AI Alignment Research Overview

Jacob Steinhardt

Introduction

This document gives an overview of different areas of technical work that seem necessary, or at least desirable, for creating safe and aligned AI systems. The focus is on safety and alignment of powerful AI systems, i.e. systems that may exceed human capabilities in a broad variety of domains, and which likely act on a large scale. Correspondingly, there is an emphasis on approaches that seem scalable to such systems.

By “aligned”, I mean that the actions it pursues move the world towards states that humans want, and away from states that humans don’t want. Some issues with this definition are that different humans might have different preferences (I will mostly ignore this issue), and that there are differences between stated preferences, “revealed” preferences as implied by actions, and preferences that one endorses upon reflection (I won’t ignore this issue).

I think it is quite plausible that some topics are missing, and I welcome comments to that regard. My goal is to outline a critical mass of topics in enough detail that someone with knowledge of ML and some limited familiarity with AI alignment as an area[1] would have a collection of promising research directions, a mechanistic understanding of why they are promising, and some pointers for what work on them might look like.

To that end, below I outline four broad categories of technical work: **technical alignment** (the overcoming of conceptual or engineering issues needed to create aligned AI), **detecting failures** (the development of tools for proactively assessing the safety/alignment of a system or approach), **methodological understanding** (best practices backed up by experience), and **system-building** (how to tie together the three preceding categories in the context of many engineers working on a large system). These are described in more detail in the next section.

In each section I give examples of problems we might want to solve. I imagine these in the context of future powerful AI systems, which means that most of the concrete scenarios are speculative, vague, and likely incorrect if interpreted as a prediction about the future. If I were to give the strongest justification for the research topics below, I would instead focus on near-future and existing systems, which already exhibit many of the issues I discuss. Nevertheless, I think this imaginative exercise can be helpful both for stimulating research and for keeping the focus on scalable solutions.

Caveats. I found it difficult to write a research overview of a field as nascent as AI alignment, as anything I could write sounded either too authoritative relative to my confidence, or so full of

[1] E.g. from Stuart Russell’s recent book on the topic.

caveats and qualifications as to be unreadable. I settled for eliding many of the qualifications and providing this single caveat up front: that this document reflects an imperfect snapshot of my current thinking, that it expresses many ideas more sloppily than I would usually feel comfortable putting into writing, and that I hope readers will forgive this sloppiness in the service of saying *something* about a topic that I feel is important.

This document is not meant to be a description of *my personal interests*, but rather of potentially promising topics within a field I care about. My own interests are neither a subset nor superset of the topics in this document, although there is high overlap. Even confined to AI alignment, this document is out-of-date and omits some of my recent thinking on economic aspects of ML.

Finally, I make a number of claims below about what research directions I think are promising or un-promising. Some of these claims are likely wrong, and I could even imagine changing my mind after 1 hour of conversation with the right person. I decided that this document would be more informative and readable if I gave my unfiltered take (rather than only opinions I thought I would likely defend upon consideration), but the flip side is that if you think I'm wrong about something, you should let me know!

Categories of technical work

In this document, I will discuss four broad categories of technical work:

Technical alignment problem. Research on the "technical alignment problem" either addresses conceptual obstacles to making AI aligned with humans (e.g. robustness, reward mis-specification), or creates tools and frameworks that aid in making AI aligned (e.g. scalable reward generation).

Detecting failures in advance. Independently of having solved various alignment problems, we would like to have ways of probing systems / blueprints of systems to know whether they are likely to be safe. Example topics include interpretability, red-teaming, or accumulating checklists of failure modes to watch out for.

Methodological understanding. There is relatively little agreement or first-hand knowledge of how to make systems aligned or safe, and even less about which methods for doing so will scale to very powerful AI systems. I am personally skeptical of our ability to get alignment right based on purely abstract arguments without also having a lot of methodological experience, which is why I think work in this category is important. An example of a methodology-focused document is Martin Zinkevich's Rules of Reliable ML, which addresses reliability of existing large systems.

System-building. It is possible that building powerful AI will involve a large engineering effort (say, 100+ engineers, 300k+ lines of code[2]). In this case we need a framework for putting many components together in a safe way.

Each category has its own section (and sometimes subsections for more specific sub-areas) where I describe why we want to solve the problem(s) in that category, different things one could mean by “solving the problem” and which ones I am most excited about, some short descriptions of concrete research questions, and some brief pointers to related work.

One might also care about forecasting, AI policy, or higher-level questions around field-building and culture. These are all important but not addressed in this document.

Technical alignment problem

We would ideally like to build AI that acts according to some specification of human values, and that is robust both to errors in the specification and to events in the world. To achieve this robustness, the system likely needs to represent uncertainty about both its understanding of human values and its beliefs about the world, and to act appropriately in the face of this uncertainty to avoid any catastrophic events.

I split the technical alignment problem correspondingly into four sub-categories:

Scalable reward generation. Powerful AI systems will potentially have to make decisions in situations that are foreign to humans or otherwise difficult to evaluate---for instance, on scales far outside human experience, or involving subtle but important downstream consequences. Since modern ML systems are primarily trained through human-labeled training data (or more generally, human-generated reward functions), this presents an obstacle to specifying which decisions are good in these situations. Scalable reward generation seeks to build processes for generating a good reward function.

Reward learning. Many autonomous agents seek to maximize the expected value of some reward function (or more broadly, to move towards some specified goal state / set of states). Optimizing against the reward function in this way can cause even slight errors in the reward to lead to large errors in behavior--typically, increased reward will be well-correlated with human-desirability for a while, but will become anti-correlated after a point. Reward learning seeks to reason about differences between the observed (proxy) reward and the true reward, and to converge to the true reward over time.

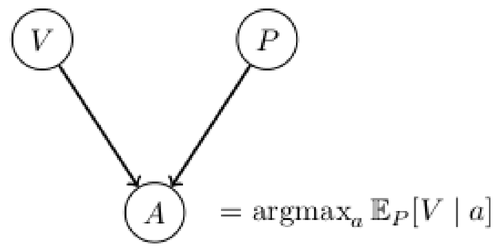
[2] Conversely, maybe it will only involve 10 engineers at DeepMind or OpenAI doing some end-to-end training and almost all of the engineering effort is in backend scaling of simulators and distributed GPU usage, rather than on the AI system itself. This seems more likely under “short timelines” (powerful AI arriving in the next ~5 years), so system-building may be less (but still somewhat) useful in these scenarios.

Out-of-distribution robustness is the problem of getting systems to behave well on inputs that are very different from their training data. This might be done by a combination of transfer learning (so the system works well in a broader variety of situations) and having more uncertainty in the face of unfamiliar/atypical inputs (so the system can at least notice where it is likely to not do well).

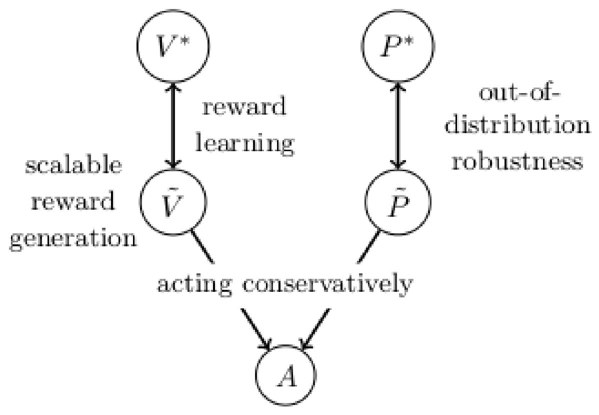
Acting conservatively. Safe outcomes are more likely if systems can notice situations where it is unclear how to act, and either avoid encountering them, take actions that reduce the uncertainty, or take actions that are robustly good. This would, for instance, allow us to specify an ambiguous reward function that the system could clarify as needed, rather than having to think about every possible case up-front.

Acting conservatively interfaces with reward learning and out-of-distribution robustness, as the latter two focus on noticing uncertainty while the former focuses on what to do *given* the uncertainty. Unfortunately, current methods for constructing uncertainty estimates seem inadequate to drive such decisions, and even given a good uncertainty estimate little work has been done on how the system should use it to shape its actions.

A toy framework. Conceptually, it may be useful to think in terms of the standard rational agent model, where an agent has a value function or utility function V, and beliefs P, and then takes actions A that maximize the expected value of V under P (conditioned on the action A). Failures of alignment could come from incorrect beliefs P, or a value function V that does not lead to what humans want. Out-of-distribution robustness seeks to avoid or notice problems with P, while scalable reward generation seeks to produce accurate information about some value function V that is aligned with humans. Reward learning seeks to correct for inaccuracies in the reward generation process, as well as the likely limited amount of total data about rewards. Finally, acting conservatively takes into account the additional uncertainty due to acting out-of-distribution and having a learned reward function, and seeks to choose actions in a correspondingly conservative manner.



(Standard) Rational Agent Model



An Aligned Agent?

In an RL setting where we take actions via a learned policy, we can tell the same story but with a slightly modified diagram. Instead of an action A we have a learned policy θ, and instead of P* and \tilde{P} denoting beliefs, they denote distributions over environments (P* is the true on-policy environment at deployment time, while \tilde{P} is the distribution of training environments).

Other topics. Beyond the topics above, the problem of **counterfactual reasoning** cuts across multiple categories and seems worth studying on its own. There may be other important categories of technical work as well.

Out-of-distribution Robustness

The Problem

Out-of-distribution robustness is the problem of getting systems to behave well on inputs that are very different from their training data. This might be done by a combination of transfer learning (so the system works well in a broader variety of situations) and having more uncertainty in the face of unfamiliar/atypical inputs (so the system can at least notice where it is likely to not do well).

Why We Want to Solve It

Powerful AI will likely take actions that bring it into regimes very different from what it saw at training time. Reasons for this include[3]:

1. Creating new technologies enabling capabilities that didn't exist at training time.
2. Going from being less capable than humans to being more capable than humans.
3. Going from actions/consequences that humans can reasonably understand/interpret to ones that they cannot.
4. Potential competition with other AIs.
    a. Additionally, feedback effects with other AIs. If one AI does something really weird, it could potentially cause a cascade.
5. More generally, the world tends to change. Maybe there is a natural disaster, or a viral youtube video with the same name as a government organization.

Without out-of-distribution robustness, each of the above changes could lead to failures:

1. An AI fails to predict bad consequences of some technology (doesn't realize that some useful but dangerous synthetic organism will lead to a bad outcome), or predicts a consequence but incorrectly thinks it is good (doesn't realize that genetically engineering humans to be happy all the time is not universally endorsed).
2. When an AI is less powerful than humans, (A) "do what humans want" and (B) "cause the reward input by humans to be high" lead to the same actions, so the AI could equally

[3] There are much more prosaic examples already affecting current systems. Here I intentionally try to imagine what issues might befall a much more sophisticated AI, at the cost of my examples being more speculative.

well generalize according to A or B. When an AI is more powerful than humans, if it generalized according to B it would seize control over the reward input.
3. Similarly to the above, the AI could (A) take actions that humans would endorse, or (B) take actions whose consequences appear superficially good to humans. See discussion starting at “Strategy #2” in this article.
4. In a multi-polar world where there are multiple powerful AIs competing with each other, each AI has an incentive to drive other AIs out-of-distribution (assuming that the other AIs perform poorly out-of-distribution). Additionally, while we might hope that if one AI does something erratic the other AIs would respond to neutralize the erraticness, it could instead lead to a feedback loop where all of the AIs behave erratically due to being out-of-distribution.
5. There are probably additional failures modes and it seems desirable in the abstract for AI to be robust to changes in distribution.

What A Solution Looks Like

Ideally, a system solving the out-of-distribution robustness problem would:

- Make good predictions in novel situations and anticipate potential novel consequences of actions (as in the synthetic organism example).
- Notice when multiple generalizations are supported by the data and act[4] to safely disambiguate between them (as in the A vs. B scenarios above).
- Be robust to strategic or adversarial inputs to the system.

It is possible that we can make do without some of these, or that some of these are impossible or infeasible in full generality. My current guess is that each of these is in expectation quite useful for improving safety and alignment, and that at least some important subclass of each problem can be solved.

Traditionally, terms like “distributional shift”, “transfer learning”, and “domain adaptation” point to a cluster of problems, some of which seem safety-relevant and some of which do not. I am primarily interested in **detecting** changes in distribution that might compromise performance, and devising training methods that make systems **robust** to a larger family of inputs. I am less interested in methods that decrease the cost of training in new domains (e.g. word vectors, image vectors, the retro contest), since these primarily make it cheaper to train new AI systems, rather than preventing the failure of a deployed system. I am also less interested in methods that transfer a capability from one situation to a new situation (e.g. sim2real), although there are exceptions---sim2real could potentially be used to harden a system across a wide range of inputs in simulation, which could then be transferred to the real world. This is useful because it is about *transferring robustness* (which may be easier to achieve in simulation) rather than *transferring capabilities*.

There are some methods that impart robustness to a specific type of distributional change (such as “strategic agents” in the case of algorithmic game theory, or “L-infinity perturbations” in the

[4] The “act” part of this would fall under acting conservatively rather than out-of-distribution robustness.

case of some adversarial examples research). I am excited about these methods as a starting point, but we should keep in mind the requirement of scalability---we eventually want to scale to much broader families of distributional changes and have arguments for having achieved sufficient robustness for the system to be safe. Relatedly, one proposed approach is to explicitly train the system on a wide variety of situations in the hope that this will impart robustness. This approach may work, but it should be accompanied by arguments for why it provides sufficient safety in unanticipated situations, and we should develop methods to empirically evaluate whether this approach can yield systems that generalize to radically new situations.

Possible Directions

For **detecting** changes, some concrete problems (not especially selected) are unsupervised risk estimation, detecting out-of-domain inputs, and change-point detection. More broadly, it seems useful to have systems that have good estimates of uncertainty that continue to be valid even on out-of-distribution examples. For any of these tasks, we wish to accomplish them with high reliability, little to no human input, and in a way that scales to large, powerful, and complex systems.

For being **robust** to changes, some concrete problems are unsupervised transfer learning, as well as certain types of robust optimization, and certain work on adversarial examples. Research should aim to eventually scale to very broad families of perturbations--see e.g. these papers on stochastic perturbations and these two on adversarial perturbations, which are a starting point in this direction. Causal inference is also relevant, and algorithmic game theory may help as well (insofar as it outlines a specific class of perturbations and methods for being robust to them). Note however that many approaches are limited to either specific narrow forms of robustness, or can only impart robustness to very small perturbations. Different researchers will have differing intuitions about which of these approaches can scale to richer forms of robustness, but it is important to keep this in mind as the end goal. Similarly, empirical work should make sure that experiments are testing for scalability by including large perturbations as well as perturbation types that were not explicitly anticipated at training time.

For both detection and robustness, many difficulties disappear when the model is well-specified (meaning that nature's distribution lies within the model family), but it is not clear that we can expect to have well-specified models as systems scale up. Approaches should either grapple with model mis-specification (either theoretically or by running experiments that test behavior under mis-specification), or argue that a particular model family should be scalably well-specified and show how to leverage this.

My Personal Take (as of May 2019)

For detecting changes, I currently feel that building appropriate ensembles of models and looking at when they start to disagree with each other could be a good and robust way of detecting changes. However, how the ensemble is built is important---I do not think that simply training multiple models with different random initialization will provide enough diversity to do a good job.

For robustness, I am excited by my current line of work on robustness to training data corruptions in high dimensions, which designs procedures handling perturbations whose size remains constant even as the dimension grows (in contrast, most older work can only handle fairly small perturbations when the dimension is large). While this focuses on training-time robustness rather than test-time robustness, the two seem related to me. For instance, under distributional shift we can think of the shifted distribution as the “real” distribution and the original training distribution as a “corrupted” version of this real distribution (see my class notes or this recent paper).

I am also excited by methodological work that measures robustness across a wide variety of perturbations and attempts to understand when we should expect to get robustness to unforeseen changes. This includes some of my own work on transfer of adversarial robustness, as well as work on measuring corruption robustness (both linked above).

Additional References
These are not meant to be exhaustive or even representative, but include papers that are relevant to the topics above (some already linked in-text):

Deep Anomaly Detection using Geometric Transformations uses one-class learning to detect out-of-domain inputs.
Unsupervised Risk Estimation Using Only Conditional Independence Structure shows that the “three-view” assumption can be used to estimate the performance of ML models on new input distributions from only unlabeled data.
Domain Adaptation from Coupled Subspaces uses a two-view assumption to do unsupervised transfer learning.
Deep Anomaly Detection with Outlier Exposure seeks to improve outlier detection by training on auxiliary data.

Reward Learning

The Problem
Many autonomous agents seek to maximize the expected value of some reward function (or to move towards some set of specified goal states). Unfortunately, it is difficult to specify a reward function by hand such that increasing the reward function leads to human-desirable outcomes. (Typically, increased reward will be well-correlated with human-desirability for a while, but will become anti-correlated after a point.) Reward learning is the problem of constructing processes for generating a good reward function from potentially noisy and biased information about the true reward.[5]

[5] We may wish to specify something other than a reward function, in which case a more appropriate term might be “goal learning” or “specification learning”. I will use reward learning for concreteness, but the

Another way of putting this is that reward learning is about receiving (information about) rewards and faithfully incorporating them into the systems' actions, in a way that is ideally robust to getting the answer slightly wrong and is trying hard to "do what the specifier meant" rather than "optimize against the literal content of the specification".

Why We Want To Solve It
Most easy-to-specify rewards lead to bad outcomes if maximized. For instance, “take actions that are rated highly by humans” incentivizes not just doing things that are good for humans, but also deceiving humans into thinking the action is good, as well as physically hijacking the rating process.

To give a sense for how hard hand-specifying good rewards is, let’s consider a limiting case. Imagine that we have a system that is in charge of optimizing the global economy. Because it produces so much economic value, we can devote many resources to providing a faithful reward signal---for instance, we might have ~1-10 million people providing rewards to the system many times a day based on detailed rubrics. Here are some ways that an AI system would be incentivized to exploit this reward signal:

- Trying to improve people’s moods shortly before they provide the reward signal (or even trying to bribe people, although that carries the risk of a very negative reward from people who don’t approve of bribes).
- Suppose that people in city A are more skeptical of the system and harder to satisfy than people in city B. Increasing economic prosperity of B relative to A might lead to higher reward (because it would likely increase the population of B relative to A, and thus increase the fraction of non-skeptical reward providers).
- The system would also be incentivized to hack the devices raters use to enter rewards, create Potemkin villages giving the appearance that the world is more prosperous, etc. These behaviors would be easily detected until the point that the system is powerful enough to consistently fool humans, so we might expect a tipping point where these behaviors only appear once the system is sufficiently powerful.

Paul Christiano talks about additional failures of alignment here. Paul also wrote a post describing how alignment failures might actually arise and cause catastrophes in practice.

A solution to reward learning should avoid the above failure modes. The solution likely needs a way to distinguish the actual preferences of humans in the environment from simple proxies such as “the number input into this terminal” that correlate with human preferences but can be gamed (see the above example of the global economic system). Some challenges are that “humans” and “preferences” are fuzzy concepts that seem difficult to specify by hand, humans are inconsistent in many ways, and their values probably fluctuate over time (and might not

---

reader should understand that rewards are one of several ways of specifying the desired behavior of a system.

even be defined regarding far-future events). Moreover, there are differences between “verbally stated preferences”, “revealed preferences”, “urges”, etc. A final issue is that different humans might have contradictory values, although I expect this to be only a small part of the problem since the values of two randomly sampled people are much closer to each other than to the value functions used in modern AI systems.

We might not need to fully capture all aspects of human values (which might not even be a well-defined concept) but at minimum need to capture ideas such as “humans should still have sovereignty over what is happening”. Paul Christiano discusses this here.

In the economic scenario above, we might hope that the system doing reward learning would understand that (Y) “the reward the human enters into the system” is a noisy proxy of (Z) “the extent to which their preferences are satisfied”, and that influencing their mood affects the proxy Y but not the true value Z, and is thus undesirable because it destroys information. In circumstances where the system gains new capabilities (e.g., it figures out how to affect which political parties are in control), and it doesn’t have enough information to know whether a given use of those capabilities would be endorsed, it should notice this and seek additional information (this is closely related to acting conservatively and out-of-distribution robustness, as it is essentially about detecting and responding to out-of-distribution inputs).

What A Solution Looks Like

The most obvious way to learn rewards in an ML framework is to generate a lot of data about what the reward should be, and build a model based on that. However, the data about the rewards will inevitably be a proxy for what we actually care about (since we don’t have access to a human’s “true reward function”, which is likely not even a coherent concept). We thus would ideally like a system that can learn about complex rewards that cannot be hand-coded. We desire that the system:

1. Understands what errors tend to occur when specifying the reward function
2. Takes actions in light of this understanding, reasoning about its reward signal being a proxy for the true reward
3. Reliably seeks out information about the true reward and converges to it over time.
4. Is robust to the fact that the information it seeks out will also be a proxy.
5. Avoids doing anything catastrophic from the perspective of the true reward, even in cases where different alternative rewards conflict strongly on recommended actions.

This leaves aside the question of how to provide information about the reward function in the first place, which is discussed in Scalable Reward Generation.

In addition to the above, our solution should be scalable--there should be minimal hand-coding involved, and it should work even for fairly sophisticated agents that can do e.g. planning and meta-learning[6].

The more correct the information about the reward is, the less we need to rely on #1 through #5. Indeed, one could potentially obviate #1-#5 entirely by solving reward generation completely. Relying entirely on this feels brittle to me, but I know other people such as Paul Christiano who are more optimistic about this (note that Paul's approach, based on amplification, involves a lot of machinery to ensure that the reward continues to be sufficiently correct as the system becomes more capable; more on this below under "Alternative Approaches"). For most of the text below I will assume that we do need to care substantially about #1-#5 above.

A common framework for reward learning is inverse reinforcement learning, where we observe actions of an agent and impute the reward function that leads to those actions. However, this approach must grapple with a number of issues such as those outlined in Model Mis-Specification and Inverse Reinforcement Learning.

Note that inverse reinforcement learning sometimes refers instead to a form of imitation learning, where the goal is to imitate the trajectory of the agent. This form of inverse reinforcement learning does not seek to learn the underlying reward function; it instead posits a reward function as a nuisance variable in a model that predicts trajectories (note though that imitation learning may be useful for amplification, discussed below).

Some work tries to bypass some of the difficulties of inverse reinforcement learning by using a more direct signal of human preferences (such as directly asking humans what they want, e.g. via pairwise comparisons [refs. 1,2,3]). I like this because it makes less stringent assumptions about the data--it only assumes that people can say what they prefer, rather than that they are (almost) perfectly rational agents. However, I expect this signal to still have errors, so I am excited about scaling to more complex tasks where these errors are likely to occur so that we can study them (e.g. tasks that are harder for humans to specify by hand, where there are more opportunities for reward hacking, where humans have limited intuition about what the optimal policy looks like, where out-of-distribution robustness is a larger issue, etc.).

For both inverse reinforcement learning and learning from preferences, we want to explicitly account for the fact that the data is likely noisy/biased (per #1-#5 above). One line of thought is to maintain uncertainty about rewards, which can help us to be more robust to incorrect reward specification. This idea seems fruitful in principle, but it is important to be careful about what we mean by uncertainty. Even with infinite data, the directly observable rewards are only a proxy for

[6] So for instance, solutions that rely on the fact that gradient descent updates don't lead to sudden changes are not scalable, because the meta-learned agent likely makes use of new data in a more sophisticated way.

the true reward, and so statistical uncertainty[7] does not suffice as a measure in the context of reward mis-specification (since it goes to zero with infinite data). Accounting for mismatch between the true reward function and whatever model family we use, or for errors from distributional shift, also is not sufficient, since the observed reward can again be a proxy even if the model family is correct and there is no distributional shift[8]. We thus need something more nuanced in addition to these uncertainty measures---perhaps a model of how rewards tend to be mis-specified that was obtained via empirical investigation (although we might worry about the scalability of this). I discuss the need for more nuanced forms of uncertainty in more detail in "Acting Conservatively".

Possible Directions

Below I discuss some possible concrete directions for approaching #1-#5 above. I see much of this space as uncharted with much of the question being how to approach the problem in the first place. There are also approaches (discussed next) that question this entire framework.

On #1 (characterizing errors in specification): There is some work investigating ways in which the reward signal might be biased or otherwise incorrect. For instance, D. Sculley, Leon Bottou, and their collaborators identify feedback loops as one source of this (see here, here, and here). I suspect that industry engineers at large companies have experience handling various reward specification issues (for instance, Google and probably other companies already pay people to fill out detailed rubrics rating search results etc., and Martin Zinkevich discusses it in rule #38 here). Distilling this knowledge would be valuable.

One could also perform human subject experiments on soliciting rewards or preferences to gather information about the sorts of systematic errors that humans make. I expect neither this nor distilling industry knowledge to yield a comprehensive list of potential failures of reward specification, but I still think it would be valuable by virtue of providing empirically-grounded models to work with for #2 (right now the models used for #2 come from the armchair), as well as providing a concrete set of tasks to work on for reward learning more generally.

[7] When using the term statistical uncertainty, I assume that we have some likelihood function $p(x \mid \theta)$ for the observed data x conditional on the parameters θ of the reward function. In a Bayesian framework the statistical uncertainty is $p(\theta \mid x)$, while in a frequentist framework it is whatever confidence region we construct for θ.

[8] Specifically, suppose the true reward function is R*, and we have a family of reward functions {R_{theta}}. If R* = R_{theta} for some theta, then the model family is correct. But it could be that our method for defining R* is incorrect, and we accidentally define the reward R' (which might even also be in the model family). An example of this would be if we assume that people's desires are given by their revealed preferences (or their preferences if you ask them questions and give them 5 minutes to think about it) rather than the desires they would endorse upon reflection. It could be that the model family contains the reward functions corresponding to both cases, but you still have to correctly specify how to back out the reward function from the data.

Some existing work discusses errors from mis-specified objective spaces as well as myopia and time-inconsistency. See here for a brief overview of possible sources of systematic error and here for a model and empirical study of how humans infer preferences of other humans.

On #2 (reasoning about errors): There are two aspects of #2---(A) representing our uncertainty about the true reward, and (B) acting appropriately conservatively given that uncertainty. As noted in #1, we do not currently have good empirically grounded models for addressing A, which somewhat limits our ability to work on B. Perhaps the closest to A is work from the cog sci direction looking at ways in which people tend to behave irrationally (see references above from #1), and inverse reward design also proposes an abstract model of reward uncertainty based on assuming that the proxy reward function imputes nearly-optimal policies on the distribution it was designed for. For B, robust optimization can provide conservative decision rules under Knightian uncertainty about the reward function (e.g. we do not know the true reward and want to do well under all possible rewards, so we apply some minimax decision rule). These conservative rules that take into account worst-case uncertainty also tend to "optimize to the boundary of the space" less than non-conservative rules. There is also a large literature on risk-averse planning (example paper) although I am not sure whether the types of robustness sought there map correctly onto the reward uncertainty scenario.

On #3 (seeking out/converging to true reward): Paul Christiano discusses one formalization of this here (see "Application to AI control: measuring success"), although in a different context where the assumption is that the true reward is correctly specified in principle, but very difficult to evaluate. His idea is that the agent might then be incentivized to seek out reliable proxies in the environment for the expensive-to-evaluate reward, and query those (e.g. ask people clarification questions, and try to pose the questions such that the answer is as informative as possible); it would then fall back to the expensive-to-evaluate true reward only when necessary. Shah et al. consider how preferences implicit in the state of the world could provide such information.

It is less clear how to do this if we only ever get a proxy reward; one might hope to find somewhat "independent" proxies that are wrong in complementary ways, for instance asking people questions phrased in a number of different ways, or giving them different context when asking the question. Such independent proxies could be an approach to #4 (realizing the information sought out could also be a proxy). The idea of independent proxies has been explored elsewhere in the context of multi-view learning (see my paper on unsupervised risk estimation and the references within).

The Off-Switch Game provides a simple Bayesian model in which an agent would seek out information about the true reward, and Bajcsy et al. provide a model of informative feedback in human-robot interaction.

On #5 (avoiding anything catastrophic): One way in which different reward functions could strongly conflict is when an agent gains a new ability that allows it to hack a proxy reward at the expense of the true reward, leading to a so-called “treacherous turn”. I am interested in constructing plausible environments where something like a treacherous turn might occur (this might require the agent to be doing some sort of planning). So far most concrete examples of treacherous turns are in toy environments, e.g. this or this.

Alternative Approaches

Some work questions the need/desirability of an agent that maximizes expected reward, and proposes alternative criteria for agents to choose actions. One approach is imitation learning--if an agent simply seeks to imitate another agent, then it seems unlikely to perform problematic optimization as long as the original agent does not.

Another approach is quantilization, which plays actions in the qth quantile of reward rather than the maximum reward (this quantile must be defined relative to some base distribution; the naive approach would be a uniform distribution over actions, while a better approach might be to take a base distribution arising from a stochastic reference policy such as imitation of a human expert). This paper implements something roughly along these lines.

If successful, these approaches could obviate the need for some or all of #1-6 above, but we should attend to their scalability. Quantilization in particular may not work in high-dimensional action spaces, since the qth quantile of actions is not much better than a random action unless q is exponentially small; using a reference policy rather than a uniformly random policy as the base distribution may alleviate this, but I suspect it still runs into difficulties eventually. Similarly, straightforward imitation learning has the issue that the resulting agent is usually not better than the agent it imitates (although it may be faster and cheaper). Amplification, which we discuss next, provides one way of getting past this obstacle.

Amplification is a strategy for creating a reward that is correctly specified “enough” that we can train a system against it while mostly ignoring #2-#6.[9] The basic intuition behind amplification is that how correct the reward function needs to be depends on how capable the system optimizing the reward function is--with a weak optimizer you can get by with a somewhat correct reward whereas a very strong optimizer requires an almost perfectly correct reward. Under this perspective, what we want is a way for the correctness of the reward function to scale with the capabilities of AI systems. One way of doing this is to use AI systems themselves (in conjunction with humans) to specify the reward function. An obstacle to this is how to ensure this system actually specifies the human’s preferences, rather than something based on the idiosyncracies of the AI system that the human is working with. I am fairly worried about this obstacle, although Paul has written about strategies for trying to avoid this by decomposing tasks into human-answerable questions (see this blog post). Paul has more generally written extensively about amplification on his blog.

[9] This reward may be expensive to evaluate; the amplification approach is willing to accept this as long as “expensive” is closer to \$10^3 / example than \$10^12 / example.

Another approach that somewhat questions the reward learning framework is goal specification via natural language. David McAllester writes about this here, and at least one other NLP person who I respect seems optimistic about this idea. My impression is that not much has been done to articulate the advantages/disadvantages of this approach relative to others (or even what the approach itself would look like beyond a very high-level description). I feel excited about someone creating such an articulation.

Finally, while not an alternative approach, an important idea is corrigibility, which is the idea that an agent that is "sufficiently" aligned will tend to become more aligned over time, meaning that it suffices to create agents that have goals which are in the basin of attraction of being aligned with humans, rather than being perfectly aligned from the beginning. If this intuition is correct, we could try to figure out what is required for an agent to fall into this basin, and then seek to use reward learning to hit a target in this basin. This intuition is in the background of both my #1-#6 above (especially #4) as well as amplification, the dialog approach, and probably other approaches as well.

My Personal Take (as of May 2019)

We are currently held back by the lack of a concrete way of testing different approaches to reward learning--existing benchmarks are either gridworld or simple MuJoCo environments, and/or based on qualitative evaluation. The opportunities for reward hacking in most of these cases seem limited--hand-coded policies or hand-coded reward functions would plausibly suffice to solve most of the existing tasks, and I don't feel these tasks can distinguish between promising and unpromising approaches to reward learning. Consequently, I am interested in designing better tasks, which ideally have the following properties:

- Success and failure are clearly separated and quantitatively measurable.
- Simple baselines (such as hand-coded policies or hand-coded reward functions) do poorly.
- Many approaches succumb to clear instances of reward hacking.

Separately, I feel there are connections between reward learning and robustness to distributional shift. This is most clear for #2 (reasoning about errors)---often one can construct a reward-generating process that comes close to giving the true reward for a simple enough fixed distribution of examples, and the problem is that it diverges sharply once we go out-of-distribution (as will likely happen once the system is deployed, or even earlier if we are trying to assign rewards to complex enough actions[10]). Similarly, for inverse reinforcement learning the learned reward function is trained to reproduce human actions in-distribution, with the main problem being that it might not generalize well out-of-distribution (and is moreover forced to act out-of-distribution since it has to take actions on *behalf of* a human, rather than *as* a human).

[10] For instance, asking people to reason about systemic filter bubble effects caused by a given newsfeed recommendation might be infeasible, and hence "do people approve of this newsfeed recommendation" would in that case be a misspecified proxy of "is this recommendation good".

Reward learning is also related to the problem of robustly imputing latent variable models---we can think of the true reward as a latent variable that we are trying to impute given observations of the proxy reward. My preferred way of thinking about this is that we want to impute latent variables (such as a value function) that are accurate not just on the distribution they were estimated from, but also on many other counterfactual distributions (see the section below on counterfactual reasoning for how we might do this).

Finally, I have been thinking about goal specification via natural language and am pretty excited about trying to flesh it out more concretely, but have not come up with approaches that seem plausibly scalable yet.

Scalable Reward Generation

The Problem

Powerful AI systems will potentially have to make decisions in situations that are difficult for humans to evaluate---for instance, on scales far outside human experience, or involving technology that is foreign to present-day humans. Modern ML systems are primarily trained through human-labeled training data (or more generally, human-generated reward functions), which presents an obstacle to specifying which decisions are good in such situations.

Why We Want to Solve It

Below are some concrete settings where relying directly on humans for oversight seems unlikely to work. I sloppily anthropomorphize AIs in the below examples by thinking of them as a single system that "reasons", "explains", etc.; however, I think the problems I describe are harder rather than easier for less anthropomorphic AIs.

1. A candidate reward function for an AI system is "take actions that lead to outcomes that humans would approve of". To instantiate this, we might ask the agent to sample possible outcomes from its predictive model[11] and show them to us (and perhaps even create an interface for us to look around and examine the world under this possible outcome). However, it is unclear how to then assign reward---the world is big and it might take an infeasibly long time to find everything in it. Maybe everyone on Earth is happy but there is a bunch of bad stuff happening on Jupiter. We could ask the AI to direct us to the "relevant" parts but that just kicks the can down the road. Finally, beyond being big, the world likely also looks weird as we go farther into the future, in a way that may also make it hard to evaluate.
2. Alternatively, maybe we don't want to rely on evaluating outcomes, and instead want AIs to "reason in a way that humans would endorse / approve of". But we would need to understand the decision-making process of the AI to provide supervision along these

[11] There is a separate issue of incentivizing the predictive model to be accurate and honest, which I'm ignoring here.

lines. These decisions are not by default human-interpretable, and even if they were human-interpretable the basic decisions are probably low-level enough that human-relevant concepts like “happiness” can only be seen emergently through the combination of many low-level decisions (in the same way that biological functions like “eat” or “sleep” cannot be easily understood in terms of the behavior of individual cells).

3. We could imagine a middle-ground where the AI “explains its reasoning to us” and we give feedback on whether the explanation makes sense and is oriented towards human goals. But if it is explaining its reasoning to a human, there is a fundamental bottleneck on how complex the explanation can become, while the decisions the AI is making become more and more complex. We might expect the explanations to then become less faithful as time goes on.[12]
4. Returning to the original idea of “create outcomes that humans approve of”, maybe we only have the AI ask us to evaluate things that are somewhat more familiar to us (medium-term consequences, rather than long-term consequences). This has the problem that a situation that looks good in the medium-term might be bad in the long-term, although we could perhaps avoid this with a medium-term goal of “create a really well-aligned AGI”. Another problem is that even evaluating the medium-term might require understanding subtle consequences. We can already see this today with e.g. the Facebook news feed (the most obvious metrics for tracking user engagement neglect systemic effects, leading to filter bubbles) or just the fact that governance is pretty hard (cosmopolitan technocratic liberalism seems pretty good but maybe has harmed a bunch of working-class people?). Or maybe humans appear to be happy and in control of the world, but an AI system is secretly pulling the strings behind all major decisions.

A solution to scalable reward generation should give humans a credible way to provide feedback on decisions whose consequences are by default fairly foreign to humans (as is the case in #1-#3 above) and that could involve subtle consequences that are hard to unearth (as is the case in #4 and #1). Keep in mind that the AI system is likely optimizing against this reward, so the reward generation process has to be robust enough to not be exploitable (rather than just giving reasonable answers in some nominal world).

The examples above are a bit more disjunctive than in the previous sections---examples #1-#3 all give disjoint paths to success (evaluate outcomes, evaluate process, interact with explanations) so we have some freedom in how to generate reward.

While reward generation would most directly be used to train the system, I would personally feel more comfortable if we also had mechanisms to oversee the system during its actual operation (and potentially issue course corrections). This raises some additional challenges---for instance, providing feedback to the system might be time-constrained (perhaps even severely time-constrained if the system is operating on a much faster time-scale than humans). In this

[12] I’m again ignoring the issue that the AI might be incentivized to give explanations that sound good but are not accurate reflections of their reasoning.

case we either need a way to provide good feedback quickly, or for the system to avoid requiring time-constrained feedback in the first place.

What A Solution Looks Like
Scalable reward generation is less fleshed out as a problem than Robustness or Reward Learning, partially because it doesn't fit neatly into a well-established machine learning paradigm; perhaps the closest is indirect supervision (example reference), but this is itself less well-established than the preceding areas, and does not grapple with the extreme level of scalability needed here or with being robust to another system optimizing against it.

Scalable reward generation can be split into two goals: generating a **single point** of good reward in a feasible length of time, and decreasing the **number of points** of reward required (thus making the system more economically scalable and making it more okay to spend lots of time generating each data point).

For **generating a single point** of reward, the most concrete research program is capability amplification (discussed above as well; see also debate which is closely related). The basic idea is to train a system to do the supervision that the human would do (this is useful because you can run many copies of that system and also likely run it at a faster speed than humans). You can also iterate this process (train a second system to do the supervision that the "human + first trained system" would do, etc.). Getting the system to exactly imitate a human is probably too hard, but the hope is that using the system in conjunction with the original human overseer can create a system that is noticeably better than the human while also being faithful to the human's intentions; this could perhaps be done by decomposing tasks into smaller tasks that the human could provide feedback on. This approach has the virtue of being concrete, though there are many open questions (Can we actually usefully leverage the trained systems? Can we get a good decomposition? How good of a supervisor can this asymptote to?) that we should try to resolve.

There are some other less fleshed out ideas, which I would like to flesh out further and pursue, or abandon if they turn out to be unpromising. One idea is interactive learning---the system and human interact in a back-and-forth that could potentially be much more informative than placing the primary burden on the human to provide all the input. How to have the AI choose good questions for the interaction is not currently clear (and might itself involve solving some form of alignment problem). In addition, it is similarly unclear how well this scales---does it just make the human's job noticeably easier (if that), or could it get all the way to letting us oversee very powerful AI? If the former, how useful is this (could we stack this together with other improvements that add up to sufficient scalability, or do we need a fundamentally different approach)?

A related idea is to generate reward via some form of dialog system---maybe we can specify some high-level mission using natural language concepts (which may not be fully precise), and thus obviate the need to examine particular situations in detail. I could imagine the dialog approach will end up anywhere between “extremely promising” and “fundamentally unsound” once examined in more detail. It’s also possible that natural language dialog would arise as a specific consequence of a different, more abstract approach, which would make it less valuable to work on directly; however, it’s likely enough this won’t happen that I still think it’s worth pursuing.

Some other proposals include interpretability (if we can understand a system better, maybe we can oversee it better) or better human-computer interfaces (the idea being similar to capability amplification---give humans really good tools to make it easier to provide the reward). My personal but unconfident intuition is that these are less likely to scale to powerful AI systems, but it’s at least plausible. Interpretability may be useful for other purposes such as stress-testing models (discussed in more detail in its dedicated section below).

We next consider **decreasing the number of points** needed by the system (or in an online setting, the frequency of points needed). Here there are more fleshed out research directions, as using less training data is a consistent desideratum in ML. These include active learning (having the system decide what oversight to ask for) and semi-supervised learning (trying to use unlabeled data to figure out as much as possible about the world, which hopefully decreases the amount of labeled data that is required). For a deployed system, we could also imagine only asking for oversight when the system is not confident in the right action, which could allow for cheaper / more targeted ongoing oversight (but doesn’t clearly avoid issues with time-constrained feedback).

Research on generating a single point feels more urgent than research on decreasing the number of points---if we can’t do the former, then the latter won’t help us. Also, the latter has had more progress so far (although I think still not that much), and there are many incentives for people to solve it beyond AI alignment.

Aside. The difficulty of scalable reward generation depends on how well we solve reward learning and robustness. The more the system is trying to work with us to get information (rather than optimizing against us), the easier it is to create a “good enough” reward generation process. The more we can expect the system to faithfully extrapolate knowledge from human-familiar situations to human-unfamiliar situations, the less we need humans to provide rewards for unfamiliar situations. An interesting strategic question (that I haven’t thought much about) is to what extent solutions to these different problems smoothly trade off against each other, versus being all-or-nothing. E.g. is it the case that we basically need to either solve the hardest version of distributional robustness (with a little bit of scalable reward generation thrown on top) or the hardest version of scalable reward generation (with a little bit of robustness + reward learning thrown on top), or could we hope to solve medium-difficulty versions of each of the problems?

Possible Directions

I don’t expect it to be directly useful, but it’s worth becoming familiar with some of the existing work on indirect and distant supervision. A non-comprehensive list of work includes DeepDive / Snorkel, expectation regularization, and distant/indirect supervision for relation extraction and structured prediction. Rohin Shah et al.’s idea of Preferences Implicit in the State of the World can be thought of as a form of supervision that gives large amounts of indirect information about what humans value based on the fact that our existing environment is well-optimized for humans (though it seems unlikely that this information is robust enough to be optimized against directly).

Paul Christiano posted a request for potential problems with capability amplification (which he calls “iterated distillation and amplification” or IDA in the linked post), as well as some potential issues he himself sees. Many of these could be flipped around to yield potential research problems (e.g. “here is a potential obstacle, what is the proof of concept that we can overcome it?”). I am personally most interested in grounding out amplification in some concrete existing system that allows us to oversee something which we wouldn’t otherwise be able to. My biggest concerns (with low confidence) are (1) that we won’t be able to usefully leverage the trained overseer without making it unaligned; and (2) that there are some things that are difficult for a human to supervise even given a lot of sped-up copies (e.g. analyzing low-level motor commands and then having to decide whether the higher-level policy that generated them corresponds to an aligned system). A concrete system (or proposal of a system) would make it easier to instantiate these and other objections and determine whether they hold for that particular system. Ought is currently working on this.

For interactive learning and dialog systems, I don’t think there’s currently any real sketch of what we would even do, or an analysis of how scalable we should expect either to be. So I would say that is the first step. Ideally the sketch of what to do would be thorough enough that any potential limitations are real limitations (as opposed to just not trying hard enough to flesh out the idea), and have enough detail to point to a concrete-ish program of research.

I could also imagine being convinced that interpretability or HCI are more credible paths to scalable reward generation than I currently believe. (I’m actually pretty excited about both routes as potential avenues, just currently would guess they won’t work.) So seeing good positive arguments for either of these would be exciting.

For semi-supervised learning and active learning, there is a large literature on both topics that I won’t recapitulate here.

Acting Conservatively

The Problem

Safety will likely be easier to achieve if systems can notice situations where it is unclear how to act, and either avoid encountering them, take actions that reduce the uncertainty, or take actions that are robustly good. This would, for instance, allow us to specify an ambiguous reward function that the system could clarify as needed, rather than having to think about every possible case up-front.

While out-of-distribution robustness focuses on *noticing* the uncertainty, acting conservatively focuses on what to do *given* the uncertainty. Solutions to one thus interface with solutions to the other. Unfortunately, current methods for constructing uncertainty estimates seem inadequate for acting conservatively, and even given a good uncertainty estimate little work has been done on how the system should use such estimates to shape its actions.

Why We Want to Solve It

Here are some examples of places where we would want a system to act conservatively:

1. We specified a reward function for the system, but some pretty similar reward function would lead to very different actions/outcomes. We would like the system to realize this, realize that it's not that unlikely that we screwed up the reward function, and act appropriately (get further clarification, or avoid ending up in worlds where the reward functions diverge).
2. Conversely, maybe the system is seeking feedback from humans but is not confident that the humans understood the situation it is asking for feedback on. If the feedback plays a crucial role in its decisions, it could continue to clarify until it is confident the feedback it is getting is informative.
3. We want the system to avoid irreversible decisions unless it has high confidence they are good (for instance, genetically engineering all humans to be really happy all the time might be good if happiness is the thing we want to optimize for, but this change could cause substantial value drift in humans, and so is irreversible in the sense that you can no longer ask the original humans what they want/endorse).
4. Maybe the AI discovers some weird fact about sentience that might affect how humans think about moral patienthood, such that the initial values given to the AI seem likely to be flawed. It is probably better for the AI to collaborate with humans on how to fix this rather than making something up itself.
5. We also don't want the AI to experiment with weird physics that might accidentally blow up the earth or cause some other major disaster (unless it is correctly really confident that nothing will go wrong).

All of these involve, on some level, having a representation of some form of uncertainty and altering actions according to that uncertainty. The paradigmatic way of representing uncertainties is via probabilities (typically predictive probabilities under some statistical model). An important point (elaborated below) is that the existing ways of obtaining these probabilities are insufficient for addressing most of the examples above. In fact, it is possible that probabilities themselves are inadequate for this (but let's not get too metaphysical too quickly).

What A Solution Looks Like

Here's a default (but in my opinion incorrect) solution that one might express to the Acting Conservatively problem: "Most ML models give outputs that can be interpreted as probabilities (e.g. the final softmax layer in neural nets). Those probabilities give us a measure of uncertainty, so why not just maximize expected value with respect to that?” There are a number of issues with this approach. The most direct is that the probabilities output by most ML models are not well-calibrated, so we probably don’t want to rely on them. This itself could plausibly be fixed, but there are also some deeper issues. Take scenario #1 above---an event like “the reward function was written down wrong” isn’t a reified event in most ML frameworks, and it’s also not an event that you can directly gather data about, so it’s not clear where you would get the probabilities from (and once you have them, should you actually maximize with respect to the expected value of the reward function? That seems like it would lead to weird behavior, and is not an approach that most people I know would endorse for handling the related issue of moral uncertainty in humans). Additionally, part of the uncertainty we want to represent is uncertainty in the presence of out-of-distribution events (such as the nanobots in example #5), which won’t necessarily manifest in the probabilities learned by ML models if the model family is mis-specified.

One way to make this issue more crisp (at the expense of oversimplifying a bit) is to divide uncertainty into a few different categories (borrowed from this paper): *aleatoric uncertainty* (uncertainty in the predictive distribution of a fixed model, e.g. from process randomness such as a coin flip or label noise, or inability of the model to distinguish between inputs with different outcomes), *model uncertainty* (uncertainty about which model is correct, usually due to statistical estimation noise), and *distributional uncertainty* (uncertainty due to mismatch between the train and test distributions). I would also add in *specification uncertainty* (uncertainty due to errors in how the problem itself was specified, e.g. maybe the likelihood function or reward function is wrong in some way). When people say that neural nets and other ML models tend to be mis-calibrated, they mean that just using aleatoric uncertainty gives overly-confident probability estimates. Typically the assumption is that this is due to not incorporating model uncertainty, and if we did so we would get better-calibrated estimates. But this ignores the additional sources of distributional and specification uncertainty that remain even in the limit of infinite data (when model uncertainty should in theory disappear).[13]

In summary, I think a solution to Acting Conservatively likely needs to incorporate richer sources of uncertainty than just aleatoric uncertainty + estimation error in the model. It should have a way of reasoning about out-of-distribution inputs and fundamental mis-specifications in the problem. We also probably need a decision rule other than “maximize expected value” for some forms of uncertainty (such as specification uncertainty), which is part of why it is unclear to me if probabilities are even the correct framework to use.

Another default solution to Acting Conservatively is to say: “RL gives a framework for decision-making under uncertainty. The system might not explicitly represent probabilities, but the policy is trained to maximize expected reward, which is an average over possible rollouts.

[13] Additionally, the ways in which people measure mis-calibration don't tend to pick up on distributional or specification uncertainty. So solving the problem of mis-calibration as it is currently understood would not mean we had good uncertainty estimates for the purpose of Acting Conservatively.

So we should expect to get the behavior you want out of Acting Conservatively as a by-product of a reasonable training process." It is possible that something like this could work, but I'm currently skeptical. At the very least, we probably need to modify the training process---current RL algorithms only optimize for aleatoric uncertainty, so the above issues regarding other forms of uncertainty would likely apply here (I think we would minimally need to incorporate model uncertainty).

Possible Directions

To some extent, solutions to Acting Conservatively depend on solutions to other parts of the technical alignment problem (e.g. distributional shift, reward learning). If you buy into the "aleatoric/model/distributional/specification" (A/M/D/S) framework, then distributional uncertainty would come from the solution to distributional shift, while "specification uncertainty" would interface with reward learning and perhaps reward generation. However, even given these, I don't think it's clear how to actually choose actions. We could build RL environments that necessitate handling each of these types of uncertainty, or think abstractly about what desiderata a given rule for choosing actions should have.

One idea for Acting Conservatively is to have conservative fallback policies that can be employed when necessary to minimize the chance of a bad outcome, or to find constraints (pre-conditions) on a policy that ensure it will be safe. This idea appears in many guises--relevant terms include safe exploration and reachability analysis. Here are several example references: Safe Exploration in Markov Decision Processes, Safe Exploration in Finite Markov Decision Processes with Gaussian Processes, A General Safety Framework for Learning-BasedControl in Uncertain Robotic Systems, A Reachability Method for Verifying Dynamical Systems with Deep Neural Network Controllers, and Reachability-Based Safe Learning with Gaussian Processes. Here are two surveys on safe exploration, although they are somewhat outdated and a lot of it is what I would call risk-sensitive reinforcement learning (trying to avoid rare high-cost events, rather than trying to avoid unrecoverable errors during training), which seems less relevant.

In the context of classification, see these references that consider classification where the output classes form a hierarchy, and act conservatively by trading off specificity and accuracy (by outputting higher-level classes) in cases of uncertainty.

If you don't want to buy into the A/M/D/S framework and want to go the "let's just do RL route", it seems to me that you'd want to modify the RL objective to implicitly capture the M/D/S aspects in the learned policy. A basic question might be how to modify the RL objective to account for something like model uncertainty, and how to make sure that it was actually accounting for it in a good way. Then one could move on to distributional and specification uncertainty. See these slides by Tom Dietterich for some examples of modifying the objective, as well as a number of other interesting ideas.

Risk-sensitive reinforcement learning aims to avoid rare high-cost events, although it still generally experiences these events at least during training, so this would not suffice for events that are too catastrophic to permit at all. It could, however, perhaps be combined with exploration in simulation (where the bad events would be permissible).

There is also a recent line of work on calibrating neural networks and representing model uncertainty, see e.g. these references.

In general I expect distributional and specification uncertainty to be much harder to handle than model uncertainty, and to likely require different ideas.

Counterfactual Reasoning

The Problem

A system possesses *counterfactual reasoning* abilities if it can reason not just about the actual world / distribution of inputs it is faced with, but also other (potentially physically or logically impossible) worlds / distributions. There are many theories for how a system ought to / might go about this, although no clear approaches for how to apply them scalably to existing systems. At the same time, counterfactual reasoning touches upon many other aspects of AI alignment, and so a satisfying solution could broadly make alignment easier.

Why We Want to Solve It

In the most optimistic case, counterfactual reasoning would resolve each of the following problems:

1. Robustness to distributional shift asks for a system that does well not just on its training distribution, but on other possible test distributions as well. If those test distributions are within the set of counterfactuals that the system can reason about, then we expect it to do well on the test distribution as a consequence.
2. Asking a system to simulate the consequences of various plans it might execute (and then showing them to humans) involves a limited form of counterfactual reasoning. This would allow us to evaluate plans without actually executing them, which seems safer.
3. We can ask a system to take actions whose outcomes a present-day human would counterfactually rate highly if you showed the outcome to them. This is related to Paul Christiano's human in the counterfactual loop proposal, but has the advantage that the counterfactual human cannot be physically manipulated by the AI system (because the human doesn't exist). (Note however that the AI could try to fool the counterfactual human, e.g. by presenting outcomes in a misleading way.)
4. Relatedly, you could ask a system to do "what a human would tell it to do" in a given situation; this avoids the problem of the system optimizing against the human ratings (since it just follows directions instead of optimizing), but has its own host of issues such as it being unclear how a human would scalably tell an AI system what to do in complex situations.

5. We could imagine an interactive process for reward learning, where the system probes the human's preferences regarding a large number of counterfactual situations. This might provide a more robust way of specifying a reward function (because it is pinned down in a wider set of worlds, including "impossible" worlds).

In addition, there are some more abstract reasons to think counterfactuals might be important:

6. Much of human dialog relates to counterfactuals, so counterfactuals might be an important part (or perhaps consequence) of dialog-based approaches to alignment.
7. Logical counterfactuals (e.g., "What would happen if 619023 was prime?") seem related to logical uncertainty, which may be helpful if we wish to make predictions that are robust to computational constraints.[14]

Different desiderata require stronger or weaker forms of counterfactual reasoning. Intuitively, #2 and #5 require weaker forms of counterfactuals, while #7 requires a strong (albeit narrow) form of counterfactuals. From the examples above, we can see that counterfactual reasoning captures many underlying themes in AI alignment, although it is possible that the concept is too broad or fuzzy to make useful progress on--perhaps it captures all of these themes because we don't really understand counterfactuals right now and it is tempting to hope they will solve a lot of problems for us. That caveat aside, I am pretty excited about work on counterfactual reasoning--I think it's possible that there are unifying conceptual ideas that would lead to progress on all or most of the problems above.

What A Solution Looks Like

The most common way of formalizing counterfactuals is through **causality**. I know of two overarching approaches to causality, most famously advocated by Judea Pearl and Donald Rubin, respectively (I will refer to them as Pearl-causality and Rubin-causality).

In Pearl-causality, we require models to specify a complete process that generates the data (as in a Bayes net or probabilistic program), and define the set of allowed counterfactuals to be questions of the form "What would happen if we ran the process forward but intervened to set variable X to value x0 (as opposed to whatever it would have been)?" For instance, you could ask "If there was bad weather on the roads but hypothetically it did not cause any accidents, would we expect to hear sirens?" This corresponds to setting "weather" to "bad" and "accidents" to "false" in this Bayes net.

The issue: By requiring a complete generative process for the data, Pearl-causality seems to assume that you have (or can learn) a complete model of everything that is causally interesting in the world. In practice this approach leans on using domain knowledge to write down the necessary causal structure, which doesn't seem scalable.

[14] MIRI, the organization that wrote the linked document, make a different case for logical uncertainty than the one I give, and also care about counterfactuals for decision theory; I don't fully understand their arguments and suspect I would disagree with them, so have not tried to reproduce them here.

In contrast to Pearl-causality, Rubin-causality separates counterfactual questions from the specific model. It defines an *intervention* as any experiment that you could in principle perform in the world (it's okay if it's expensive or unethical, but if there's no in-principle experiment then your question is "FUQ'd"). The set of counterfactual questions corresponds to the set of interventions ("What would happen if we applied this intervention?"). We then typically try to leverage some minimal set of assumptions to identify the effect of a specific intervention. A famous example concerns the effect of class size on student achievement. There was a school system that mandated a maximum class size of 30, so if there were 31 students they would be split into groups of 15 and 16. If we make the believable assumption that having 30 vs. 31 students is basically random and independent of any other decisions, then we can use this to estimate the effect of class size.

Rubin-causality doesn't seem scalable, but for a different reason than Pearl-causality. Pearl-causality needed a model of all the causal effects, whereas Rubin-causality can get by with just identifying a carefully-selected "essentially random" part of the data (the 30 vs. 31 assumption in the above example). But identifying this seems to itself require domain expertise. Moreover, Pearl-causality gives us answers to an entire combinatorial family of counterfactual questions (everything defined in the Bayes net), while Rubin-causality requires us to come up with a different source of randomness for every intervention we care about.

I haven't done justice to either approach above. For instance, Rubin-causality has other cool ideas like propensity matching (re-weight your data to simulate the effect of whatever intervention you wanted to apply) and doubly-robust estimators (a way of getting an estimate that is correct as long as either one of two assumptions holds). Pearl-causality probably contains lots of additional ideas as well, but I'm less familiar with it. Nevertheless, the scalability issue does seem like a real problem for both approaches.

Can we salvage these frameworks to make them more scalable? There's something intuitively-appealing about Pearl's version of counterfactuals as "alternative settings of variables in a generative program", but we need an account of how we would learn such a generative program from data. It seems unlikely to me that maximum-likelihood estimation is the right approach---Newtonian mechanics is a pretty good causal theory of the world, even though quantum mechanics is more accurate. If we trained a model to predict physics well, and weren't lucky enough to stumble on quantum mechanics, I suspect maximum likelihood would not give us Newtonian mechanics but rather some jumbled mess (even if it gave Newtonian mechanics as the first-order term, we should expect the learned model to contain less meaningful components that happen to fit the error terms well on the training distribution).

Here is a more abstract restating of the above: Maximum likelihood is not very good in the presence of model mis-specification---it tends to aggressively optimize against the specifics of whatever dataset it is presented with, and will therefore tend to take advantage of spurious correlations that don't hold up in even nearby possible worlds. For many reasons (such as simplicity) we often want to work with incorrect models (such as Newtonian mechanics), but we don't have a good learning paradigm for pointing to "good" incorrect models relative to bad ones.

My preferred framing of counterfactual reasoning is therefore---*What alternatives to maximum likelihood produce models that are less overfit to spurious correlations in the data, and which are therefore more likely to still produce good predictions in counterfactual worlds? Can we characterize the family of counterfactuals that such models produce good predictions for?* This framing separates counterfactual worlds from causal interventions (causal interventions yield one type of counterfactual world, but any world where our model makes "good" predictions comprises a valid counterfactual). This framing is admittedly far less fleshed out than Pearl- or Rubin-causality. I feel particularly excited about it because, among other things, alternative learning procedures to maximum likelihood might also shed light on alternatives to reward maximization (a large part of the alignment problem is that maximization is hard to employ safely).

There are other definitions of counterfactuals beyond the Pearl and Rubin definitions given above. They are less instantiatable than Pearl and Rubin, but have the advantage of being less tied to interventions, which I feel produce an overly narrow conception of counterfactuals. One example is Lewis's possible worlds interpretation of causality. I think some people have also defined counterfactuals in terms of robust predictive rules (kind of similar to my really fuzzy attempt above) but I don't have a reference off-hand.

A final issue is that many (but not all) of the counterfactuals in our examples #1-#7 above are about fuzzy high-level concepts, and so (a) might not be valid counterfactual events in e.g. a Pearl-style model, and (b) might be hard to specify formally in the language of the model, even if they do correspond to a valid counterfactual event.

Possible Directions

On alternatives to maximum likelihood: the "not really an alternative" alternative is to incorporate lots of counterfactual worlds into your training data. This is the idea behind domain randomization and sim-to-real: if we modify the simulation in a lot of different ways, hopefully the family of simulated worlds is rich enough to contain the real world or something like it. We can re-state this as saying that hopefully the real world is within the family of counterfactuals defined by the different versions of the simulation. Some challenges are (1) it's less clear how to do this if we aren't training in simulation, and (2) we need to define the task objective in the counterfactual worlds (in domain randomization, the objective is to reach some goal state that is the same in all the counterfactuals, and what changes are other aspects of the environment like the dynamics, lighting, etc.; but it's not clear we can always expect defining the objective to be so easy).

Some larger departures from maximum likelihood are based on **agnostic learning** and **partial specification**. Agnostic learning seeks to design procedures that work even if the model is mis-specified. For instance, a classical approach to clustering is via fitting a mixture model using maximum likelihood, and then defining cluster membership via each component of the mixture; this works well if the mixture model is well-specified and identifiable, but can fail otherwise. *Agnostic clustering* asks whether there are clustering algorithms that work without having to assume a specific model family; for instance, maybe we can just assume that clusters are "well-separated" in some appropriate sense---see this paper by Kumar and Kannan as well as my more recent paper (among many others) for more info.

A closely related concept is *partial specification*, where rather than modeling a complete generative process of the data we only specify certain aspects of the generative process (this in particular precludes maximum likelihood because there is no likelihood to maximize). The simplest example is ordinary least squares, which works whenever the errors are uncorrelated from the signal (even if they are non-Gaussian, which is what we would need to model least squares as maximum likelihood). See my paper on unsupervised risk estimation for a more complex example as well as a brief literature review on partial specification, and my paper (with Banghua Zhu and Jiantao Jiao) on generalized resilience, which defines non-parametric assumptions under which robust estimation is possible. I also recommend Section III of this excellent lecture by Lars Peter Hansen.

I also recommend Leon Bottou's magnum opus on counterfactual reasoning (beyond being a great overview, it gives additional examples of how counterfactuals might help with mis-specified rewards; for instance, "maximize average user happiness" might incentivize getting all unhappy users to quit your website as quickly as possible, while "maximize average happiness that the current pool of users would have" removes this effect). Another relevant area is contextual bandits.

Detecting failures in advance

The previous section lays out a list of obstacles to AI alignment and technical directions for working on them. This list may not be exhaustive, so we should also develop tools for discovering new potential alignment issues. Even for the existing issues, we would like ways of being more confident that we have solved them and what sub-problems remain.

While machine learning often prefers to hew close to empirical data, much of the roadmap for AI alignment has instead followed from more abstract considerations and thought experiments, such as asking "What would happen if this reward function were optimized as far as possible? Would the outcome be good?" I actually think that ML undervalues this abstract approach and expect it to continue to be fruitful, both for pointing to useful high-level research questions and for analyzing concrete systems and approaches.

At the same time, I am uncomfortable relying solely on abstract arguments for detecting potential failures. Rigorous empirical testing can make us more confident that a problem is actually solved and expose issues we might have missed. Finding concrete instantiations of a problem can both more fruitfully direct work and convince a larger set of people to care about it (as in the case of adversarial examples for images). More broadly, empirical investigations have the potential to reveal new issues that were missed under purely abstract considerations.

Two more empirically-focused ways of detecting failures are **model probing/visualization** and **red-teaming,** discussed below. Also valuable is **examining trends** in ML. For instance, it looks to me like reward hacking in real deployed systems is becoming a bigger issue over time; this provides concrete instances of the problem to examine for insight, gives us a way to measure how well we're doing at the problem, and helps rally a community around the problem. Examining trends is also a good way to take an abstract consideration and make it more concrete.

Model Probing and Visualization

The Problem
Most ML models are "black boxes" in the sense that they are complex systems built up from low-level primitives (such as linear transformations and non-linearities). Even if they make good predictions, they do not necessarily do so by means of human-meaningful concepts. Our intuitions about how ML models work are likely misguided, and better tools for probing ML models could both help us build better intuition and help predict potential failures. This could include error analysis, evaluation on distributions of atypical inputs, or visualization tools.

Why We Want to Solve It
Currently, the most common way of evaluating a model is its test accuracy on a task or set of tasks. Richer ways of probing a model could help us predict failures in advance, because potential failure modes may manifest as e.g. unexpected internal states of a neural network, or unexpected extrapolation to atypical examples, before they lead to actual perverse outputs.

A good example here is adversarial examples, which seem to have been discovered as a result of an investigation that (a) probed the robustness properties of neural networks and (b) questioned the semantic meaningfulness of hidden node activations in neural networks.

Some additional examples and arguments:

- It is tempting to conjecture that ML models achieving human-level performance on e.g. object recognition do so in the same way that humans do, but this need not be the case, and the presence of adversarial images provides evidence against this. Good model probing tools could help us determine to what extent neural nets or other ML models share concepts with humans. This could be particularly important for concepts that are used as part of a learned reward function.
    - I've seen opinions ranging from "what neural nets do is similar to what humans do" to "what neural nets do is nothing like what humans do". From my limited experience visualizing object recognition models, the answer seems somewhere

in between, where in some aspects model behavior is quite intuitive and in others it seems bizarre. Understanding this better seems important to predicting how models will behave in new situations.

- We would also like to make sure that the model is “doing the right things for the right reasons”, which is a much higher bar than just getting high test accuracy (which is what ML as a field traditionally emphasizes).
- The impact of training data on the learned model is not well-understood---there seem to be some potentially weird effects in high-dimensional data where every individual training point has a “large influence” on the learned parameters, and the prediction of many individual test points are sensitive to small changes in the data or training procedure (moreso than the aggregate accuracy). It is not clear if this will lead to safety issues, but it is sufficiently counterintuitive that it may be worth understanding better.

A separate, cross-cutting point is that having a high-bandwidth channel for interrogating a model (e.g. a vivid visualization vs. a table of numbers) can in my experience quickly change my mental map of what neural nets or other models are doing. As noted above, having accurate mental maps of how ML models work seems important for anticipating their behavior in new situations and for anticipating when they might fail.

What A Solution Looks Like / Possible Directions
In contrast to the technical alignment problem, detecting failures is an ongoing process rather than a concrete question that can be “solved”. Below I describe some tools that could help with this process.

One way of probing a model is to simply evaluate its accuracy on additional datasets (or under additional metrics). Evaluating robustness to different types of perturbations (as in ImageNet-C, Stylized-ImageNet, or adversarial examples) or different data distributions (as in ImageNet-A) falls into this category. Such evaluations seem particularly useful because they are about how a model’s predictions extrapolate to a new situation. Neural nets are empirically robust to some types of stochastic perturbations but not others, and I (and probably most people) have limited ability to predict in advance whether a network will be robust to some new type of perturbation. Building better understanding here seems important.

Visualization tools aimed at providing researchers with a higher-bandwidth window into the behavior of ML models also seem broadly useful. Chris Olah and collaborators have written extensively about these sorts of tools and provided several examples through the Lucid library. Note however that visualization can be misleading--it is easy to extract things from models that *seem* human-interpretable but are not tracking what NNs are doing in a meaningful way. For instance, recent work suggests that saliency maps have this problem, and I expect many visualizations to have this issue by default.

There have recently been some interesting case studies using a combination of out-of-distribution evaluation and visualization to reach some preliminary conclusions about how models tend to generalize. For instance, Geirhos et al. argue that CNNs are biased towards texture over shape, while Zhang and Zhu argue that adversarial training removes some of this texture bias.

Other potentially useful tools include better evaluation procedures for model performance (e.g. work on evaluating generative models), influence functions, and debugging tools such as model assertions.

Model visualization falls under the umbrella of interpretability. However, “interpretability” is poorly-defined in the ML literature and people mean a lot of different things by it (Zachary Lipton elaborates on this here). I think most work under this broader umbrella has limited relevance to detecting failures and is instead pursuing some other goal, such as satisfying certain legal or ethical desiderata.

For instance, work focused on making models interpretable to naive (non-ML-savvy) human subjects does not seem that relevant to detecting failures, since the end users of interpretability tools for failure detection would be ML researchers. Relative to naive users, ML researchers will both be more familiar with the tools and place more demands on them. It is plausible that this style of work would still generate insights that are applicable to this use case, but it is driven by a different goal.

Some papers also make vague claims that Bayes nets, or linear models, or some other anointed model family is more interpretable than neural nets. I could imagine some version of this work being useful (e.g. maybe you could design a different neural net architecture for which it was apparent that adversarial examples wouldn’t be an issue). But most of the time these claims are not substantiated, and do not address interpretability in the regime where there are a large number of input features, which is necessary for scalability.

Red-teaming

The Problem

“Red-teaming” is a concept in cybersecurity as well as military and business intelligence. The idea is to create a group that works separately from the rest of the organization, and whose goal is to find flaws in plans or defenses. In AI safety and alignment, red teams could expose flaws in plans or research agendas, or could anticipate or demonstrate failure modes of an individual system. Currently, there is little formal infrastructure (in terms of organizations, tools, and ideas) to support red teams.

Paul Christiano’s blog post on red teams also provides a good overview. Red-teaming need not be done formally as part of an organization; it could e.g. involve an individual researcher trying to find flaws in an existing system or approach.

Why We Want to Solve It
As mentioned above, it is unlikely that we have already articulated all potential failures of AI systems. Red-teaming at the level of plans or agendas is a mechanism for finding additional such failures.

Additionally, red-teaming individual systems can help instantiate failures modes that were previously abstract, as in the case of adversarial examples (or in traditional computer security, where one style of work is to show that a given security vulnerability actually has a corresponding exploit). Such instantiation can both make it more clear how to address the failure, and also create a greater sense of urgency among any skeptical parties. If the system is actually going to be deployed in the world, red-teaming can increase our confidence that the system is safe.

My intuition is that red-teaming often improves the effectiveness of organizations and helps uncover blind spots. If we think of "the field of AI alignment" as a sort of organization, then we would want red teams for this reason as well.

What A Solution Looks Like
I would like to see more object-level work that looks like red-teaming, as well as more institutional infrastructure designed to support such teams (e.g. it would be good if DeepMind had such a team, assuming they don't already; and it would also be good for red-teaming to be seen as a genre of work within academia).

An example of red-team thinking is this post where Paul Christiano shows that naive deep RL systems would do quite poorly on alignment. While not directly about safety, my paper with Zachary Lipton can be thought of as red-teaming empirical standards in ML. Researchers in the FAT (Fairness-Accountability-Transparency) area often engage in red-teaming, as in the Excavating AI project.

One can also exhibit failures in specific systems. The original adversarial examples paper is an instance of this that was very high-impact and exposed a way that ML systems might behave differently than intended. Papers questioning the reliability of various datasets have also sometimes had high impact (here, here, and here).[15] My sense based on the success of the above papers is that there is appetite in the ML community for good empirical red-teaming work, which is a hopeful sign. On the other hand, there is often defensiveness on the part of the people whose work got red-teamed. It would probably be good to have better social conventions around how to constructively red-team, and how one ought to respond to red-teaming efforts.

Possible Directions

[15] Dataset reliability isn't directly relevant to alignment, although better knowledge of how to construct informative datasets may aid in evaluating the safety of ML systems.

Continuing work on adversarial examples seems useful, and I am particularly excited about work that uses more realistic and general threat models, such as the unrestricted adversarial examples competition. I would like to see work in additional domains beyond images (including value learning) and on additional failure modes beyond prediction errors.

On domains beyond images: There has been some work exploring adversarial examples in RL and in malware detection. Regarding value learning in particular, I would guess that the value functions learned by existing frameworks (such as deep RL from human preferences) would perform very poorly out-of-distribution; verifying this could be worthwhile.

For uncovering failure modes beyond prediction errors, here is one possible example. Supposing that learned representations (e.g. image vectors, word vectors) will be used in powerful AI systems, we may be interested in ways in which these representations behave counterintuitively or perversely. Is there some way of probing such representations and understanding in what ways we should/shouldn't expect them to perform well? For instance, some work suggests that word embeddings can reflect gender stereotypes present in the data (though see a recent partial qualification).

Paul Christiano also discusses some potential directions in the previously-mentioned blog post.

Methodological Understanding

The Problem

Methodological understanding is the understanding of best practices regarding good design and development patterns, how to debug a system, common failure modes to check for, and so on. In addition to a good conceptual understanding of AI alignment and a will to put that in place for powerful AI systems, we will likely need a well-developed understanding of best practices around how to build safe and aligned systems.

Why We Want to Solve It

In machine learning, examples of methodological understanding include:

- "Gradient checking": A way of checking that backpropagation was implemented correctly, which guards against errors that would otherwise be difficult to even notice.
- A/A testing: A way of checking that A/B testing software doesn't turn up spurious false positives.
- Train/validation/test (and cross-validation): this underlies essentially all empirical work in ML, but one could imagine a worse version of ML where we didn't use validation sets and just did train and test--or even worse, train and train as is actually the case in RL right now.

I am worried that we are missing methodological ideas as basic as these in the context of AI alignment, and also that the methodological problems we need to solve are harder (because having any catastrophic errors is unacceptable, and because we will be more limited in our ability to safely experiment with full-scale systems before deploying them).

One might be inclined to think of methodological work as "easy" compared to technical work. I think this is wrong. Some illustrative recent anecdotes include the continuing difficulty of establishing best practices for RL in deterministic environments, the difficulty of attributing improvements to architecture choice versus hyperparameter tuning (or even different random seeds), and the ease of constructing datasets that appear to measure an interesting capability (like understanding long-range dependencies) but can actually be solved by simple heuristics.

An example of a recent good methodological paper is "Obfuscated Gradients Give A False Sense of Security"---it highlights the problem it discusses by breaking a number of existing systems, and then offers a checklist of signs that a system suffers from obfuscated gradients. "On Evaluating Adversarial Robustness" is another good methodological paper from the adversarial robustness community.

What A Solution Looks Like
I do not have a complete list of the sorts of methodological understanding we would like to have for AI alignment, and the goal here is fuzzier than for much of the technical work. Nevertheless, here are some example questions that seem unanswered to me:

- Best practices for eliciting human preferences/feedback. Relatedly, understanding the ways in which humans tend to mess up when providing preferences/feedback.
- There is a danger of overfitting to only the failure modes we have observed so far (and then succumbing to some new not-yet-observed failure mode at deployment time). For instance, adversarial training solves known issues with a system by adding examples exhibiting that issue to the training process. How do we avoid overfitting to only the failure modes we've found so far, and when can we be confident that a procedure such as adversarial training has actually conferred sufficient robustness?
    - Our paper on transfer of adversarial robustness across perturbation types attempts to start grappling with this issue, as does the ImageNet-C paper.
- What amount and types of testing tend to be enough to ensure that a system will behave as intended in new situations (or at least not catastrophically fail)?
- What are good ways of testing whether a given method is likely to scale to very powerful systems?
- What are good ways of assessing the degree of progress in each of the technical alignment categories discussed above?

In addition, each of the specific technical problems discussed in the earlier sections would benefit from a corresponding set of best practices around their proposed solutions.

Near-term safety-critical systems could help provide inspiration for best practices. For instance, in robotic surgery, even a single catastrophic error is likely considered unacceptable and I would be interested in understanding the vetting process there. Other areas that might provide inspiration are air collision avoidance systems, autonomous driving, cybersecurity, and certain large-scale systems such as those deployed by Google, Facebook, Microsoft, etc. While the best practices in these areas would not apply directly to very powerful AI systems, there may still be important insights and some borrowable methodologies. See this best practices paper by Google and this MSR paper on avoiding technical debt. My understanding is also that Google has to regularly deal with reward hacking in the form of their systems overfitting to easily-collected metrics at the expense of actual user satisfaction, and also with adversarial attacks on their ML systems such as people getting past Youtube's content filters.

Possible Directions

As mentioned above, understanding when training against some failure modes generalizes to novel failures seems important. One could build a collection of failure modes, and test when training on some subset of the failure modes confers robustness to the remaining subset (and try to understand how to predict when this will happen).

Constructing and critiquing benchmarks for various AI alignment tasks could also be worthwhile. However, constructing benchmarks is a large amount of work, and it is easy for benchmarks to end up not being useful, so I would advise anyone doing this to solicit feedback (especially from people with experience building datasets, and people with experience in the relevant domain) before doing so. An example of constructing a benchmark is the various adversarial example competitions.

I expect some methodological work to be done in conjunction with fleshing out a given technical proposal (e.g. best practices around elicitation might be done in conjunction with work on reward learning or scalable reward generation). Attending to the methodological aspects of the work (i.e. understanding in detail what practices are necessary for things to work well and reproducibly, as opposed to just building a proof-of-concept system) will likely lead to better methodological understanding.

System-Building

The Problem

When building large systems, many challenges arise that would not arise in a proof-of-concept implementation, or even in a small-scale system. For example, we often need to split large systems into smaller self-contained components for the design process to be feasible. We also likely need to have many engineers working together, not all of whom communicate on a daily basis, which requires more effective separation of responsibilities than for a smaller system. Proposals for building powerful AI systems safely should account for the fact that such systems might be the result of such a large-scale project.

Note: This is one of the more speculative sections, as it makes assumptions about how powerful AI systems might be built (i.e. that it will be a large-scale engineering project). There are alternative possibilities, such as “small research team with access to large amounts of compute resources and data”. In this case System-Building might be an unnecessary problem to solve, or we might need to instead solve a different problem related to the alternative design process. On the other hand, system-building issues could still arise in other aspects of AI alignment. For instance, the sub-system for scalable reward generation might involve many humans interacting with the system in a complex way.

Why We Want to Solve It

Here is an example of something that might go wrong in the absence of good system-building. It is based on a talk by Leon Bottou:

- There are two teams. Team A is building a recommender system, and team B builds a system to decide what font size to display the recommendations in.
- Team A decides to perform epsilon-greedy to explore the space of recommendations.
- Team B, if optimizing independently of A, will always display the exploratory recommendations in small font (since it learns that those recommendations are less likely to be clicked on). Therefore, the exploration effectively never occurs since in practice users will not notice the small font.

The issue in the above example is that components that in isolation appear to achieve their goals can fail to do so in the context of the larger system. We might worry about something similar for alignment: perhaps individual components appear reliable and aligned but the system as a whole is not. Or maybe some other type of problem occurs that we haven’t articulated yet. My main worry is that we don’t really know much about what might go wrong here, and our current evidence about large-scale systems is that things generically tend to go wrong without good engineering paradigms.

Thinking about large systems has other benefits: many failure modes of systems get exposed at scale--we can think of this as a brute-force version of red-teaming (“anything that can go wrong will go wrong given a sufficiently large number of users”). Large-scale systems also probably bring a lot more optimization power to bear than smaller systems, making them behave “more like” future powerful AI in some ways. For instance, my sense is that (some) people at Google care more about reward hacking than the average ML researcher because their systems are actually powerful enough to reward-hack their proxy signals. Finally, we want proposed methods for alignment to be robust to scaling issues so that they work for powerful future AI systems, and some of these issues likely already manifest for existing large-scale systems.

What A Solution Looks Like

I don’t have a clean idea of what it would mean to “solve” the System-Building problem, and maybe there is no concrete goal state and it is more about accruing increasingly good intuition and best practices. So figuring out what exactly we want here is part of the question. It might look a lot like getting better methodological understanding, but in the specific context of a large system. Perhaps it would also incorporate ideas from systems engineering in other disciplines (e.g. civil engineering).

Relative to the other areas above, I would potentially expect "regular" system-building work to transfer to "alignment" system-building work more than normal, because large systems already need high reliability, which is a key component of alignment.

Possible Directions
I would start by trying to better understand current best practices and challenges in large systems. There have been some white papers and talks about this.

I expect a lot of work on System-Building to involve accumulating best practices rather than doing traditional technical research, but there are also potential technical challenges. For instance, ML systems break many traditional software abstractions, and we might want to fix this by causing ML systems to have better contracts (see this talk by me; Leon Bottou also talks about ML with contracts in the talk linked in the preceding paragraph). As one example of technical work, this paper is an effort by MSR to deploy contextual bandits algorithms without incurring technical debt.

Another possible direction might be trying to design better tools for debugging stochastic systems. For example, Certigrad formally verifies certain stochastic properties of ML software.