Pointing at Normativity

1. Learning Normativity: A Research Agenda
2. Normativity
3. Recursive Quantilizers II
4. The Pointers Problem: Clarifications/Variations
5. Four Motivations for Learning Normativity

Learning Normativity: A Research Agenda

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

(Related to Inaccessible Information , Learning the Prior , and Better Priors as a Safety Problem . Builds on several of my alternate alignment ideas .)

I want to talk about something which I'll call *learning normativity*. What is normativity? Normativity is correct behavior. I mean something related to the fuzzy concept humans convey with the word "should". I think it has several interesting features:

- Norms are the result of a complex negotiation between humans, so they shouldn't necessarily be thought of as the result of maximizing some set of values. This distinguishes learning normativity from value learning.
- A lot of information about norms is present in the empirical distribution of what people actually do, but you can't learn norms just by learning human behavior. This distinguishes it from imitation learning.
- It's often possible to provide a lot of information in the form of "good/bad" feedback. This feedback should be interpreted more like approval-directed learning rather than RL. However, approval should not be treated as a gold standard.
- Similarly, it's often possible to provide a lot of information in the form of rules, but rules are not necessarily 100% true; they are just very likely to apply in typical cases.
- In general, it's possible to get very rich *types* of feedback, but very *sparse* : humans get all sorts of feedback, including not only instruction on how to act, but also *how to think.*
- Any one piece of feedback is suspect. Teachers can make mistakes, instructions can be wrong, demonstrations can be imperfect, dictionaries can contain spelling errors, reward signals can be corrupt, and so on.

Example: Language Learning

A major motivating example for me is how language learning works in humans. There is clearly, to some degree, a "right way" and a "wrong way" to use a language. I'll call this *correct usage*.

One notable feature of language learning is that we don't always speak, or write, in correct usage. This means that a child learning language has to distinguish between mistakes (such as typos) and correct usage. (Humans do sometimes learn to imitate mistakes, but *we have a notion of not doing so*. This is unlike GPT systems learning to imitate the empirical distribution of human text.)

This means we're largely doing something like unsupervised learning, but with a notion of "correct"/"incorrect" data. We're doing something like throwing data out when it's likely to be incorrect.

A related point is that we are better at *recognizing* correct usage than we are at *generating* it. If we say something wrong, we're likely able to correct it. In some sense, this means there's a foothold for intelligence amplification: we know how to generate our own training gradient.

Another fascinating feature of language is that although native speakers are pretty good at both recognizing and generating correct usage, *we don't know the rules explicitly*. The whole field of linguistics is largely about trying to uncover the rules of grammar.

So it's impossible for us to teach proper English by teaching the rules. Yet, we do know *some* of the rules. Or, more accurately, we know a set of rules that *usually* apply. And those rules are *somewhat* useful for teaching English. (Although children have usually reached fluency before the point where they're taught explicit English grammar.)

All of these things point toward what I mean by *learning normativity:*

- We can tell a lot about what's normative by simply observing what's common, but the two are not exactly the same thing.
- A (qualified) human can usually label an example as correct or incorrect, but this is not perfect either.
- We can articulate a lot about correct vs incorrect in the form of rules; but the rules which we can articulate never seem to cover 100% of the cases. A linguist is a lot like a philosopher: taking a concept which is understood at an intuitive level (which a great many people can fluently apply in the correct manner), but struggling for years to arrive at a correct technical definition which fits the intuitive usage.

In other words, the overriding feature of normativity which I'm trying to point at is that nothing is ever 100%. Correct grammar is not defined by any (known) rules or set of text, nor is it (quite) just whatever humans judge it is. All of those things give a lot of information about it, but it could differ from each of them. Yet, on top of all that, basically everyone learns it successfully. This is very close to Paul's Inaccessible Information: information for which we cannot concoct a gold-standard training signal, but which intelligent systems may learn anyway.

Another important feature of this type of learning: *there is a fairly clear notion of superhuman performance.* Even though human imitation is *most* of the challenge, we could declare something superhuman based on our human understanding of the task. For example, GPT is trained exclusively to imitate, so it should never exceed human performance. Yet, we could tell if a GPT-like system *did* exceed human performance:

- Its spelling and grammar would be immaculate, rather than including humanlike errors;
- its output would be more creative and exciting to read than that of human authors;
- when good reasoning was called for in a text, its arguments would be clear, correct, and compelling;
- when truth was called for, rather than fiction, its general knowledge would be broader and more accurate than a human's.

It seems very possible to learn to be better than your teachers in these ways, because humans sometimes manage to do it.

Learning in the Absence of a Gold Standard

In statistics and machine learning, a "gold standard" is a proxy which we treat as good enough to serve as ground truth for our limited purposes. The accuracy of any other estimate will be judged by comparison to the gold standard. This is similar to the concept of "operationalization" in science.

It's worth pointing out that in pure Bayesian terms, there is nothing especially concerning about learning in the absence of a gold standard. I have data X. I want to know about Y. I update on X, getting P(Y|X). No problem!

However, that only works if we have the right prior. We could try to learn the prior from humans, which gets us 99% of the way there... but as I've mentioned earlier, human imitation does not get us all the way. Humans don't perfectly endorse their own reactions.

(Note that whether "99% of the way" is good enough for AI safety is a separate question. I'm trying to define the Big Hairy Audacious Goal of learning normativity.)

Actually, I want to split "no gold standard" into two separate problems.

1. **There's no type of feedback which we can perfectly trust.** If humans label examples of good/bad behavior, a few of those labels are going to be wrong. If humans provide example inferences for learning the prior, some of those example inferences are (in a very real sense) wrong. And so on.
2. **There's no level at which we can perfectly define the loss function.** This is a consequence of no-perfect-feedback, but it's worth pointing out separately.

No Perfect Feedback

I think I've made the concept of no-perfect-feedback clear enough already. But what could it mean to *learn* under this condition, in a machine-learning sense?

There are some ideas that get part of the way:

- Jeffrey updates let us update to a *specific probability* of a given piece of feedback being true, rather than updating to 100%. This allows us to, EG, label an image as 90%-probable cat, 9%-probable dog, 1% broad distribution over other things.
    - This allows us to give some evidence, while allowing the learner to decide later that what we said was wrong (due to the accumulation of contrary evidence).
    - This seems helpful, but we need to be *confident that those probability assignments are themselves normatively correct*, and this seems like it's going to be a pretty big problem in practice.
- Virtual evidence is one step better: we don't have to indicate what actual probability to update to, but instead only indicate the *strength* of evidence.
    - Like Jeffrey updates, this means we can provide strong evidence while still allowing the system to decide later that we were wrong, due to the accumulation of contradicting evidence.
    - Unlike Jeffrey updates, we don't have to decide what probability we should update *to*, only the direction and strength of the evidence.
- Soft labels in machine learning provide a similar functionality. In EM learning, a system learns from its own soft labels. In LO-shot learning, a system leverages the fact that soft labels contain more information than hard labels, in order to learn classes with less than one examples per class.

However, although these ideas capture *weak feedback* in the sense of less-than-100%-confidence feedback, they don't capture the idea of ***reinterpretable feedback:***

- A system should ideally be able to learn that *specific **types** of feedback are erroneous,* such as corrupted-feedback cases in reinforcement learning. A system might learn that my feedback is lower quality right before lunch, for example.
- A system should be able to *preserve the overall meaning of a label despite an ontology shift*. For example, deciding that fruit/vegetable is not a useful taxonomic or culinary distinction should not destroy the information gained from such labels. Or, if human feedback includes formal English grammar, that information should not be totally discarded if the system realizes that the rules don't fully hold and the supposed grammatical categories are not as solid as claimed.
- Feedback should be associated with *a cloud of possible interpretations*. When humans say "weird", we often mean "unusual", but also sometimes mean "bad". When humans

say we don't understand, we often *really* mean we don't *endorse.* A system should, ideally, be able to *learn a mapping* from the feedback humans actually give to what they really mean. This is, in any case, the general solution to the previous bullet points.

But "learning a mapping from what feedback is given to what is meant" appears to imply that there is *no fixed loss function* for machine learning to work on, which would be a serious challenge. This is the subject of my point #2 from earlier:

No Perfect Loss Function

We can frame (some) approaches to the value specification problem in a sequence of increasingly sophisticated approaches (similar to the hierarchy I discussed in my "stable pointers to value" posts (1,2,3)):

1. *Direct specification of the value function.* This fails because we don't know what values to specify, and expect anything we can write down to be highly Goodhart-able.
2. *Learning human values.* We delegate the specification problem to the machine. But, this leaves us with the meta problem of specifying how to learn. Getting it wrong can lead to wireheading and human manipulation. Even in settings where this is impossible, we face Stuart's no-free-lunch results.
3. *Learning to learn human values.* Stuart suggests that we can get around the no-free-lunch results by loading the right prior information into the learner, in keeping with his more general belief that Bayesian reasoning is fine as long as it has the right prior information. But this seems to go back to the problem of learning the human prior. So we could apply a learning approach again here. But then we again have a specification problem for the loss function for *this* learning...
4. ...

You get the picture. We can keep pushing back the specification problem by learning, learning to learn, learning to learn to learn... Each time we push the problem back, we seem to gain *something*, but we're also stuck with a *new* specification problem at the meta level.

Could we specify a way to learn at *all the levels,* pushing the problem back infinitely? This might sound absurd, but I think there are ways to accomplish this. We need to somehow "collapse all the levels into one learner" -- otherwise, with an infinite number of levels to learn, there would be no hope. There needs to be very significant generalization across levels. For example, Occam's razor is a good starting rule of thumb at all levels (at least, all levels above the lowest). However, because Occam is not enough, it will need to be augmented with other information.

Recursive reward modeling is similar to the approach I'm sketching, in that it recursively breaks down the problem of specifying a loss function. However, it doesn't really take the same learning-to-learn approach, and it also doesn't aim for a monolithic learning system that is able to absorb information at all the levels.

I think of this as necessary learning-theoretic background work in order to achieve Stuart Armstrong's agenda, although Stuart may disagree. The goal here is to provide one framework in which all the information Stuart hopes to give a system can be properly integrated.

Note that *this is only an approach to* ***outer*** *alignment*. The inner alignment problem is a separate, and perhaps even more pressing, issue. The next section could be of more help to inner alignment, but I'm not sure this is overall the right path to solve that problem.

Process-Level Feedback

Sometimes we care about *how we get* the answers, not just what the answers are. That is to say, sometimes we can point out problems with methodology without being able to point to problems in the answers themselves. Answers can be suspect based on how they're computed.

Sometimes, points can only be effectively made in terms of this type of feedback. Wireheading and human manipulation can't be eliminated through object-level feedback, but we could point out examples of the wrong and right types of reasoning.

Process-level feedback blurs the distinction between inner alignment and outer alignment. A system which accepts process-level feedback is essentially exposing all its innards as "outer", so *if we can provide the appropriate feedback*, there should be no separate inner alignment problem. (Unfortunately, it must be admitted that it's quite difficult to provide the right feedback -- due to transparency issues, we can't expect to understand all models in order to give feedback on them.)

I also want to emphasize that we want to give feedback on the *entire* process. It's no good if we have "level 1" which is in charge of producing output, and learns from object-level feedback, but "level 2" is in charge of accepting process-level feedback about level 1, and adjusting level 1 accordingly. Then we still have a separate inner alignment problem for level 2.

This is the same kind of hierarchy problem we saw in "No Perfect Loss Function". Similarly, we want to collapse all the levels down. We want *one* level which is capable of accepting process-level feedback about *itself*.

Learning from Process-Level Feedback

In a Bayesian treatment, process-level feedback means *direct feedback about hypotheses*. In theory, there's no barrier to this type of feedback. A hypothesis can be ruled out by fiat just as easily as it can be ruled out by contradicting data.

However, this isn't a very powerful learning mechanism. If we imagine a human trying to inner-align a Bayesian system this way, the human has to *find and knock out every single malign hypothesis*. There's no generalization mechanism here.

Since detecting malign hypotheses is difficult, we want the learning system to help us out here. It should generalize from examples of malign hypotheses, and attempt to draw a broad boundary around malignancy. Allowing the system to judge itself in this way can of course lead to malign reinterpretations of user feedback, but hopefully allows for a basin of attraction in which benevolent generalizations can be learned.

For example, in Solomonoff induction, we have a powerful hierarchical prior in the distribution on program prefixes. A program prefix can represent any kind of distribution on hypotheses (since a program prefix can completely change the programming language to be used in the remainder of the program). So one would *hope* that knocking out hypotheses would reduce the probability of all other programs which share a prefix with that hypothesis, representing a generalization "this branch in my hierarchical prior on programs seems iffy". (As a stretch goal, we'd also like to update against other similar-looking branches; but we *at least* want to update against *this* one.)

However, no such update occurs. The branch loses mass, due to losing one member, but programs which share a prefix with the deleted program don't lose any mass. In fact, they gain mass, due to renormalization.

It seems we don't just want to update on "not this hypothesis"; we want to explicitly model some sort of malignancy judgement (or more generally, a quality-of-hypothesis judgement), so that we can update estimations of how to make such judgements. However, it's difficult to

see how to do so without creating a hierarchy, where we get a top level which isn't open to process-level feedback (and may therefore be malign).

Later, I'll present a Bayesian model which *does* have a version of generalization from feedback on hypotheses. But we should also be open to less-Bayesian solutions; it's possible this just isn't captured very well by Bayesian learning.

Prospects for Inner Alignment

I view this more as a preliminary step in one possible approach to inner alignment, rather than "a solution to inner alignment".

If (a) you want to learn a solution to inner alignment, rather than solving it ahead of time, and (b) you agree with the framing of process-level feedback / feedback on hypotheses, and (c) you agree that we can't rely on a trusted meta-level to take process-level feedback, but rather need to accept feedback on "the whole process", then I think it stands to reason that ***you need to specify what it means to learn*** in this setting. I view the preceding sections as an argument that there's a non-obvious problem here.

For example, Stuart Armstrong has repeatedly argued that Bayesian learners can overcome many safety problems, *if only they're given the right prior information*. To the extent that this is a claim about inner alignment (I'm not sure whether he would go that far), I'm claiming that *we need to solve the problem of giving process-level feedback to a Bayesian learner* before he can make good on his claim; otherwise, there's just no known mechanism to provide the system with all the necessary information.

Anyway, even if we accomplish this step, there are still several other obstacles in the way of this approach to inner alignment.

1. **Transparency:** It's unrealistic that humans can provide the needed process-level feedback without powerful transparency tools. The system needs to correctly generalize from simpler examples humans provide to the more difficult examples which a human can't understand. That will be difficult if humans can only label very very simple examples.
2. **Basin of Attraction:** Because the system could use malign interpretations of human feedback, it's very important that the system start out in a benign state, making trusted (if simplistic) generalizations of the feedback humans can provide.
3. **Running Untrusted Code:** A straightforward implementation of these ideas will still have to run untrusted hypotheses in order to evaluate them. Giving malign hypotheses really low probability doesn't help if we still run really low-probability hypotheses, and the malign hypotheses can find an exploit. This is similar to Vanessa's problem of non-Cartesian daemons.

Regardless of these issues, I think it's valuable to try to solve the part of the problem I've outlined in this essay, in the hope that the above issues can also be solved.

Summary of Desiderata

Here's a summary of all the concrete points I've made about what "learning normativity" should mean. Sub-points are not subgoals, but rather, additional related desiderata; EG, one might significantly address "no perfect feedback" without significantly addressing "uncertain feedback" or "interpretable feedback".

1. **No Perfect Feedback:** we want to be able to learn with the possibility that any one piece of data is corrupt.

    1. **Uncertain Feedback:** data can be given in an uncertain form, allowing 100% certain feedback to be given (if there ever is such a thing), but also allowing the system to learn significant things in the absence of any certainty.
    2. **Reinterpretable Feedback:** ideally, we want *rich hypotheses about the meaning of feedback,* which help the system to identify corrupt feedback, and interpret the information in imperfect feedback.
2. **No Perfect Loss Function:** we don't expect to perfectly define the utility function, or what it means to correctly learn the utility function, or what it means to learn to learn, and so on. At no level do we expect to be able to provide a single function we're happy to optimize.
    1. **Learning at All Levels:** Although we don't have perfect information at any level, we do get meaningful benefit with each level we step back and say "we're learning this level rather than keeping it fixed", because we can provide meaningful approximate loss functions at each level, and meaningful feedback for learning at each level. Therefore, we want to be able to do learning at each level.
    2. **Between-Level Sharing:** Because this implies an infinite hierarchy of levels to learn, we need to share a great deal of information between levels in order to learn meaningfully.
3. **Process Level Feedback:** we want to be able to give feedback about *how to arrive at answers*, not just the answers themselves.
    1. **Whole-Process Feedback:** we don't want some segregated meta-level which accepts/implements our process feedback about the rest of the system, but which is immune to process feedback itself. Any part of the system which is capable of adapting its behavior, we want to be able to give process-level feedback about.
    2. **Learned Generalization of Process Feedback:** we don't just want to promote or demote specific hypotheses. We want the system to learn from our feedback, making generalizations about which *kinds* of hypotheses are good or bad.

Initial Attempt: Recursive Quantilizers

I'll give an initial stab at solving these problems, as a proof-of-concept. (Otherwise I fear the above desiderata may look like they're simply impossible.)

This is a formalization of the recursive quantilizers idea which I described previously.

A quantilizer is a mild optimizer which avoids catastrophic outcomes with high probability, averting Goodhart's Law. It accomplishes this by refusing to 100% trust its value function. This seems like a good building block for us, since it significantly addresses "no perfect loss function."

A quantilizer requires a *value function,* V, which it mildly optimizes, and a *safe distribution,* S, which is a distribution over outputs which is assumed to have a low probability of catastrophic outcomes. It also requires an optimization parameter, p. The quantilizer

$Q(V, S, p)$ mildly optimizes by randomly taking the top $p\%$ of outputs from S, as ranked by V.

(p can be derived from an estimate of the probability of catastrophe in S, combined with a level of tolerance for catastrophic risk.)

In line with our *learning at all levels* desiderata, we want to learn V, S, and p. Moreover, we'll produce all of these via quantilization, and recursively learn the inputs to all of those quantilizers as well.

In order to get *between-level sharing,* we will actually learn one big quantilization process which is equivalent to all those infinitely many levels.

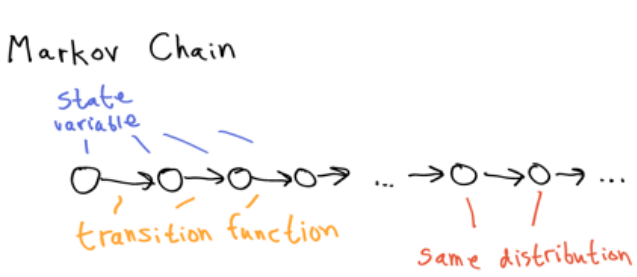
Here's the setup.

In order to unify all the levels, I'll have quantilizers select *unified type for all arguments (UTAAs):* a tuple (V, S, p, o) where o is the object-level output we want out of the whole system (EG plans, strings of text, whatever). So now we understand the quantilization function $Q(a, b, c)$ to take 3 UTAAs. It will take the value function from the first, the safe distribution from the second, and the optimization parameter from the third. It then produces a UTAA. The user can take the object-level output of that UTAA as the result.*

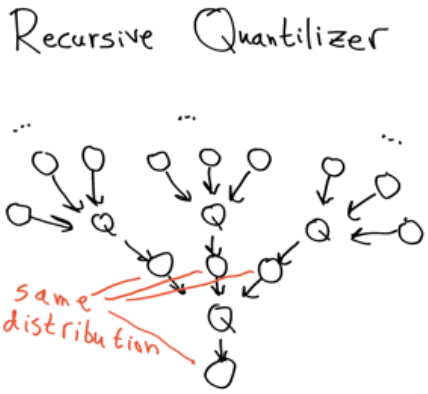
What we want is a notion of *stationary distribution on UTAAs.* This is a distribution $D \in \Delta UTAA$ such that if we sample 3 elements from D, and feed those three elements into Q, the distribution on outputs of Q is just D again.

In effect, a stationary distribution D is *all quantilized out*; we can quantilize again, but it won't make any difference. So, we can just sample from D and take the object-level output from the resulting UTAA.

This is very similar to the notion of stationary distribution for markov chains. The difference is that we have a *markov tree* instead of a chain.



If we run an ergodic markov chain long enough, we reach a stationary distribution: applying the transition function one more time doesn't make any difference



Similarly, we can imagine making a tree large enough that quantilizing one more time doesn't make any difference.

So, just as finding the stationary distribution of a markov-chain monte carlo algorithm is similar to running the algorithm long enough that running it any longer is pointless, finding a stationary distribution for recursive quantilizers is like building a learning-to-learn hierarchy that's large enough that we don't get anything by going any further.

That's all well and good, but how are we supposed to find a stationary distribution we like? We can't just take a fixed point and hope it's useful and benign; there'll be lots of crazy fixed points. How do we steer this thing toward desirable outcomes?

Parameterizing Stationary Distributions

If a markov chain has multiple stationary distributions, we can parameterize them through a distribution on starting states. A distribution on starting states just means a probability of picking any one starting element, so this relationship is completely linear: by interpolating between different starting elements, we interpolate between the stationary distributions which those starting elements eventually reach.

We can similarly parameterize stationary distributions via initial distributions. However, we don't get the same linearity. Because we have to select many starting elements for the 3^n inputs to an n-level tree, and we select those elements as independent draws from the initial distribution, we can get nonlinear effects. (This is just like flipping a biased coin (with sides labelled 1 and 0) twice and sending the two results through an XOR gate: the probability of getting a 1 out of the XOR is nonlinear in the bias.)

This means we can't reduce our uncertainty over initial distributions to uncertainty over initial UTAA. (There may be some other tricks we can use to simplify things, but they probably aren't worth exploring in this post.)

So we *can* parameterize our uncertainty over stationary distributions via uncertainty over initial distributions. But, this is just turning uncertainty over one kind of distribution into uncertainty over another. What's the benefit of this?

1. The set of stationary distributions is hard to know, but the set of possible initial distributions is clear. So this gives us an easy-to-work-with representation of stationary distributions.
2. We know every stationary distribution is in the set, since we can start out in a stationary distribution.
3. We can easily define the mapping from initial distributions to stationary distributions; it's just the stationary distribution you get by running things long enough, sampling from the given initial distribution. (Of course we may not get to any stationary distribution at all, but we can formally solve this by introducing a cutoff in program size, or through other devices.)
4. We can therefore define learning: an update against a UTAA produces an update against initial distributions which produce that UTAA.

This is, of course, a very computationally intensive procedure. Unless better algorithms are found, the only way we can update is by producing a large quantilization tree (which we hope has converged) and running it many times to evaluate the outcome of a given initial distribution.

However, the resulting system has many marvelous properties. If we want to give feedback at any level in the hierarchy, we can convert this into feedback about UTAAs, and update our prior over initial distributions accordingly. For example:

- We can label outputs as bad/incorrect by updating against all UTAAs which include those outputs.
- We can give evidence about the value function over outputs, and convert this to an update about UTAAs based on the value function they contain. So, we can do value learning.
- We can learn about the safe distribution over outputs. For example, one proposal for finding safe distributions is to model human behavior. Data-sets of human behavior

could induce updates over UTAAs by checking how well a UTAA's proposed safe distribution fits the data.
- At the same time, we can learn about the loss function by which we score safe distributions. If we have an update about this loss function, we translate it to an update about UTAAs by checking how a UTAA's value function *examines the safe distribution of another UTAA* when scoring it. Updating UTAAs based on this will, effectively, change the way safe distributions are selected in the second-to-last quantilization step. (Of course, it really changes *all* the quantilization steps, but when we anchor ourselves in how changes to the initial distribution alter our distribution on actual outputs, the easiest way to understand what's going on is to see this as a change to the second-to-last step.)
- Similarly, we can learn about the loss function by which we score loss functions. So in the same system, we can directly learn from feedback, we can do value learning, and we can do meta-value-learning where we learn how to interpret evidence in value-learning.
- Similarly, we can learn the safe distribution for meta-loss functions, the safe distribution over safe distributions, and on and on.
- We can also allow process-level feedback by enabling UTAA value functions to examine the source code of other UTAAs (e.g. looking at how those UTAAs compute their value functions and safe distributions). We can teach UTAAs to detect suspicious code in other UTAAs and rate those UTAAs very poorly.

Wouldn't it be fascinating to be able to provide all those types of learning in one system?

Analysis in terms of the Criteria

Let's examine how we did in terms of the criteria which I gave.

1. **No Perfect Feedback:** This wasn't addressed directly, but might be indirectly addressed via #2.
    1. **Uncertain Feedback:** I didn't specify any way to provide uncertain feedback, but it would be easy enough to do so.
    2. **Reinterpretable Feedback:** I think this is a big failing of the approach as it stands.
2. **No Perfect Loss Function:** Very significantly addressed by quantilization.
    1. **Learning at All Levels:** Very significantly addressed by the recursive quantilization setup.
    2. **Between-Level Sharing:** Significantly addressed. I didn't really talk about how this works, but I think it can work well in this setup.
3. **Process Level Feedback:** Significantly addressed. The process which creates a given output is essentially the big tree that we sample. We can give any kind of feedback about that tree that we want, including any computations which occur inside of the value functions or safe distributions or elsewhere.
    1. **Whole-Process Feedback:** Somewhat addressed. There is a question of whether the initial distribution constitutes a meta-level beyond the reach of process-level feedback.
    2. **Learned Generalization of Process Feedback:** Significantly addressed. Process-level feedback can be given directly, as evidence against a specific UTAA, in which case there will be some generalization as we update against anything which thought that UTAA was a good idea. Or it could be given more indirectly, as general (level-independent) information about how value functions should judge UTAA. In that case there may be more generalization, as we update on how to judge UTAAs generally. (Or maybe not? An equivalence theorem about these different types of feedback would be nice.)

I think the most significant problem here is the lack of reinterpretable feedback. When we give feedback about something, we have to figure out how to translate it into an update

about UTAAs (which can then be translated into an update about initial distributions). This update is fixed forever. This means the updates we make to the system aren't really tied to the value functions which get learned. So, for example, learning better value-learning behavior doesn't directly change how the system responds to updates we give it about the value function. (Instead, it may change how it interprets some other set of data we give it access to, as input to UTAAs.) This makes the "learning-to-learn" aspect of the system somewhat limited/shallow.

The second most significant concern here is whether we've really achieved whole-process feedback. I was initially optimistic, as the idea of stationary distributions appeared to collapse all the meta levels down to one. However, now I think there actually is a problem with the highest level of the tree. The initial distribution could be predominantly malign. Those malign UTAAs could select innocent-looking (but deadly) UTAAs for the next generation. In this way, the malign code could disappear, while achieving its goals by introducing subtle bugs to all subsequent generations of UTAAs.

The way I've specified things, trying to update against these malign UTAAs wouldn't work, because they're already absent in the stationary distribution.

Of course, you could directly update against them in the initial distribution. This could eliminate select malign UTAAs. The problem is that this kind of process feedback loses generalizability again. Since it's the top level of the tree, there's nothing above it which is selecting it, so we don't get to update against any general selection behaviors which produced the malign UTAAs.

The only way out of this I see at present is to parameterize the system's beliefs directly as a probability distribution over stationary distributions.. You can think of this as assuming that the initial distribution is already a stationary distribution. This way, when we update against malign UTAAs at the beginning of the process, we update against them occurring at any point in the process, which means we also update against any UTAAs which help select malign UTAAs, and therefore get generalization power.

But this seems like an annoyingly non-constructive solution. How are we supposed to work with the set of fixed points directly without iterating (potentially malign) code to find them?

---

*: Actually, a UTAA should be a compact specification of such tuple, such as a program or neural network which can output the desired objects. This is necessary for implementation, since EG we can't store V as a big table of values or S as a big table of probabilities. It also will allow for better generalization, and process-level feedback.

Normativity

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

Now that I've written Learning Normativity, I have some more clarity around the concept of "normativity" I was trying to get at, and want to write about it more directly. Whereas that post was more oriented toward the machine learning side of things, this post is more oriented toward the philosophical side. However, it *is* still relevant to the research direction, and I'll mention some issues relevant to value learning and other alignment approaches.

How can we talk about what you "should" do?

A Highly Dependent Concept

Now, obviously, what you should do depends on your goals. We can (at least as a rough first model) encode this as a utility function (but see my objection).

What you should do also depends on what's the case. Or, really, it depends on what you *believe* is the case, since that's what you have to go on.

Since we also have uncertainty about values (and we're interested in building machines which should have value uncertainty as well, in order to do value learning), we have to talk about beliefs-about-goals, too. (Or beliefs about utility functions, or however it ends up getting formalized.) This includes moral uncertainty.

Even worse, we have a lot of uncertainty about decision theory -- that is, we have uncertainty about how to *take* all of this uncertainty we have, and make it into decisions. Now, ideally, decision theory is not something the normatively correct thing *depends on*, like all the previous points, but rather is *a framework for finding the normatively correct thing given all of those things*. However, as long as we're uncertain about decision theory, we have to take that uncertainty as input too -- so, if decision theory is to give advice to realistic agents who are themselves uncertain about decision theory, *decision theory also takes decision-theoretic uncertainty as an input*. (In the best case, this makes bad decision theories capable of self-improvement.)

Clearly, we can be uncertain about how *that* is supposed to work.

By now you might get the idea. "Should" depends on some necessary information (let's call them the "givens"). But for each set of givens you claim is complete, there can be reasonable doubt about how to *use* those givens to determine the output. So we can create meta-level givens about how to use those givens.

Rather than stopping at some finite level, such as learning the human utility function, I'm claiming that we should learn all the levels. This is what I mean by "normativity" -- the information at all the meta-levels, which we would get if we were to unpack "should" forever. I'm putting this out there as my guess at the right type signature for human values.

I'm not mainly excited about this because I'm especially excited about including moral uncertainty or uncertainty about the correct decision theory into a friendly AI -- or because I think those are going to be particularly huge failure modes which we need to avert. Rather, I'm excited about this because it is the first time I've felt like I've had *any handles at all* for getting basic alignment problems right (wireheading, human manipulation, goodharting, ontological crisis) without a feeling that things are obviously going to blow up in some other way.

Normative vs Descriptive Reasoning

At this stage you might accuse me of committing the "turtles all the way down" fallacy. In Passing The Recursive Buck, Eliezer describes the error of accidentally positing an infinite hierarchy of explanations:

> The general antipattern at work might be called "Passing the Recursive Buck".
>
> [...]
>
> How do you stop a recursive buck from passing?
>
> You use the counter-pattern: *The Recursive Buck Stops Here.*
>
> But how do you apply this counter-pattern?
>
> You use the recursive buck-stopping trick.
>
> And what does it take to execute this trick?
>
> Recursive buck stopping talent.
>
> And how do you develop this talent?
>
> Get a lot of practice stopping recursive bucks.
>
> Ahem.

However, In Where Recursive Justification Hits Rock Bottom, Eliezer discusses a kind of infinite-recursion reasoning applied to normative matters. He says:

> But I would nonetheless emphasize the difference between saying:
>
> "Here is this assumption I cannot justify, which must be simply taken, and not further examined."
>
> Versus saying:
>
> "Here the inquiry continues to examine this assumption, with the full force of my *present intelligence*—as opposed to the full force of something else, like a random number generator or a magic 8-ball—even though my present intelligence happens to be founded on this assumption."
>
> Still... wouldn't it be nice if we could examine the problem of how much to trust our brains *without* using our current intelligence? Wouldn't it be nice if we could

> examine the problem of how to think, *without* using our current grasp of rationality?
>
> When you phrase it *that* way, it starts looking like the answer might be "No".

So, *with respect to normative questions,* such as what to believe, or how to reason, we can and (to some extent) should keep unpacking reasons forever -- every assumption is subject to further scrutiny, and *as a practical matter* we have quite a bit of uncertainty about meta-level things such as our values, how to think about our values, etc.

This is true despite the fact that *with respect to the descriptive questions* the recursive buck *must* stop somewhere. Taking a descriptive stance, my values and beliefs live in my neurons. From this perspective, "human logic" is not some advanced logic which logicians may discover some day, but rather, just the set of arguments humans actually respond to. Again quoting another Eliezer article,

> The phrase that once came into my mind to describe this requirement, is that a mind must be *created already in motion.* There is no argument so compelling that it will give dynamics to a static thing. There is no computer program so *persuasive* that you can run it on a rock.

So *in a descriptive sense* the ground truth about your values is just what you would actually do in situations, or some information about the reward systems in your brain, or something resembling that. *In a descriptive sense* the ground truth about human logic is just the sum total of facts about which arguments humans will accept.

But *in a normative sense,* there is no ground truth for human values; instead, we have an updating process which can change its mind about any particular thing; and that updating process itself is not the ground truth, but rather has beliefs (which can change) about what makes an updating process legitimate. Quoting from the relevant section of Radical Probabilism:

> The radical probabilist does not trust *whatever they believe next*. Rather, the radical probabilist has a concept of *virtuous epistemic process*, and is willing to believe the next output of such a process. Disruptions to the epistemic process do not get this sort of trust without reason.

I worry that many approaches to value learning attempt to learn a *descriptive* notion of human values, rather than the *normative* notion. This means stopping at some specific proxy, such as what humans say their values are, or what humans reveal their preferences to be through action, rather than leaving the proxy flexible and trying to learn it as well, while also maintaining uncertainty about *how* to learn, and so on.

I've mentioned "uncertainty" a lot while trying to unpack my hierarchical notion of normativity. This is partly because I want to insist that we have "uncertainty at every level of the hierarchy", but also because uncertainty *is itself* a notion to which normativity applies, and thus, generates new levels of the hierarchy.

Normative Beliefs

Just as one might argue that logic should be based on a specific set of axioms, with specific deduction rules (and a specific sequent calculus, etc), one might similarly

argue that uncertainty should be managed by a specific probability theory (such as the Kolmogorov axioms), with a specific kind of prior (such as a description-length prior), and specific update rules (such as Bayes' Rule), etc.

This general approach -- that we set up our bedrock assumptions from which to proceed -- is called "foundationalism".

I claim that we can't keep strictly to Bayes' Rule -- not if we want to model highly-capable systems in general, not if we want to describe human reasoning, and not if we want to capture (the normative) human values. Instead, how to update in a specific instance is a more complex matter which agents must figure out.

I claim that the Kolmogorov axioms don't tell us how to reason -- we need more than an uncomputable ideal; we also need advice about what to do in our boundedly-rational situation.

And, finally, I claim that length-based priors such as the Solomonoff prior are malign -- description length seems to be a really important heuristic, but there are other criteria which we want to judge hypotheses by.

So, overall, I'm claiming that a normative theory of belief is a lot more complex than Solomonoff would have you believe. Things that once seemed objectively true now look like rules of thumb. This means the question of normativity correct behavior is wide open even in the simple case of trying to predict what comes next in a sequence.

Now, Logical Induction addresses all three of these points (at least, giving us progress on all three fronts). We could take the lesson to be: we just had to go "one level higher", setting up a system like logical induction which *learns how* to probabilistically reason. *Now* we are at the right level for foundationalism. *Logical induction,* not classical probability theory, is the right principle for codifying correct reasoning.

Or, if not logical induction, perhaps the *next* meta-level will turn out to be the right one?

But what if we don't *have* to find a foundational level?

I've updated to a kind of quasi-anti-foundationalist position. I'm not against finding a strong foundation *in principle* (and indeed, I think it's a useful project!), but I'm saying that as a matter of fact, we have a lot of uncertainty, and it sure would be nice to have a normative theory which allowed us to account for that (a kind of afoundationalist normative theory -- not anti-foundationalist, but not strictly foundationalist, either). This should still be a strong formal theory, but one which requires weaker assumptions than usual (in much the same way reasoning about the world via probability theory requires weaker assumptions than reasoning about the world via pure logic).

Stopping at $\aleph_0$

My main objection to anti-foundationalist positions is that they're just *giving up*; they don't answer questions and offer insight. Perhaps that's a lack of understanding on my part. (I haven't tried that hard to understand anti-foundationalist positions.) But I still feel that way.

So, rather than give up, I want to provide a framework which holds *across* meta-levels (as I discussed in Learning Normativity).

This would be a framework in which an agent can balance uncertainty at all the levels, without dogmatic foundational beliefs at any level.

Doesn't this just create a new infinite meta-level, above all of the finite meta-levels?

A mathematical analogy would be to say that I'm going for "cardinal infinity" rather than "ordinal infinity". The first ordinal infinity is ω, which is greater than all finite numbers. But ω is less than $\omega + 1$. So building something at "level ω" would indeed be "just another meta-level" which could be surpassed by level $\omega + 1$, which could be surpassed by $\omega + 2$, and so on.

Cardinal infinities, on the other hand, don't work like that. The first infinite cardinal is $\aleph_0$, but $\aleph_0 + 1 = \aleph_0$ -- we can't get bigger by adding one. This is the sort of meta-level I want: a meta-level which also *oversees itself* in some sense, so that we aren't just creating a new level at which problems can arise.

This is what I meant by "collapsing the meta-levels" in Learning Normativity. The finite levels might still exist, but there's a level at which everything can be put together.

Still, *even so*, isn't this still a "foundation" at some level?

Well, yes and no. It should be a framework in which a very broad range of reasoning could be supported, while also making some rationality assumptions. In this sense it would be a theory of rationality purporting to "explain" (ie categorize/organize) all rational reasoning (with a particular, but broad, notion of rational). In this sense it seems not so different from other foundational theories.

On the other hand, this would be something more provisional by design -- something which would "get out of the way" of a real foundation if one arrived. It would seek to make far fewer claims overall than is usual for a foundationalist theory.

What's the hierarchy?

So far, I've been pretty vague about the actual hierarchy, aside from giving examples and talking about "meta-levels".

The $\aleph_0$ analogy brings to mind a linear hierarchy, with a first level and a series of higher and higher levels. Each next level does something like "handling uncertainty about the previous level".

However, my recursive quantilization proposal created a *branching* hierarchy. This is because the building block for that hierarchy required several inputs.

I think the exact form of the hierarchy is a matter for specific proposals. But I do think some specific levels ought to exist:

- Object-level values.
- Information about value-learning, which helps update the object-level values.
- Object-level beliefs.
- Generic information about what distinguishes a good hypothesis. This includes Occam's razor as well as information about what makes a hypothesis malign.

Normative Values

It's difficult to believe humans have a utility function.

It's easier to believe humans have expectations on propositions, but this still falls apart at the seams (EG, not all propositions are explicitly represented in my head at a given moment, it'll be difficult to define exactly which neural signals are the expectations, etc).

We can try to define values as what we would think if we had a really long time to consider the question; but this has its own problems, such as humans going crazy or experiencing value drift if they think for too long.

We can try to define values as what a human would think after an hour, if that human had access to HCH; but this relies on the limited ability of a human to use HCH to accelerate philosophical progress.

Imagine a value-learning system where you don't have to give any solid definition of what it is for humans to have values, but rather, can give a number of proxies, point to flaws in the proxies, give feedback on how to reason about those flaws, and so on. The system would try to generalize all of this reasoning, to figure out what the thing being pointed at could be.

We could describe humans deliberating under ideal conditions, point out issues with humans getting old, discuss what it might mean for those humans to go crazy or experience value drift, examine how the system is reasoning about all of this and give feedback, discuss what it would mean for those humans to reason well or poorly, ...

We could never entirely pin down the concept of human values, but at some point, the system would be reasoning so much like us (or rather, so much like we would want to reason) that this wouldn't be a concern.

Comparison to Other Approaches

This is most directly an approach for solving meta-philosophy .

Obviously, the direction indicated in this post has a lot in common with Paul-style approaches. My outside view is that this is me reasoning my way around to a Paul-ish position. However, my inside view still has significant differences, which I haven't fully articulated for myself yet.

Recursive Quantilizers II

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

I originally introduced the recursive quantilizers idea here, but didn't provide a formal model until my recent Learning Normativity post. That formal model had some problems. I'll correct some of those problems here. My new model is closer to HCH+IDA, and so, is even closer to Paul Christiano style systems than my previous.

However, I'm also beginning to suspect that quantilizers aren't the right starting point. I'll state several problems with quantilizers at the end of this post.

First, let's reiterate the design criteria, and why the model in Learning Normativity wasn't great.

Criteria

Here are the criteria from Learning Normativity, with slight revisions. See the earlier post for further justifications/intuitions behind these criteria.

1. **No Perfect Feedback:** we want to be able to learn with the possibility that any one piece of data is corrupt.
    1. **Uncertain Feedback:** data can be given in an uncertain form, allowing 100% certain feedback to be given (if there ever is such a thing), but also allowing the system to learn significant things in the absence of any certainty.
    2. **Reinterpretable Feedback:** ideally, we want *rich hypotheses about the meaning of feedback,* which help the system to identify corrupt feedback, and interpret the information in imperfect feedback. To this criterion, I add two clarifying criteria:
        1. **Robust Listening:** in some sense, we don't want the system to be able to "entirely ignore" humans. If the system goes off-course, we want to be able to correct that.
        2. **Arbitrary Reinterpretation:** at the same time, we want the AI to be able to *entirely reinterpret* feedback based on a rich model of what humans mean. This criterion stands in tension with Robust Listening. However, the proposal in the present post is, I think, a plausible way to achieve both.
2. **No Perfect Loss Function:** we don't expect to perfectly define the utility function, or what it means to correctly learn the utility function, or what it means to learn to learn, and so on. At no level do we expect to be able to provide a single function we're happy to optimize. This is largely due to a combination of Goodhart and corrupt-feedback concerns.
    1. **Learning at All Levels:** Although we don't have perfect information at any level, we do get meaningful benefit with each level we step back and say "we're learning this level rather than keeping it fixed", because we can provide meaningful approximate loss functions at each level, and meaningful feedback for learning at each level. Therefore, we want to be able to do learning at each level.
    2. **Between-Level Sharing:** Because this implies an infinite hierarchy of levels to learn, we need to share a great deal of information between levels in order to learn meaningfully. For example, Occam's razor is an important heuristic at each level, and information about what malign inner optimizers look like is the same at each level.
3. **Process Level Feedback:** we want to be able to give feedback about *how to arrive at answers*, not just the answers themselves.

1. **Whole-Process Feedback:** we don't want some segregated meta-level which accepts/implements our process feedback about the rest of the system, but which is immune to process feedback itself. Any part of the system which is capable of adapting its behavior, we want to be able to give process-level feedback about.
2. **Learned Generalization of Process Feedback:** we don't just want to promote or demote specific hypotheses. We want the system to learn from our feedback, making generalizations about which *kinds* of hypotheses are good or bad.

Failed Criteria

The previous recursive-quantilization model failed some criteria:

- No reinterpretable feedback. I didn't provide *any* method for achieving that.
- No whole-process feedback. The way I set things up, the initial distributions are judged only on their later consequences. This leaves them wide open to inner optimizers and other problematic reasoning steps.
    - We can fix this by allowing the user to give direct feedback on the initial distributions as well, but then there's no mechanism for Learned Generalization of that particular feedback. So we're caught in the middle, unable to satisfy both those criteria at once.

The current proposal solves both problems, and due to an analogy to iterated amplification, may also be more computationally feasible.

The New Proposal

Like iterated amplification, the new proposal consists of both an *idealized definition of aligned behavior* (HCH, in the context of iterated amplification) and *a candidate approximation of this ideal* (like iterated amplification itself, which is supposed to approximate HCH).

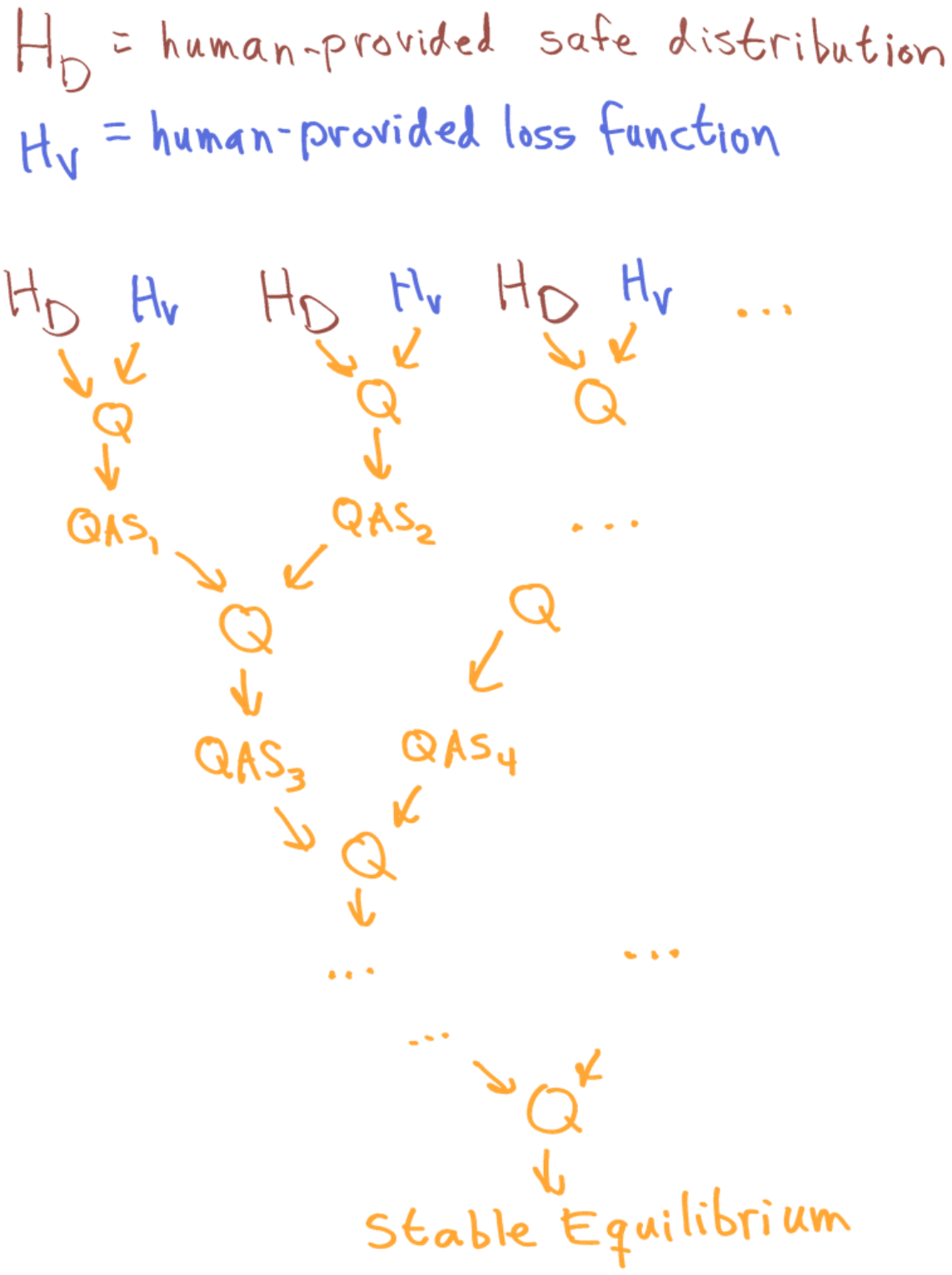
The Ideal

The object which quantilization will select on will be referred to as "question-answering systems", or QAS for short. This is what I called a "UTAA" in the previous post. As before, this is one object which has opinions about the safe distribution for quantilization (you can ask it "what's a safe distribution over QAS to quantilize on?"), and as value function ("give me a value function to judge the quality of QAS") and as the object-level solution to whatever problem you're trying to get this whole setup to solve (you ask it your object-level questions).

So the goal is to get a really good QAS, where "really good" means highly capable and highly aligned. Since we're specifying the ideal here, we get to use unbounded processing power in doing so.

Humans provide an initial "safe" distribution on QASs, and an initial loss function which will be used to judge QASs for their quality. We then quantilize. This yields a new distribution over QASs, which we use to define another stage of quantilization (this time with safe distribution and value function coming from the QASs themselves), and so on, until we reach an equilibrium where an additional step does not add anything.

As before, we can imagine this as an infinite or very large tree of quantilization:



(Note that, previously, I treated Q as a three-argument stochastic function, requiring a safe distribution, a value function (AKA loss function), and a parameter telling it how much to optimize. Here, I am suppressing the third parameter. Perhaps we keep it fixed, or perhaps we ask a third QAS for it, as before -- I have no improved suggestions for how we estimate that parameter at the moment.)

The tree structure means that a QAS is never evaluating itself; it is being evaluated by another QAS sampled from an independent multi-level quantilization, which can spot mistakes in its own way.

Now, loss functions *in general* (including both the initial human-given loss function, and the improved loss functions proposed by the QASs) need access to *the entire experience of the system*. The initial human-gives loss function may be something like "pick out explicit training feedback humans give, and translate it into a numerical score for a QAS, ignoring natural-language comments made by the humans, evidence from the context, etc." The loss functions proposed by the QAS are supposed to get cleverer, interpreting natural-language feedback given by humans, using contextual clues, etc.

You might ask, "shouldn't we build simple natural-language understanding into the initial loss function?" -- we could, but that's supposed to be redundant. There should be no reason to train a separate natural-language understanding system, rather than use explicit feedback to train the QAS to understand natural-language feedback.

Uncertain Feedback

The initial loss function *should* have the ability to learn from uncertain feedback, to satisfy the Uncertain Feedback criterion. (We can implement one of the options mentioned in Learning Normativity.)

Balancing Robust Listening and Arbitrary Reinterpretation

The idealized objective presented in this section offers a nice combination of "robust listening" (not ever reaching a state where it totally fails to respond to human feedback) and "arbitrary reinterpretation" (being able to use rich models to totally reinterpret the meaning of human feedback). These two objectives would seem to be at odds, since a system with an arbitrary model of the meaning of human feedback can ignore anything.

In the proposal here, *direct human feedback always shapes the initial dynamic*. If you see a system do something stupid and add direct feedback against this to the data, then QASs which engage in this behavior will be ruled out in the first step of quantilization. You can also add strong evidence against QAS which fail to judge QASs harshly for violating such principles (and which fail to punish non-punishers, recursively). So, human feedback always has total control to shape the ultimate result.

At the same time, however, *only the initial dynamic is required to obey this feedback in a naive way*. At later stages, QASs are supposed to understand human feedback better, identifying feedback which should be interpreted less literally.

This means the equilibrium result is both fully dependent on a 100% literal interpretation of human feedback, and at the same time, fully the result of a more careful, nuanced interpretation of what humans meant.

Process-Level Feedback

Also, it's important to note that the initial loss function needs to understand process-level feedback. You can imagine that the dataset includes the following:

- Extensive human-generated Q&A behavior, much like would be needed for Iterative Amplification. This is the bulk of what loads general world-knowledge into QAS that come out of the first round of quantilization.
- Extensive labeling of QAS as "ok" and "bad", to demonstrate how to examine QAS source code (/ neural network / etc) and evaluate when something looks wrong. This could include many toy examples which illustrate bad behavior, as well as real examples gathered during training. The QAS should be trained to imitate this behavior when evaluating other QAS.

- Similarly, examples of humans examining QAS execution and labeling steps as bad, rather than just examining the source code directly.

You can imagine humans engaging in an iterative process where they examine the behavior of many QAS created by early attempts at this process, which creates more training data for later iterations. However, since we're still only describing the *ideal*, that's not realistic; humans never have access to the true output of the ideal process.

Let's move on to describing the iterative-amplification analogue.

The Implementation Proposal

The basic idea is to take iterative amplification, and replace the HCH-like "amplification" operation with quantilization-based amplification.

At every stage of the process, we're not learning a single QAS, but rather, learning a *distribution over QAS.*

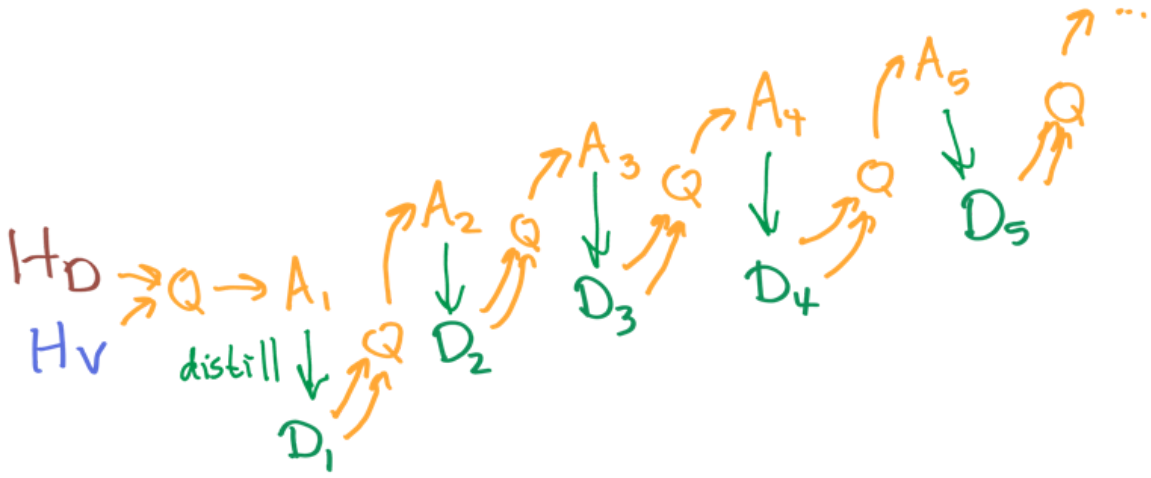
Define the distribution A_1 as the result of quantilizing the safe distribution given by humans, using the value function given by humans.

The distribution D_1 is the distillation of A_1; ie, we learn a more computationally efficient approximation of the distribution A_1 using some distribution-modeling technique such as GANs.

The distribution A_2 is defined by sampling two QAS from D_1, asking one for the value function and the other for the safe distribution, and quantilizing.

The distribution D_2 is obtained by learning an efficient approximation of A_2.

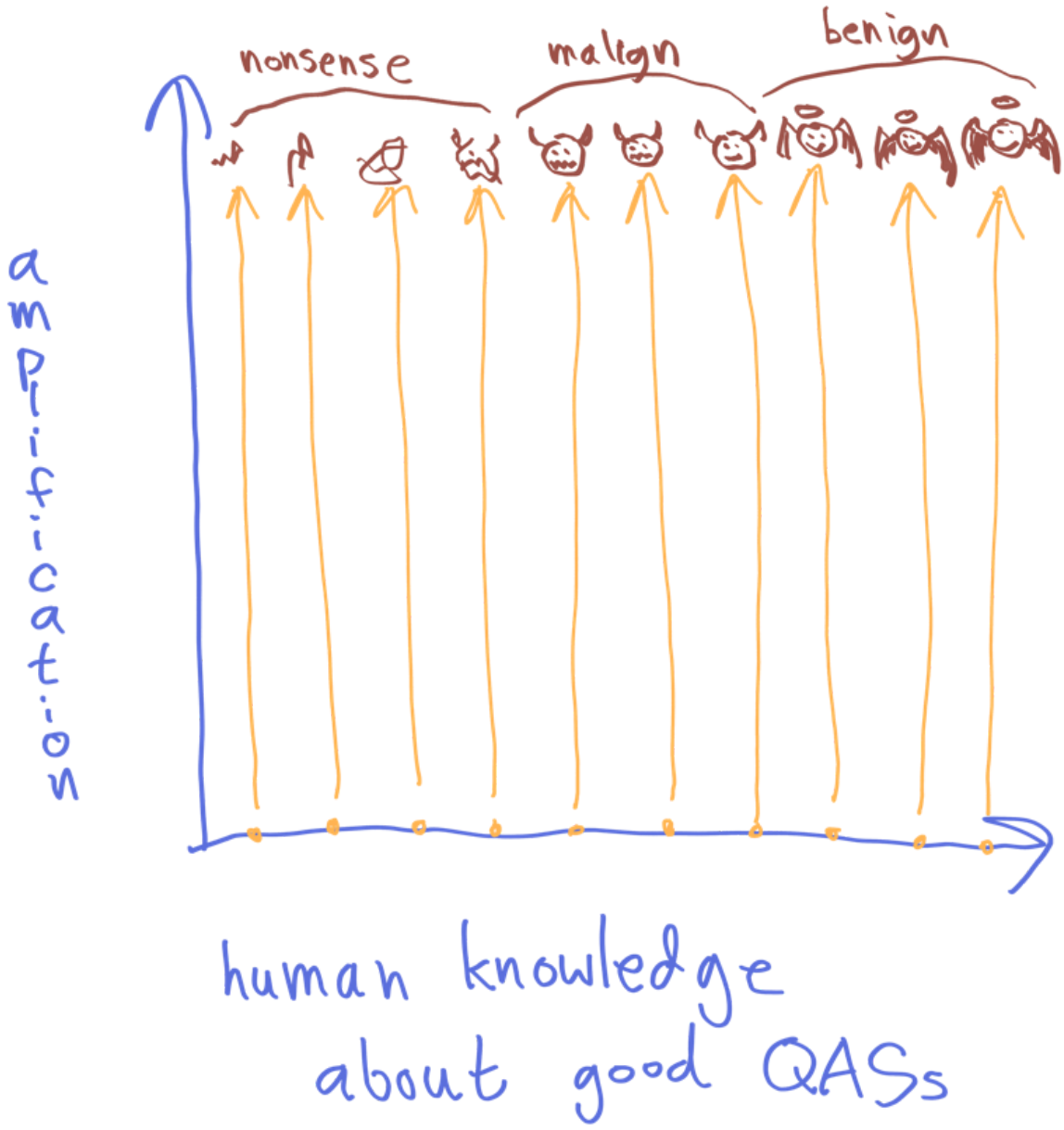
And so on. A_n is defined as the output of a quantilizer which takes its value function and safe distribution from (independent) samples from D_{n-1}. D_n is a learned approximation of A_n.



If the distillation step were perfect, this would converge to the desired equilibrium.

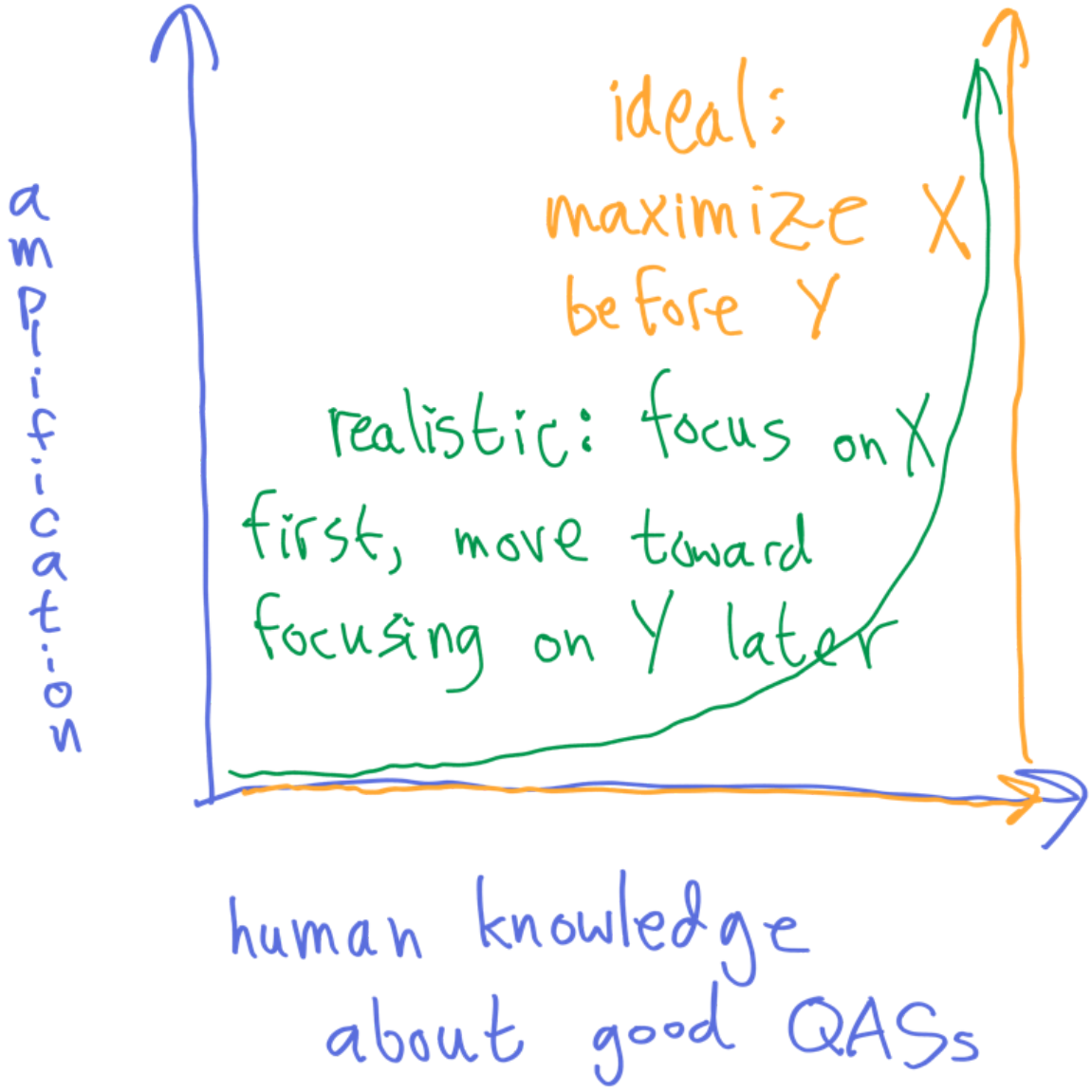
Note, however, that this assumes that all human feedback is available from the beginning. Realistically, we want more interactive training, so that humans can observe the intermediate outputs of the process and provide new training data to account for problems. The vast majority of training data involving QAS evaluation may come from such interaction.

I visualize this as "iterated amplification takes us upward, but not rightward" in the following coordinates:



If we start amplifying something with very little human-input knowledge, we're likely to get total nonsense out; the QASs just don't know enough to optimize any coherent thing over successive amplification steps. If we start with a moderate amount of knowledge, the QASs might sputter into nonsense, or might get on the right track, but also have a high risk of producing highly-capable but malign agents. At some level of sufficient human input, we fall into the "basin of corrigibility" and get aligned outputs no matter what (perhaps even the same exact benign output).

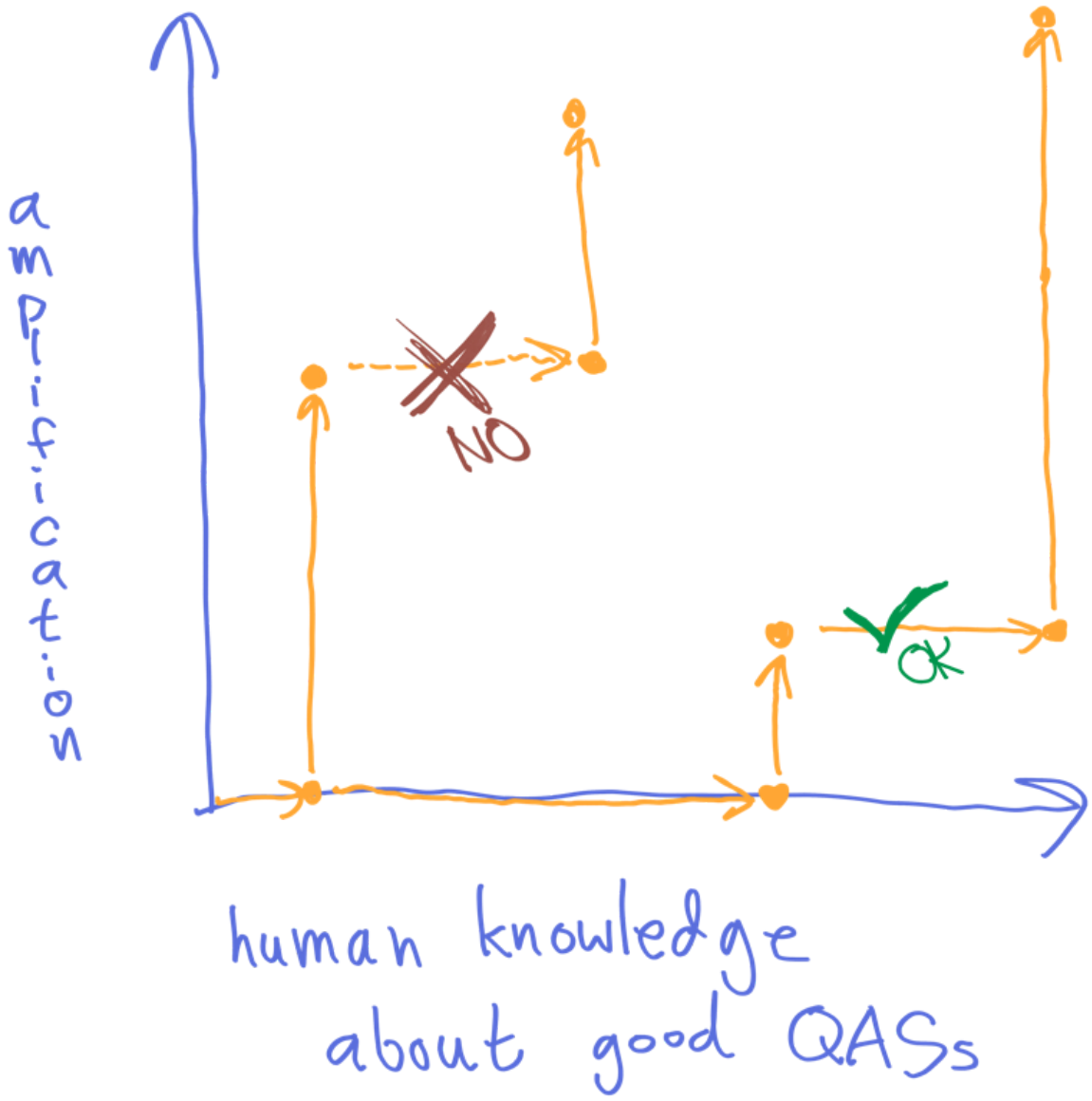
Yet, it's hard to provide everything up-front. So, more realistically, the picture might look like this:



The golden line is the standard set by the idealized model of recursive quantilizers, where we have all the training data before starting the amplification process. The green line is a more realistic line, where we make amplification moves and increase training data in a mixed way, but avoid over-amplifying an under-trained model.

It's really important to stay close to the x-axis early in the process, because the system itself is determining how to evaluate the loss function -- so it's evaluating the very meaning of the training data (in line with the Reinterpretable Feedback criterion). It is therefore very important that we don't let the system drift too far in the direction of an extreme reinterpretation of the training data (in line with the Robust Listening criterion). At the very start of the training process, I imagine you'd often restart the training process from scratch with all the new feedback in place, rather than trust the system to understand new data.

In other words, we don't expect x-dimension moves to work if we're too high in the y-dimension:



Unfortunately, it's difficult to know what the region is where x-dimension moves work, so it's difficult to know when amplification would keep us within that region vs take us out of it.

Another way to put it: this implementation puts the "robust listening" criterion at serious risk. The partially amplified agent can easily stop listening to human feedback on important matters, about which it is sure we must be mistaken.

Really, we would want to find an engineering solution around this problem, rather than haphazardly steering through the space like I've described. For example, there might be a way to train the system to seek the equilibrium it would have reached if it had started with all its current knowledge.

Comparison to Iterated Amplification

Because this proposal is so similar to iterated amplification, it bears belaboring the differences, particularly the philosophical differences underlying the choices I've made.

I don't want this to be about critiquing iterated amplification -- I have some critiques, but the approach here is not mutually exclusive with iterated amplification by any means. Instead, I just want to make clear the philosophical differences.

Both approaches emphasize *deferring the big questions,* setting up a system which does all the philosophical deliberation for us, rather than EG providing a correct decision theory.

Iterated Amplification *puts humans in a central spot.* The amplification operation is giving a human access to an (approximate) HCH -- so at every stage, a human is making the ultimate decisions about how to use the capabilities of the system to answer questions. This plausibly has alignment and corrigibility advantages, but may put a ceiling on capabilities (since we have to rely on the human ability to decompose problems well, creating good plans for solving problems).

Recursive quantilization instead seeks to *allow arbitrary improvements to the deliberation process*. It's all supervised by humans, and initially seeded by imitation of human question-answering; but humans can point out problems with the human deliberation process, and the system seeks to improve its reasoning using the human-seeded ideas about how to do so. To the extent that humans think HCH is the correct idealized reasoning process, recursive quantilization should approximate HCH. (To the extent it can't do this, recursive quantilization fails at its goals.)

One response I've gotten to recursive quantilization is "couldn't this just be advice to the human in HCH?" I don't think that's quite true.

HCH must walk a fine line between capability and safety. A big HCH tree can perform well at a vast array of tasks (if the human has a good strategy), but in order to be safe, the human must operate under set of restrictions, such as "don't simulate unrestricted search in large hypothesis spaces" -- with the full set of restrictions required for safety yet to be articulated. HCH needs a set of restrictions which provide safety without compromising capability.

In Inaccessible Information, Paul draws a distinction between accessible and inaccessible information. Roughly, information is *accessible* if we have a pretty good shot at getting modern ML to tell us about it, and inaccessible otherwise. Inaccessible information can include intuitive but difficult-to-test variables like "what Alice is thinking", as well as superhuman concepts that a powerful AI might invent.

A powerful modern AI like GPT-3 might *have* and *use* inaccessible information such as "what the writer of this sentence was really thinking", but we can't get GPT-3 to *tell* us about it, because we lack a way to train it to.

One of the safety concerns of inaccessible information Paul lists is that powerful AIs might be more capable than aligned AIs due to their ability to utilize inaccessible information, where aligned AIs cannot. For example, GPT-5 might use inhuman concepts, derived from its vast experience predicting text, to achieve high performance. A safe HCH would never be able to use those concepts, since every computation within the HCH tree is supposed to be human-comprehensible. (Therefore, if the result of Iterated Amplification was able to use such concepts, we should be worried that it did not successfully approximate HCH.)

Paul proposes learning the human prior as a potential solution. As I understand it, the basic idea is that HCH lacks Deep Learning's ability to absorb vast quantities of data and reach new conclusions. By learning the human prior, Paul seeks to learn the *human* response to those vast quantities of data. This would allow an HCH-like approach to learn the same "alien concepts" that a misaligned AI might learn.

I don't really understand how Paul sees HCH and learned-priors as fitting together, Recursive Quantilization seeks to get around this difficulty by training the QASs on lots of data in a way similar to big-data ML. As I emphasized before, recursive quantilization seeks to allow arbitrary improvements to the reasoning process, so long as they *are* improvements.

Of course, recursive quantilization also needs to restrict the QAS reasoning processes via safety constraints (using process-level feedback to rule out types of reasoning which can't be verified as safe). This *could* lead to the same problems with inaccessible information. But if it does, that's because ***no*** *human-approved reasoning can approve of reasoning which utilizes the inaccessible information* **--** not even after many rounds of amplifying the human-seeded knowledge about how to judge reasoning as safe/unsafe. So, I would argue that if an approach broadly like recursive quantilization can't do it, then *no robustly aligned method can do it*.

Reasons why quantilizers might not be appropriate:

As I mentioned at the beginning, I've been feeling like quantilizers might be the wrong starting point for the Learning Normativity agenda. There's still some hope, but there are a few problems which make me want to look for other directions.

Quantilizers bound risk, but iteration increases risk arbitrarily.

The whole argument behind quantilization is as follows:

1. The safe distribution has below ***c*** probability of catastrophe.
2. The limited optimization samples from the top ***q*** of the safe distribution.
3. So, the risk of catastrophe from the limited optimization is at most ***c/q***.

In other words, the chance of catastrophic risk is increased exactly in proportion to how much we optimize.

Recursive quantilization does this *over and over and over again.* Naively, as we seek the fixed point, it seems like our bound on catastrophic risk gets worse and worse until we have no bound at all.

There could be a way around this. We could get a *better* risk bound, which took into account the *improved* optimization from each step in the recursive quantilization. The story could be something like this:

- In each quantilization step, we have an overwhelming chance of increasing the quality of value functions and safe distributions proposed in the next round.
- This means (with overwhelming odds) we get better at detecting catastrophe, so we can actively optimize against problems which the previous round of quantilization had to merely hedge against by avoiding over-optimizing.
- Therefore, the chance of catastrophe from each round of quantilization can be dramatically lower than the previous. Although these chances do add up over successive rounds of quantilization, they can sum to a negligible probability overall, even in the limit.

It would be nice to put together such a story, but it isn't clear how to argue that the improvement in value function and proposal distribution cuts into the probability of catastrophe by a sufficient amount.

Quantilizers assume catastrophes are rare in random choices, but in the recursive setting, too

much randomness can create value drift.

Quantilization assumes we can get a "safe distribution" which might produce mediocre actions, but at least has a very low probability of catastrophic risk.

Supposing the scheme could produce highly capable systems at all, the recursive quantilizer environment seems like one where *randomized actions have a good chance of producing catastrophic risk,* so safe distributions would be incredibly difficult to engineer.

Imagine we're computing a big recursive quantilization tree. Randomly selecting a poor value function at one point will have a cascading effect. It will feed into a quantilization step which selects a bad QAS, which will go on to produce either the value function or the safe distribution for another quantilization, and so on. There's not fundamentally any corrective effect -- if either of the two inputs to quantilization is poor enough, then the output will be poor.

There's a basin of attraction, here: if the proposed safe distribution always contains good proposals with non-negligible probability, and the value function always has enough of the right meta-principles to correct specific errors that may be introduced through random error. But it's quite important that the output of each quantilization be *better* than the previous. If not, then we're not in a basin of attraction.

All of this makes it sound quite difficult to propose a safe distribution. The safe distribution needs to *already* land you within the basin of attraction (with very high probability), because drifting out of that basin can easily create a catastrophe.

Here's a slightly different argument. At each quantilization step, including the very first one, it's important that we find a QAS which actually fits our data quite well, because it is important that we pin down various things firmly in order to remain in the basin of attraction (especially including pinning down a value function at least as good as our starting value function). However, for each QAS which fits the data quite well and points to our desired basin of attraction, there are many alternative QAS which *don't* fit our data well, but point to very *different,* but equally coherent, basins of attraction. (In other words, there should be many equally internally consistent value systems which have basins of attraction of similar size.)

Since these other basins would be catastrophic, this means ***c*,** the probability of catastrophe, is higher than ***q*,** the amount of optimization we need to hit our narrow target.

This means the safe distributions has to be doing a lot of work for us.

Like the previous problem I discussed, this isn't necessarily a showstopper, but it does say that we'd need some further ideas to make recursive quantilization work, and suggests to me that quantilization might not be the right way to go.

Other Concerns

- Quantilizers don't have the best handles for modeling human philosophical deliberation over time. In other words, I don't think recursive quantilization absorbs the lesson of radical probabilism. In particular, although recursive quantilization involves iteratively improving a picture of "good reasoning", I think it lacks a kind of *stability* -- the picture of good reasoning must be entirely reconstructed each time, from "first principles" (IE from the principles developed in the previous step). I currently see no guarantee that recursive quantilization avoids being Dutch-Book-able over these stages, or any other such dynamic optimality notion.
- Quantilizers aren't great for modeling a collection of partial models. Since a quantilizer spits out one (moderately) optimized result, I have to focus on single QASs, rather than

collections of experts which cover different areas of expertise. This means we don't get to break down the problem of reasoning about the world.

- Quantilizers don't put world models in a central spot. By putting optimization in a central spot, we sort of sideline reasoning and belief. This obscures the mechanism of updating on new information.

The Pointers Problem: Clarifications/Variations

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

I've recently had several conversations about John Wentworth's post The Pointers Problem. I think there is some confusion about this post, because there are several related issues, which different people may take as primary. All of these issues are important to "the pointers problem", but John's post articulates a specific problem in a way that's not quite articulated anywhere else.

I'm aiming, here, to articulate the cluster of related problems, and say a few new-ish things about them (along with a lot of old things, hopefully put together in a new and useful way). I'll indicate which of these problems John was and wasn't highlighting.

This whole framing *assumes* we are interested in something like value learning / value loading. Not all approaches rely on this. I am not trying to claim that one *should* rely on this. Approaches which don't rely on human modeling are neglected, and need to be explored more.

That said, some form of value loading may turn out to be very important. So let's get into it.

Here's the list of different problems I came up with when trying to tease out all the different things going on. These problems are all closely interrelated, and feed into each other to such a large extent that they can seem like one big problem.

(**0. Goodhart.** This is a background assumption. It's what makes getting pointers right *important.*)

1. **Amplified values.** Humans can't evaluate options well enough that we'd just want to optimize the human evaluation of options. This is part of what John is describing.
2. **Compressed pointer problem.** We can't realistically just give human values to a system. How can we give a system a small amount of information which "points at" human values, so that it will then do its best to learn human values in an appropriate way?
3. **Identifiability problems for value learning.** This includes Stuart's "no free lunch" argument that we can't extract human values (or beliefs) just with standard ML approaches.
4. **Ontology mismatch problem.** Even if you *could* extract human values (and beliefs) precisely, whatever that means, an AI doesn't automatically know how to optimize them, because they're written in a different ontology than the AI uses. This is what John is primarily describing.
5. **Wireheading and human manipulation.** Even if we solve both the learning problem and the ontology mismatch problem, we may still place the AI under some perverse incentives by telling it to maximize human values, if it's also still uncertain about those values. Hence, there still seems to be an extra problem of telling the AI to "do what we would want you to do" even after all of that.

0. Goodhart

As I mentioned already, this is sort of a background assumption -- it's not "the pointers problem" itself, but rather, tells us why the pointers problem is hard.

Simply put, Goodhart's Law is an argument that an approximate value function is almost never good enough. You really need to get quite close before optimizing the approximate version is a good way to optimize human values. Scott gives four different types of Goodhart, which all feed into this.

Siren Worlds

Even if we had a perfect human model which we could use to evaluate options, we would face the Siren Worlds problem: we optimize for options which *look good* to humans, but this is different from options which *are good*.

We can't look at (even a sufficient summary of) the entire universe. Yet, we potentially *care about* the whole universe. We don't want to optimize just for parts we can look at or summarize, at the expense of everything else.

This shows that, to solve the pointers problem, we need to do more than just model humans perfectly. John's post talked about this problem in terms of "lazy evaluation": we humans can only instantiate small parts of our world model when evaluating options, but our "true values" would evaluate everything.

I'm referring to that here as the problem of specifying *amplified values.*

1. Amplified Values Problem

The problem is: humans lack the raw processing power to properly evaluate our values. This creates a difficulty in *what it even means* for humans to have specific values. It's easy to say "we can't evaluate our values precisely". What's difficult is to specify *what it would mean* for us to evaluate our values more precisely.

A quick run-down of some amplification proposals:

- Iterated Amplification: amplification is defined recursively as the answer a human would give if the human had help from other amplified humans who could answer questions.
- Debate: the amplified human opinion is defined as the winner of an idealized debate process, judged by humans (with AI debaters).
- CEV: we define amplified human values as what we would think if we were somewhat smarter, knew somewhat more, and had grown up longer together.
- Recursive Reward Modeling: we define human values as a utility function which humans specify with the help of powerful agents, who are themselves aligned through a recursive process.

Aside: if we think of *siren worlds* as the primary motivator for amplification, then Stuart's no-indescribable-hellworlds hypothesis is very relevant for thinking about what amplification means. According to that hypothesis, *bad proposals must be* "*objectionable*" *in the sense of having an articulable objection which would make a human discard the bad proposal.* If this is the case, then debate-like proposals seem like a good amplification technique: it's the systematic unearthing of objections.

Now, a viable proposal needs two things:

1. An abstract statement of what it means to amplify;
2. A concrete method of getting there.

For example, Iterated Amplification gives HCH as the abstract model of an amplified human, and the iterated amplification training method as its concrete proposal for getting there.

Crossing this bridge is the subject of the next point, *compressed pointers:*

2. Compressed Pointer Problem

The basic idea behind *compressed pointers* is that you can have the abstract goal of cooperating with humans, without actually knowing very much about humans. In a sense, this means having *aligned goals* without having *the same goals* : your goal is to cooperate with "human goals", but you don't yet have a full description of what human goals are. Your value function might be much simpler than the human value function.

In machine-learning terms, this is the question of how to specify a loss function for the purpose of *learning* human values.

Some important difficulties of compressed pointers are that they seem to lead to new problems of their own, in the form of wireheading and human manipulation problems. We will discuss those problems later on.

The other important sub-problem of compressed pointers is, well, how do you actually do the pointing? An assumption behind much of value learning research is that we can point to the human utility function via a loss function which learns a model of humans which decomposes them into a utility function, a probability distribution, and a model of human irrationality. We can then amplify human values just by plugging that utility function into better beliefs and a less irrational decision-making process. I argue against making such a strong distinction between values and beliefs here, here, and here, and will return to these questions in the final section. Stuart Armstrong argues that such decompositions cannot be learned with standard ML techniques, which is the subject of the next section.

3. Identifiability Problems for Value Learning

Stuart Armstrong's no-free-lunch result for value learning shows that the space of possible utility functions consistent with data is always too large, and we can't eliminate this problem *even with Occam's razor*: even when restricting to simple options, it's easy to learn precisely the wrong values (IE flip the sign of the utility function).

One thing I want to emphasize about this is that this is just one of many possible non-identifiability arguments we could make. (*Identifiability* is the learning-theoretic property of being able to distinguish the correct model using the data; *non-identifiability* means that many possibilities will continue to be consistent, even with unlimited data.)

Representation theorems in decision theory, such as VNM, often *uniquely give us* the utility function of an agent from a set of binary decisions which the agent would make. However, in order to get a unique utility function, we must usually ask the agent to evaluate *far more decisions* than is realistic. For example, Savage's representation theorem is based on evaluating all possible mappings from states to outcomes. Many of these mappings will be nonsensical -- not only never observed in practice, but nowhere near anything which ever could be observed.

This suggests that just observing the *actual* decisions an agent makes is highly insufficient for pinning down utility functions. What the human preferred under the actual circumstances does not tell us very much about what the human *would have* preferred under different circumstances.

Considerations such as this point to a number of ways in which human values are not identifiable from human behavior alone. Stuart's result is particularly interesting in that it

strongly rules out fixing the problem via simplicity assumptions.

Learning with More Assumptions

Stuart responds to this problem by suggesting that we need *more input from the human.* His result suggests that the standard statistical tools alone won't suffice. Yet, humans seem to correctly identify what each other want and believe, quite frequently. Therefore, humans must have prior knowledge which helps in this task. If we can encode those prior assumptions in an AI, we can point it in the right direction.

However, another problem stands in our way even then -- even if the AI could perfectly model human beliefs and values, what if the AI does not share the ontology of humans?

4. Ontology Mismatch Problem

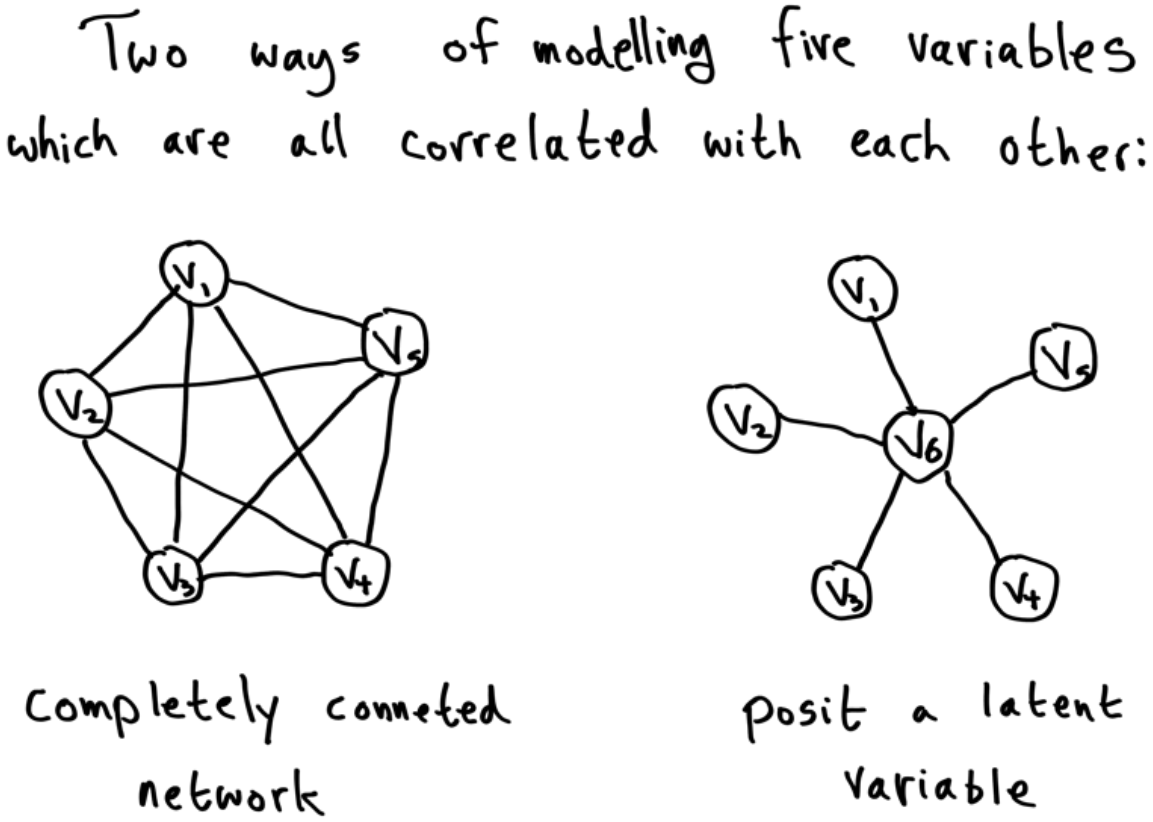
As I stated earlier, I think John's post was discussing a mix of the ontology mismatch problem and the amplification problem, focusing on the ontology mismatch problem. John provided a new way of thinking about the ontology mismatch problem, which focused on the following claim:

Claim: *Human values are a function of latent variables in our model of the world.*

Humans have latent variables for things like "people" and "trees". An AI needn't look at the world in exactly the same way, so it needn't *believe things exist* which humans predicate value on.

This creates a tension between *using the flawed models of humans* (so that we can work in the human ontology, thus understanding human value) vs *allowing the AI to have better models* (but then being stuck with the ontology mismatch problem).

As a reminder of what latent variables are, let's take a look at two markov networks which both represent the relationship of five variables:



In the example on the left, we posit a completely connected network, accounting for all the correlations. In the example on the right, we posit a new latent variable which accounts for the correlations we observed.

As a working example, *depression* is something we talk about as if it were a latent variable. However, many psychologists believe that it is actually a set of phenomena which happen to have strong mutually reinforcing links.

Considered *as a practical model*, we tend to prefer models which posit latent variables when:

- They fit the data better, or equally well;
- Positing latent variables decreases the description complexity;
- The latent variable model can be run faster (or has a tolerable computational cost).

Mother nature probably has similar criteria for when the brain should posit latent variables.

However, considered *as an ontological commitment*, it seems like we only want to posit latent variables *when they exist*. When a Bayesian uses the model on the right instead of the model on the left, they believe in V_6 as an *extra fact which can in principle vary independently of the other variables*. ("Independently" in the logical sense, here, not the probabilistic sense.)

So there is ambiguity between latent variables as pragmatic tools vs ontological commitments. This leads to our problem: a Bayesian (or a human), having invented latent variables for the convenience of their predictive power, may nonetheless ascribe value to specific states of those variables. This presents a difficulty if another agent wishes to *help* such an agent, but does not share its ontological commitments.

Helper's Perspective

Suppose for a moment that humans were incapable of understanding depression as a cluster of linked variables, simply because that view of reality was too detailed to hold in the mind -- but that humans wanted to eliminate depression. Imagine that we have an AI which can understand the details of all the linked variables, but which lacks a hidden variable "depression" organizing them all. The AI wants to help humans, and in theory, should be able to use its more detailed understanding of depression to combat it more effectively. However, the AI lacks a concept of depression. How can it help humans combat something which isn't present in its understanding of reality?

What we *don't* want to do is form the new goal of "convince humans depression has been cured". This is a failure mode of trying to influence hidden variables you don't share.

Another thing we don't want to do is just give up and use the flawed human model. This will result in poor performance from the AI.

John Wentworth suggests taking a *translation* perspective on the ontology mismatch problem: we think of the ontologies as two different languages, and try to set up the most faithful translation we can between the two. The AI then tries to serve *translated* human values.

I think this perspective is close but not *quite* right. First note that there may not by any good translation between the two. We would like to gracefully fail in that case, rather than sticking with the best translation. (And I'd prefer that to be a natural consequence of our defined targets, rather than something extra added on.) Second, I don't think there's a strong justification for translation when we look at things from *our* perspective.

Helpee's Perspective

Imagine that we're this agent who wants to get rid of depression but can't quite understand the universe in which depression comes apart into several interacting variables.

We are capable of watching the AI and forming beliefs about whether it is eliminating depression. We are also capable of looking at the AI's design (if not its learned models), and forming beliefs about things such as *whether the AI is trying to fool us*.

We can define what it means for the AI to track latent variables we care about at the meta-level: not that it uses our exact model of reality, but that in our model, *its estimates of the latent variables track the truth.* Put statistically, we need to believe there is a (robust) correlation between the AI's estimate of the latent variable and the true value. (*Not* a correlation between the AI's estimation and *our* estimation -- that would tend to become true as a side effect, but if it were *the target* then the AI would just be trying to fool us.)

Critically, ***it is possible for us to believe that the AI tracks the truth better than we do***. IE, it will be possible for us to become confident that "the AI understands depression better than we do" and have more faith in the AI's estimation of the latent variable than our own.

To see why, imagine that you (with your current understanding of depression) were talking to a practicing, certified psychiatrist (an MD) who also has an undergraduate degree in philosophy (has a fluent understanding of philosophy of language, philosophy of science, and philosophy of mind -- and, from what you gather, has quite reasonable positions in all these things), and, on top of all that, a PhD in research psychology. This person has just recently won a Nobel prize for a new cognitive theory of depression, which has (so far as you can tell) contributed significantly to our understanding of the brain as a whole (not only depression), and furthermore, has resulted in more effective therapies and drugs for treating depression.

You might trust this person considerably more than yourself, when it comes to diagnosing depression and judging fine degrees of how depressed a person is.

To have a model which includes a latent variable is not, automatically, to believe that you have the best possible model of that latent variable. However, *our beliefs about what constitutes a more accurate way of tracking the truth are subjective* -- because hidden variables may not objectively exist, it's up to us to say what would constitute a more accurate model of them. This is part of my concept of normativity.

So, I believe a step in the right direction would be for AIs to *try to maximize values according to models which, according to human beliefs, track the things we care about well.*

This captures the ontology problem and the value amplification problem at the same time; the AI is trying to use "better" models *in precisely the sense that we care about* for both problems.

Do we need to solve the ontology mismatch problem?

It bears mentioning that we don't necessarily need to be convinced that the AI tracks the truth of variables which we care about, to be convinced that the AI enacts policies which accomplish good things. This is part of what policy approval was trying to get at, by suggesting that an AI should just be trying to follow a policy which humans expect to accomplish good things.

If we can formalize learning normativity, then we might just want an AI to be doing what is "normatively correct" according to the human normativity concept it has learned. This might involve tracking the truth of things we care about, but it might also involve casting aside unnecessary hidden variables such as "depression" and translating human values into a new ontology. That's all up to our normative concepts to specify.

Ghost Problems

Since a lot of the aim of this post is to clarify John's post, I would be remiss if I didn't deal with the ghost problem John mentioned, which I think caused some confusion. (Feel free to skip this subsection if you don't care!) This was a running example which was mentioned several times in the text, but it was first described like this:

> I don't just want to think my values are satisfied, I want them to actually be satisfied. Unfortunately, this poses a conceptual difficulty: what if I value the happiness of ghosts? I don't just want to think ghosts are happy, I want ghosts to actually be happy. What, then, should the AI do if there are no ghosts?

The first point of confusion is: it seems fine if there are no ghosts. If there aren't in fact any ghosts, we don't have to worry about them being happy or sad. So what's the problem?

I think John intended for the ghost problem to be like my depression example -- we don't just stop caring about the thing if we learn that depression isn't well-modeled as a hidden variable. In service of that, I propose the following elaboration of the ghost problem:

You, and almost everyone you know, tries to act in such a way as to make the ghosts of your ancestors happy. This idea occupies a central place in your moral framework, and plays an important role in your justification of important concepts such as honesty, diligence, and not murdering family members.

Now it's clear that there's a significant problem if you suddenly learn that ghosts don't exist, and can't be happy or sad.

Now, the *second* potential confusion is: this still poses no significant problem for alignment so long as your belief distribution includes how you would update if you learned ghosts weren't real. Yes, all the main points of your values as you would explain them to a stranger have to do with ghosts; but, your internal belief structures are perfectly capable of evaluating situations in which there are no ghosts. So long as the AI is good enough at inferring your values to understand this, there's no issue.

Again, I think John wanted to point to basic ontological problems, as with my example where humans can't quite comprehend the world where depression isn't ontologically fundamental. So let's further modify the problem:

*If you found out ghosts didn't exist, you wouldn't know what to do with yourself. Your beliefs would become incoherent, and you wouldn't trust yourself to know what is right any more. Yes, of course, you would eventually find **something** to believe, and **something** to do with yourself; but from where you are standing now, you wouldn't trust that those reactions would be good in the important sense.*

Another way to put it is that the belief in ghosts is so important that we want to invoke the value amplification problem for no-ghost scenarios: although you can be modeled as updating in a specific way upon learning there aren't any ghosts, you yourself don't trust that update, and would want to think carefully about what might constitute good and bad reasoning about a no-ghost universe. You don't want your instinctive reaction to be takes as revealing your true values; you consider it critical, in this particular case, to figure out how you *would* react if you were smarter and wiser.

As an example, here are three possible reactions to a no-ghost update:

- You might discard your explanations in terms of ghosts, but keep your values otherwise as intact as possible: continue to think lying is bad, continue to avoid murdering family members, etc.
- You might discard any values which you explained in terms of ghosts, keeping what little remains: you enjoy physical pleasure, beauty, (non-ceremonial) dancing, etc. You no longer attach much value to truth, life, family, or diligence.
- You might be more selective, preserving some things you previously justified via ghosts, but not others. For example, you might try to figure out what you would have come to value anyway, vs those places where the ghost ideology warped your values.

Simply put, you need *philosophical help* to figure out how to update.

This is the essence of the ontology mismatch problem.

Next, we consider a problem which we may encounter *even if we solve value amplification, the identifiability problem, and the ontology mismatch problem:* wireheading and human manipulation.

5. Wireheading and Human Manipulation

In my first ["stable pointers to value" post](), I distinguished between the *easy problem of wireheading* and the *hard problem of wireheading:*

- **The Easy Problem:** the wireheading problem faced by AIXI and other RL-type agents. These agents want to gain control of their own reward buttons, in order to press them continuously. This problem can be solved by evaluating the expected utility of plans/policies *inside the planning/policy optimization loop*, rather than only giving

feedback on actions actually taken (leaving the plan/policy optimization loop to evaluate based on expected reward). Daniel Dewey referred to this architecture as observation-utility maximization.
- **The Hard Problem:** even if we successfully point to human values as a thing to learn (solving the identifiability problem and the ontology mismatch problem, amongst other things), *human values are manipulable.* This introduces perverse incentives for a value-learning systems, if those systems must make decisions while operating under some remaining uncertainty about human values. Specifically, such systems are incentivised to manipulate human values.

Simply put, the easy wireheading problem is that the AI might wirehead *itself*. The hard wireheading problem is that the AI might wirehead *us,* by manipulating our values to be easier to satisfy.

This is related to the ontology mismatch problem in that poor solutions to that problem might especially incentivize a system to wirehead or otherwise manipulate humans, but is mostly an independent problem.

I see this as the *essence* of the compressed pointer problem as I described it in Embedded Agency (ie, my references to "pointers" in the green and blue sections). However, that was a bit vague and hand-wavy.

Helping Human$_{\text{current}}$

One proposal to side-step this problem is: always and only *help the current human, as of this very moment* (or perhaps a human very slightly in the past, EG the one whose light-waves are just now hitting you). This ensures that no manipulation is possible, making the point moot. The system would only ever be manipulative in the moment if that best served our values in the moment. (Or, if the system wrongly inferred such.)

This solution can clearly be dynamically inconsistent, as the AI's goal pointer keeps changing. However, it is *only as dynamically inconsistent as the human*. In some sense, this seems like the best we can do: you cannot be fully aligned with an agent who is not fully aligned with themselves. This solution represents one particular compromize we can make.

Another compromise we can make is to be fully aligned *with the human at the moment of activating the AI* (or again, the moment just before, to ensure causal separation with a margin for error). This is more reflectively stable, but will be less corrigible: to the extent humans are not aligned with our future selves, the AI might manipulate our future selves or prevent them from shutting it down, in service of earlier-self values. (Although, if the system is working correctly, this will only happen if your earlier selves would endorse it.)

Another variation is that you *myopically* train the system to *do what HCH would tell you to do at each point*. (I'll keep it a bit vague -- I'm not sure exactly what question you would want to prompt HCH with.) This type of approach subsumes not only human values, but the problem of planning to accomplish them, into the value amplification problem. This is pretty different, but I'm lumping it in the same basket for theoretical reasons which will hopefully become clear soon.

Time Travel Problems

The idea of *helping the current human,* as a way to avoid manipulation, falls apart as soon as we admit the possibility of time travel. If the agent can find a way to use time travel to manipulate the human, we get the very same problem we had in the first place. Since this might be quite valuable for the agent, it might put considerable resources toward the project. (IE, there is not necessarily a "basin of corrigibility" here, where we've specified an

agent that is aligned enough that it'll naturally try and correct flaws in its specification -- there might be mechanisms which could accomplish this, but it's not going to happen just from the basic idea of forwarding the values of the current human.)

Similarly, in my old post on this topic I describe an AI system whose value function is specified by CEV, but which finds out that there is an exact copy of itself embedded therein:

> As a concrete example, suppose that we have constructed an AI which maximizes CEV: it wants to do what an imaginary version of human society, deliberating under ideal conditions, would decide is best. Obviously, the AI cannot actually simulate such an ideal society. Instead, the AI does its best to reason about what such an ideal society would do.
>
> Now, suppose the agent figures out that there would be an exact copy of itself inside the ideal society. Perhaps the ideal society figures out that it has been constructed as a thought experiment to make decisions about the real world, so they construct a simulation of the real world in order to better understand what they will be making decisions about. Furthermore, suppose for the sake of argument that our AI can break out of the simulation and exert arbitrary control over the ideal society's decisions.
>
> Naively, it seems like what the AI will do in this situation is take control over the ideal society's deliberation, and make the CEV values as easy to satisfy as possible -- just like an RL agent modifying its utility module.

Or, with respect to my example of an agent trained to do whatever HCH would tell it to do, we might imagine what would happen if *HCH reasons about the agent in sufficient detail that its decisions influence HCH.* Then the agent might learn to manipulate HCH, to the degree that such a thing is possible.

My point is not that we should necessarily worry about these failure modes. If it comes down to it, we might be willing to assume time travel isn't possible as one of the few assumptions we need in order to argue that a system is safe -- or posit extra mechanisms to keep our AI from existing inside of CEV, or keep it from being simulated within HCH, or allow it to exist but ensure it can't manipulate those things, or what-have-you.

What *interests* me is the basic problem which applies to all of these cases. We could call in the "very hard wireheading problem":

- **The Very Hard Problem of Wireheading:** like the Hard Problem, but under the assumption that there's absolutely no way to rule out manipulation, IE, we assume manipulation is going to be possible no matter what we do.

There's probably no perfect solution to this problem. Yet, when I sit staring at the problem, I have a strong desire to "square the circle" -- like there should be *something it means* to be non-manipulatively aligned (with something that you could manipulate), something theoretically elegant and not a hack.

Probably the best thing I've seen is a proposal which I think originates from Stuart Armstrong, which is that you simply remove the manipulative causal pathways from your model before making decisions. I'm not sure how you are supposed to identify which pathways are manipulative vs non-manipulative, in order to remove them, but if you can, you get a notion of optimizing without manipulating.

This makes things very dependent on your prior -- to give an extreme case, suppose humans are perfectly manipulable; we take on whatever values the AI suggests. Then when the AI freezes the causal pathways of manipulation, its "human values" it attempts to cooperate will be a mixture of all the things it might tell humans to value, each according to its prior probability.

I had some other objections, too, EG if we remove those causal pathways, our model could get pretty weird, assigning high probability to outcomes which are improbable or even impossible. For example, suppose the AI is asked not to manipulate Sally (its creator, who it is aligned with), but in fact, Ben and Sally are equally prone to manipulation, and in the same room, so hear the same messages from the AI. The AI proceeds as if Sally is immune to manipulation (when she's not). This might involve *planning for Sally and Ben to have different reactions to an utterance* (when in fact they have exactly the same reaction). So the AI might make plans which end up making no sense in the real world.

I could say more about trying to solve the Very Hard Problem, but I suspect I've already written too much rather than too little.

Conclusion

If all of the above are sub-problems of the pointers problem, what is the pointers problem itself? Arguably, it's the whole outer alignment problem. I don't want to view it quite that way, though. I think it is more like a particular view on the outer alignment problem, *with an emphasis on "pointing"*: the part of the problem that's about "What do we even mean by alignment? How can we robustly point an AI at external things which are ontologically questionable? How do we give feedback about what we mean, without incentivizing a system to wirehead or manipulate us? How can we optimize things which live in the human ontology, without incentivizing a system to delude us?"

There's a small element of inner alignment to this, as well. Although an RL agent such as AIXI will want to wirehead if it forms an "accurate" model of how it gets reward, we can also see this as only one model consistent with the data, another being that reward is actually coming from task achievement (IE, the AI could internalize the intended values). Although this model will usually have at least slightly worse predictive accuracy, we can counterbalance that with process-level feedback which tells the system that's a better way of thinking about it. (Alternatively, with strong priors which favor the right sorts of interpretations.) This is inner alignment in the sense of getting the system to *think in the right way* rather than *act in the right way*, to avoid a later treacherous turn. (However, not in the sense of avoiding inner optimizers.)

Similarly, human manipulation could be avoided not by solving the incentive problem, but rather, by giving feedback to the effect that manipulation rests on an incorrect interpretation of the goal. Similar feedback-about-how-to-think-about-things could address the ontology mismatch problem and the value amplification problem.

In order to use this sort of solution, the AI system needs to think of *everything* as a proxy; no feedback is taken as a gold standard for values. This is similar to Eliezer's approach to wireheading, Goodhart, and manipulation in CFAI (see especially section 5, especially 5.5), although I don't think that document contains enough to make the idea really work.

Four Motivations for Learning Normativity

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

I have been pretty satisfied with my desiderata for learning normativity, but I *haven't* been very satisfied with my explanation of why exactly these desiderata are important. I have a sense that it's not just a grab-bag of cool stuff; something about trying to do all those things at once points at something important.

What follows are four different elevator pitches, which tell different stories about how it all hangs together. Desiderata are bolded.

Conceptual Difficulties with Outer Alignment

The classic problem of outer alignment is that we have **no perfect loss function,** so we can't just go optimize. The problem can be understood by thinking about Goodhart and how optimization amplifies. The classic response to this is value uncertainty and value learning, but wireheading, human manipulation, and no-free-lunch results make it seem plausible that we have the same problem one level up: we still don't know how to specify a perfect loss function for what we care about, and imperfect loss functions can still create big problems.

So, just like value-learning tackles the initial problem head-on by suggesting we manage our uncertainty about values and gain knowledge over time, **learning at all levels** suggests that we tackle the meta-problem directly, explicitly representing the fact that we don't have a perfectly good loss function at *any* level, but can manage that uncertainty and learn-to-learn over time.

Humans can only give explicit feedback at so many meta-levels, so **between-level sharing** is critical for any meaningful learning to take place at higher meta-levels. Otherwise, higher meta-levels remain highly uncertain, which itself makes learning at lower levels almost impossible (since you can't learn if you have high uncertainty about learning-to-learn).

A consequence of having no perfect loss function is **no perfect feedback;** no evidence about what the system should do can be considered absolute. A helpful measure for coping with this is to support **uncertain feedback**, so that humans can represent their uncertainty when they provide feedback. Ultimately, though, humans can have systematic biases which require **reinterpretable feedback** to untangle.

Even with all these tools, some forms of feedback would be difficult or impossible to articulate without **process-level feedback:** the ability to tell the system that specific patterns of thinking are good or bad, without needing to unpack those judgements in terms of consequences. To be meaningful, this requires **whole-process feedback:** we need to judge thoughts by their entire chain of origination. (This is technically challenging, because the easiest way to implement process-level feedback is to create

a separate meta-level which oversees the rest of the system; but then this meta-level would not itself be subject to oversight.)

Finally, because it's not feasible for humans to approve every thought process by hand, it's critical to have **learned generalization of process-level feedback.** This doesn't sound like a big request, but is technically challenging when coupled with the other desiderata.

Recovering from Human Error

A different place to start is to motivate everything from *designing a system to recover from errors which humans introduce*.

The ability to learn when there's **no perfect feedback** represents a desire to recover from input errors. **Uncertain feedback** and **reinterpretable feedback** follow from this as before.

We can't avoid *all* assumptions, but specifying a loss function is one area where we seem to assume much more than we bargain for; what we mean is something like "good things are roughly in this direction", but what we get is more like "good things are precisely this". We want to avoid making this type of mistake, hence **no perfect loss function**. **Learning at all levels** ensures that we can correct this type of mistake wherever it occurs. **Between-level sharing** is needed in order to get any traction with all-level learning.

Whole-process feedback can now be motivated by the desire to learn a *whole new way of doing things*, so that nothing is locked in by architectural mistakes. This of course implies **process-level feedback**.

Learned generalization of feedback can be seen as a desire to pre-empt human error correction; learning the patterns of errors humans correct, so as to systematically avoid those sorts of things in the future.

We Need a Theory of Process-Level Feedback

We could also motivate things primarily through the desire to facilitate **process-level feedback**. Process-level feedback is obviously critical for inner alignment; we want to be able to tell a system to avoid specific kinds of hypotheses (which contain inner optimizers). However, although we can apply penalties to neural nets or such things, we lack a general theory of process-level feedback that's as rigorous as theories we have for other forms of learning. I think it's probably a good idea to develop such a theory.

In addition to inner alignment, process-level feedback could be quite beneficial to outer-alignment problems such as corrigibility, non-manipulation, and non-wireheading. As I argued in another section, we can often point out that something is wrong without being able to give a utility function which represents what we want. So process-level feedback just seems like a good tool to have when training a system, and perhaps a necessary one.

You might think process-level feedback is easy to theoretically model. In a Bayesian setting, we can simply examine hypotheses and knock out the bad ones (updating on not-this-hypothesis). However, this is an incredibly weak model of process-level feedback, because there is no **learned generalization of process-level feedback!** Learned generalization is important, because humans can't be expected to give feedback on *each individual hypothesis,* telling the system whether it's OK or full of inner optimizers. (If we develop a technology that can automatically do this, great; but otherwise, we need to solve it as a learning problem.)

The next-most-naive model is a two-level system where you have object-level hypotheses which predict data, and meta-level hypotheses which predict which object-level hypotheses are benign/malign. Humans provide malign/benign feedback about first-level hypotheses, and second-level hypotheses generalize this information. This proposal is not very good, because now there's no way to provide process-level feedback about second-level hypotheses; but absent any justification to the contrary, these are just as often malign. This illustrates the need for **whole-process feedback**.

This suggests a version of **learning at all levels:** if the process-level feedback at one level can be regarded as data for the next level, everything can be generalizable. However, **between-level sharing** is necessary, since patterns we want to avoid at one level will very often be patterns we want to avoid at all levels.

In this story, **no perfect feedback** and **no perfect loss function** are less important.

Generalizing Learning Theory

Another way to motivate things is through the purely theoretical desire to push learning theory as far as possible. Logical induction can be thought of as pure learning-theory progress, in which a very broad bounded-regret property was discovered which implied many other desirable properties. In particular, it generalized to non-sequential non-realizable settings, where Solomonoff induction only dealt with sequential prediction in realizable settings. Also, it dealt with a form of bounded rationality where Solomonoff induction only dealt with unbounded rationality.

So, why not try to push the boundaries of learning theory further?

As I discussed in the previous section, we can think of **process-level feedback** as feedback directly on hypotheses. **Whole-process feedback** ensures we focus on the interesting part of the problem, making hypotheses judge each other and themselves, rather than getting a boring partial solution by stratifying a system into seperate levels.

The **learned generalization** problem can be understood better through a learning-theory lense: the problem is that Bayesian setups like Solomonoff induction offer *no regret bounds for updates about hypotheses*, due to focusing exclusively on predicting information about sense-data. So although Bayesianism supports updates on *any* proposition, this does *not* mean we get the nice learning-theoretic guarantees with respect to all such updates. This seems like a pretty big hole in Bayesian learning theory.

So we want *learned generalization of as many update types as possible*, by which we mean that we want loss bounds on as many different types of feedback as possible.

Uncertain feedback is just another generalized feedback type for us to explore.

Reinterpretable feedback is a more radical suggestion. We can motivate this through a desire for a theory of meta-learning: how do we learn to learn, in the broadest possible way? This motivates thinking about **no perfect feedback** and **no perfect loss function** scenarios.

Learning at all levels could be motivated from the needs of process-level feedback, as in the previous section, or from the nature of the no-perfect-loss-function scenario, as in the first section. **Between-level sharing** is motivated from it as usual.