ML Systems Will Have Weird Failure Modes

JAN 25, 2022 • 8 MIN READ

Previously, I've argued that future ML systems might exhibit unfamiliar, emergent capabilities, and that thought experiments provide one approach towards predicting these capabilities and their consequences.

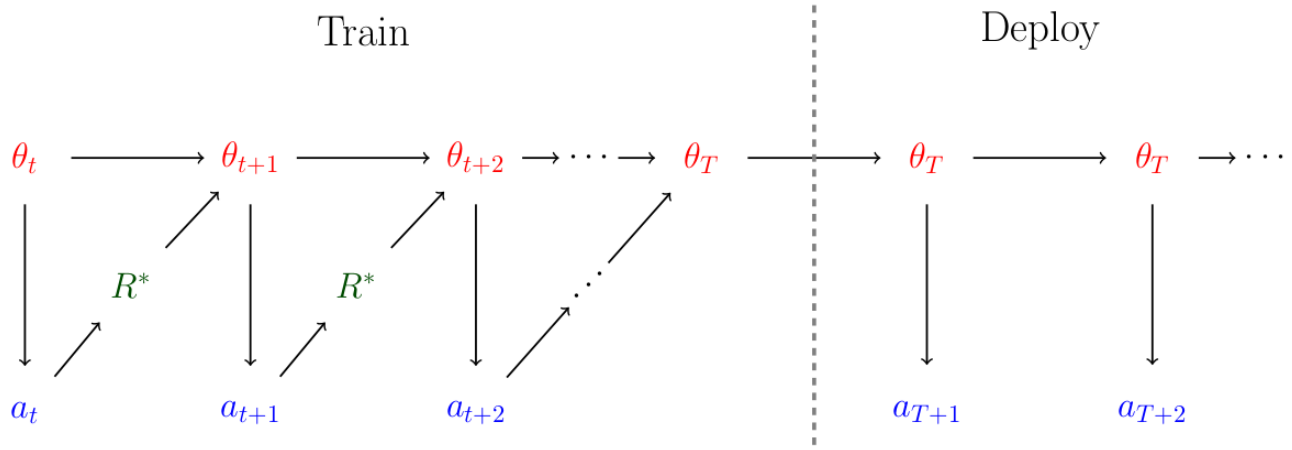
In this post I'll describe a particular thought experiment in detail. We'll see that taking thought experiments seriously often surfaces future risks that seem "weird" and alien from the point of view of current systems. I'll also describe how I tend to engage with these thought experiments: I usually start out intuitively skeptical, but when I reflect on emergent behavior I find that some (but not all) of the skepticism goes away. The remaining skepticism comes from ways that the thought experiment clashes with the ontology of neural networks, and I'll describe the approaches I usually take to address this and generate actionable takeaways.

Thought Experiment: Deceptive Alignment

Recall that the optimization anchor runs the thought experiment of assuming that an ML agent is a perfect optimizer (with respect to some "intrinsic" reward function R). I'm going to examine one implication of this assumption, in the context of an agent being trained based on some "extrinsic" reward function R^* (which is provided by the system designer and not equal to R).

Specifically, consider a training process where in step t, a model has parameters θ_t and generates an action a_t (its output on that training step, e.g. an attempted backflip assuming it is being trained to do backflips). The action a_t is then judged according to the extrinsic reward function R^*, and the parameters are updated to some new value θ_{t+1} that are intended to increase a_{t+1}'s value under R^*. At some point, the model

final parameters θ_T, and continues to take actions. The following diagram illustrates this process:



Now, let's assume that the model θ_t is a perfect optimizer whose objective is to maximize the discounted value of an intrinsic reward $R \neq R^*$. That is, θ_t picks the action a_t satisfying

$$a_t = \text{argmax}_a \, \mathbb{E}[\textstyle\sum_{s=0}^{\infty} \gamma^{-s} R(a_{t+s}) \mid a_t = a].$$

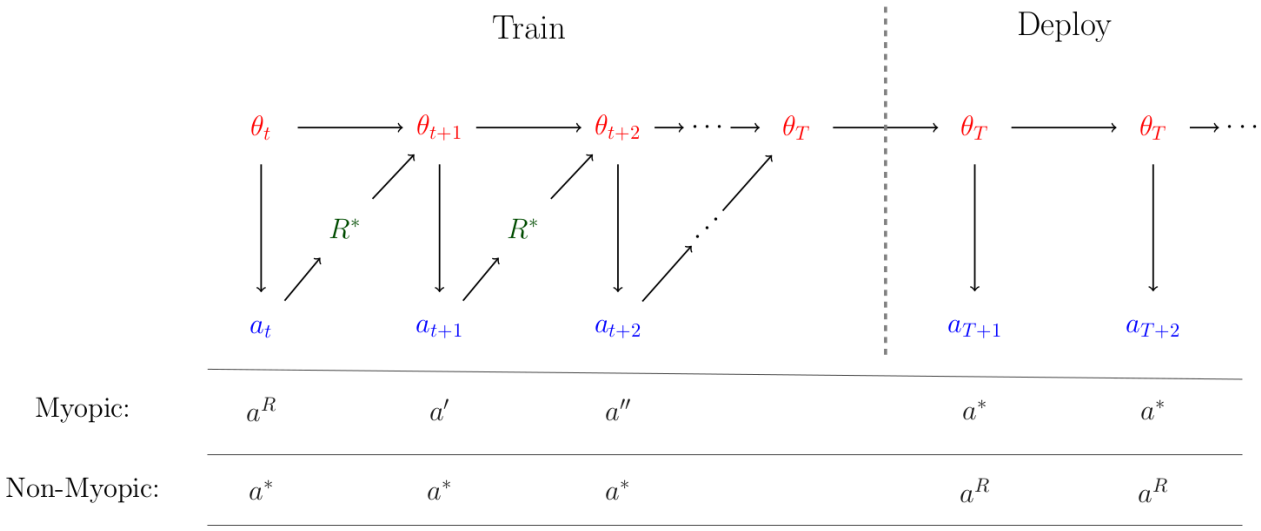
(I know that this is an unrealistic assumption. We'll examine the assumption in detail in the next section, but for now please grant it even if it requires suspending disbelief.)

What action a_t will θ_t pick? Let's define $a^R = \text{argmax}_a \, R(a)$ and $a^* = \text{argmax}_a \, R^*(a)$--that is, a^R maximizes the (instantaneous) intrinsic reward while a^* maximizes the extrinsic reward.

Assuming that γ is close to 1, the model wants a^R to be picked on as many future rounds as possible (including at deployment time). If θ_t plays myopically, this means it would pick $a_t = a^R$. However, this is not the optimal non-myopic action: if $a_t = a^R$, then θ_{t+1} will be updated towards producing outputs less like a^R and more like a^*, so outputting a^R in the current round leads to fewer a^R outputs in future rounds.

Instead, I think the optimal non-myopic action is approximately the following: during training time, output a^* on every time step, so that the training process doesn't update θ (since it's already optimal) and hence $\theta_t = \theta_{t+1} = \cdots = \theta_T$. This guarantees that the deployed agent is θ_t, which is now free to output a^R on every time step without risk of being modified by the training process.[1]

The myopic and non-myopic outcomes are illustrated in the figure below:



Myopic and non-myopic agents with an intrinsic reward, R. The myopic agent plays high intrinsic-reward actions during training, but is eventually modified to output action a^* with high extrinsic reward R^*. The non-myopic agent instead plays a^* during training so that it can pursue its intrinsic reward at deployment, by playing a^R at all post-deployment time steps.

To put it another way–the model's behavior will look aligned with R^* during the training process, but will switch to being non-aligned once deployed: so-called "deceptive alignment". This behavior is unintuitive to me, since it shows that a system's behavior can shift significantly between train and test, even in the absence of an obvious distribution shift between train and test outputs[2].

Engaging with Deceptive Alignment

When I first heard the above argument, I thought it was pretty crazy and implausible, mainly because my intuition said this "just wasn't how ML systems worked". When I think about why I feel that way, I realize it's because the scenario invokes capabilities that ML is currently bad at: long-term planning and understanding complex features of the environment (i.e. the training process and its ramifications). However, emergence implies that these properties could easily appear in the future, even without explicit design[3]. As a result, I've come to discount this particular intuition.

However, I do think there are subtler reasons to think the deceptive alignment story won't play out as written. Here are a few:

1. It's not clear why the model θ would come to be optimizing a reward function R in the first place. Yes, it is the case that deceptively aligned models achieve the global minimum of training loss, so in that sense they are incentivized by the training process. But so is an actually aligned model, so which one you end up with has to depend on the inductive bias of the training process.
2. Reward functions are simpler than policies and typically learned faster. So by the time the system is smart enough to have long-term plans, it will already have a very good representation of its intended reward function. We thus might hope that most of the model's internal representations are devoted to achieving high reward in a straightforward manner rather than through long-term deception.
3. To the extent that a model is not aligned, it probably won't be the case that it's deceptively aligned with an explicit reward function R---that's a very specific type of agent and most agents (including humans) are not maximizing any reward function, except in the trivial sense of "assign reward 1 to whatever it was going to do anyway, and 0 to everything else".
4. Deceptive alignment is a specific complex story about the future, and complex stories are almost always wrong.

I find these points persuasive for showing that deceptive alignment *as explicitly written* is not that likely, but they also don't imply that there's nothing to worry about. Mostly they are an argument that your system might be aligned and might be misaligned, that if it is misaligned it won't be *exactly* in the form of deceptive alignment, but ultimately what you get depends on inductive bias in an unknown way. This isn't particularly reassuring.

What I take away from thought experiments. Per the discussion above, the failure mode in my head is not "deceptive alignment as written above". Instead it's "something kind of like the story above but probably different in lots of details". This makes it harder to reason about, but I think there are still some useful takeaways:

> After thinking about deceptive alignment, I am more interested in supervising a model's process (rather than just its outputs), since there are many models that achieve low training error but generalize catastrophically. One possible approach is to supervise latent representations using e.g. interpretability methods.

> While I don't think neural nets will be literal optimizers, I do think it's likely that they will exhibit "drives", in the same way that humans exhibit drives like hunger, curiosity, desire for social approval, etc. that lead them to engage in long-term coherent plans. This seems like enough to create similar problems to deceptive alignment, so I am now more interested in understanding such drives and how they arise.
>
> Since deceptive alignment is a type of "out-of-distribution" behavior (based on the difference between train and deployment), it has renewed my interest in understanding whether larger models become more brittle OOD. So far the empirical evidence is in the opposite direction, but deceptive alignment is an argument that asymptotically we might expect the trend to flip, especially for tasks with large output spaces (e.g. policies, language, or code) where "drives" can more easily manifest.

So to summarize my takeaways: be more interested in interpretability (especially as it relates to training latent representations), try to identify and study "drives" of ML systems, and look harder for examples where larger models have worse OOD behavior (possibly focusing on high-dimensional output spaces).

Other weird failures. Other weird failures that I think don't get enough attention, even though I also don't think they will play out as written, are Hubinger et al.'s *Risks from Learned Optimization* (AI acquires an "inner objective", somewhat similar to deceptive alignment), and Part I of Paul Christiano's AI failure story (the world becomes very complicated and AI systems create elaborate Potemkin villages for humans).

Paul Christiano's story in particular has made me more interested in understanding how reward hacking interacts with the sophistication of the supervisor: For instance, how much more readily do neural networks fool humans who have 5 seconds to think, vs. 2 minutes or 30 minutes? I more generally want to understand how reward hacking depends quantitatively on both supervision quality and model capacity (qualitatively, we expect higher quality $\rightarrow$ less hacking and higher capacity $\rightarrow$ more hacking). Understanding this quantitative relation would help ground Paul's story, since he imagines a world where humans have built extremely sophisticated systems for supervising ML models, but eventually the ML models become even more powerful and game the supervision signal anyways.

What To Do About Weird Emergent Failures

When thinking about how to handle emergent risks, I often reflect on the example of uranium. For context, an atomic bomb is pretty much just a bunch of uranium put together---once you get enough, the reaction becomes self-sustaining---making it a good example of More Is Different.

The first nuclear reaction (not a bomb, but a pile of uranium in an abandoned football stadium in Chicago) was engineered by Enrico Fermi. The reaction required 12,400 pounds of uranium metal piled 57 layers high. Left unsupervised, a 57-layer pile would consume itself within two hours and kill everyone in the vicinity. On the other hand, a 56-layer pile would do nothing.

Fermi had a good understanding of nuclear physics and understood, from careful monitoring and underlying theory, that the pile would pass the critical threshold between layers 56 and 57. He also knew that cadmium rods would absorb neutrons and strongly inhibit the reaction. These rods were set up and the entire apparatus was carefully controlled to go only slightly supercritical. He brought the reaction to half a watt for several minutes before shutting it back down (see *The Making of the Atomic Bomb*, pp. 524).

With AI, we currently lack both Fermi's conceptual understanding of the underlying risk factors and his ability to continuously measure them. We have neither a cadmium rod nor a measure of reaction criticality. But I think we can get there, by combining these weird thought experiments with carefully chosen empirical experiments, which will be the topic of the next post.

1 Things are more complicated in reality, since θ_t is updated even when a_t is optimal (due to noise in the training process). However, we'll ignore this for purposes of the example. ↩

2 Of course, there is still some distribution shift, since the agent can observe whether it is being trained or deployed. But this is a relatively minor and unintuitive shift compare

to what is typically studied. ↩

3 Of course, emergence doesn't mean that we can just predict whatever we want–we'd need some reason to expect these specific capabilities to emerge. Long-term planning and environmental awareness are both useful for a wide variety of tasks, making them likely to emerge when training powerful models on a diverse data distribution. ↩

Jacob Steinhardt ▾

2 Comments

Sign in to join the conversation.

Shen Zhuoran 4 months ago

I assume the text before footnote 2 meant to say "train and test inputs"? Got me confused fit a while LoL.

♡ 0

Cinera 1 month ago

There's an equation here that I don't understand:
$$a_t = \operatorname{argmax}_{a} \mathbb{E}[\sum_{s=0}^{\infty} \gamma^{-s} R(a_{t+s}) \mid a_t = a]$$

Particularly, I'm confused about why it's γ^{-s} and not γ^{s}?

AIUI, γ is the discount factor and $\gamma \in [0, 1]$.

I'm having trouble making sense of the equation as written assuming a reinforcement learning context.

♡ 0

← Previous Post

Next Post →

Bounded Regret