utility over the world as well (“have the next policy implemented be this one”), and in fact this isn’t really much of a reward maximizer at all, because it explicitly bypasses reward as a concept altogether! In general, in an embedded setting, any preference the policy has over “reward" (or "observations") can be mapped onto a preference over facts of the world.

Thus, whether the RL agent wireheads is a function of how powerful the RL algorithm is, how easy wireheading is for a policy to implement, and the inductive biases of the RL algorithm. Depending on these factors, the RL algorithm might favor the policies with mesaobjectives that care about the “right parts” of the world (i.e the thing we actually care about), or the wrong parts (i.e the reward register, the transistors in the CPU).



Humans as RL agents

We can view humans as being RL agents, and in particular consisting of the human reward circuitry (RL algorithm) that optimizes the rest of the brain (RL policy/mesaoptimizer) for reward.

This resolves the question of why humans don't want to wirehead—because you identify with the rest of the brain and so "your" desires are the mesaobjectives. "You" (your neocortex) know that sticking electrodes in your brain will cause reward to be maximized, but your reward circuitry is comparatively pretty dumb and so doesn't realize this is an option until it actually gets the electrodes (at which point it does indeed rewire the rest of your brain to want to keep the electrodes in). You typically don't give into your reward circuitry because your RL algorithm is pretty dumb and your policy is more powerful and able to outsmart the RL algorithm by putting rewards out of reach. However, this doesn't mean your policy always wins against the RL algorithm! Addiction is an example of what happens when your policy fails to model the consequences of doing something, and then your reward circuitry kicks into gear and modifies the objective of the rest of your brain to like doing the addictive thing more.

In particular, one consequence of this is we also don't need to postulate the existence of some kind of special as yet unknown algorithm that only exists in humans to be able to explain why humans end up caring about things in the world. Whether humans wirehead is determined by the same thing that determines whether RL agents wirehead.