Techniques for optimizing worst-case performance

Paul Christiano Follow
Feb 1, 2018 · 10 min read

If powerful ML systems fail catastrophically, they may be able to quickly cause irreversible damage. To be safe, it's not enough to have an average-case performance guarantee on the training distribution—we need to ensure that even if our systems fail on new distributions or with small probability, they will never fail *too* badly.

The difficulty of optimizing worst-case performance is one of the most likely reasons that I think prosaic AI alignment might turn out to be impossible (if combined with an unlucky empirical situation).

In this post I want to explain my view of the problem and enumerate some possible angles of attack. My goal is to communicate why I have hope that worst-case guarantees are achievable.

None of these are novel proposals. The intention of this post is to explain my view, not to make a new contribution. I don't currently work in any of these areas, and so this post should be understood as an outsider looking in, rather than coming from the trenches.

Malign vs. benign failures and corrigibility

I want to distinguish two kinds of failures:

- "Benign" failures, where our system encounters a novel situation, doesn't know how to handle it, and so performs poorly. The resulting behavior may simply be erratic, or may serve an external attacker. Their effect is similar to physical or cybersecurity vulnerabilities—they create an opportunity for destructive conflict but don't systematically disfavor human values. They may pose an existential risk when combined with high-stakes situations, in the same way that human incompetence may pose an existential risk. Although these failures are important, I don't think it is necessary or possible to eliminate them in the worst case.

- "Malign" failures, where our system continues to behave competently but applies its intelligence in the service of an unintended goal. These failures systematically favor whatever goals AI systems tend to pursue in failure scenarios, at the expense of human values. They constitute an existential risk independent of any other destructive technology or dangerous situation. Fortunately, they seem both less likely and potentially possible to avoid even in the worst case.

I'm most interested in malign failures, and the narrower focus is important to my optimism.

The distinction between malign and benign failures is not always crisp. For example, suppose we try to predict a human's preferences, then search over all strategies to find the one that best satisfies the predicted preferences. Guessing the preferences even a little bit wrong would create an adversarial optimizer incentivized to apply its intelligence to a purpose at odds with our real preferences. If we take this approach, incompetence does systematically disfavor human values.

By aiming for corrigible rather than optimal behavior (see here or here) I'm optimistic that it is possible to create a sharper distinction between benign and malign failures, which can be leveraged by the techniques below. But for now, this hope is highly speculative.

Amplification

I believe that these techniques are much more likely to work if we have access to an overseer who is significantly smarter than the model that we are trying to train. I hope that amplification makes this possible.

It seems realistic for a strong overseer to recognize an (input, output) pair as a malign failure mode (though it may require a solution to informed oversight). So now we have a concrete goal: find a model that never gives an output the overseer would diagnose as catastrophically bad.

Historically researchers in the AI safety community have been extremely pessimistic about reliability. I think part of that pessimism is because they have been imagining working with models much smarter than the overseer.

Techniques

I'll describe three categories of techniques:

- Adversarial training
- Verification
- Transparency

Previous versions of this list included implicit ensembles, e.g. Bayesian neural nets. I still think there might be useful techniques in that space, but I've become pessimistic enough that it no longer makes the list.

Adversarial training

(*Examples: Explaining and Harnessing Adversarial Examples, Adversarial Machine Learning at Scale.)*

If we want to know whether our model ever fails catastrophically, it's natural to train an adversary to search for catastrophe-inducing inputs. If the adversary can't find one, then we hope that none exist. In order to produce a catastrophe-free model, we can integrate the results of testing into the training process.

The basic problem with this strategy is that no matter how hard you test, it's always possible (realistically, it's likely) that there are possible failure cases your adversary didn't notice. For example, suppose that it takes 2^N seconds to find the Nth failure. Because nature will be many times more complex than our training procedure, it's inevitable that there are many types of failures that nature can find but we can't.

We can attack this problem by trying to make the adversary's job easier. Some possibilities:

- Use interpretability techniques (see below) to help the adversary identify possible failures.
- Use interpretability to expand what counts as catastrophic—if the adversary can get the model to think in a way that looks like it *could* lead to catastrophic behavior under different situations, that could count as a success for the adversary.
- Combine with verification (see below) to help guide the adversary's search and expand the space of allowed attacks.

- Allow the adversary to directly control some of the internal computation of our model rather than merely controlling inputs. This requires a way to constrain the actions of the adversary—we want them to be able to produce any failure that could arise naturally, but if we give them too much power then they can trivially cause a catastrophic failure.
- Change the system's inputs so they are easier for the adversary to fake: see the "distorting observations" and "abstracting observations" sections of this post.
- Make the adversary more powerful, without changing their task, by applying amplification.

Adversarial training faces other difficulties as well. For example, it might take a long time for adversarial training to converge, or training might never converge due to non-convexity. I think these are significantly less likely to be deal-breakers, but they are certainly important research problems if we actually want to apply adversarial training.

Verification

(Examples: Certified Defenses against Adversarial Examples, Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks)

In verification the goal is to find a certificate that there is no input on which a system behaves badly.

Verification has (at least) two problems:

- **Complexity**. It's really, really hard to verify properties of a computationally interesting model.
- **Specification**. It's unclear what we should be trying to verify.

Handling complexity is definitely challenging. But if we are free to train the model in order to facilitate verification, and if we are only interested in certifying some "easy" property that the model satisfies with slack, then it's not clearly doomed.

Prima facie, specification looks more like an unfixable deal-breaker. In the rest of this section I'll give three possible approaches for finding specifications. I think none of these is satisfactory on their

own, but they leave me optimistic that verification can be useful despite specification difficulty.

My current view is that specification and complexity are both potentially surmountable difficulties, but that it's likely their resolutions will result in partial solutions that need to be combined with other approaches.

Use a slow model of overseer as a specification. If our overseer is a physical process, to actually get a specification we'd need to learn a model of the overseer. So we've just shifted the problem to the model of the overseer.

Fortunately, this move does potentially made the problem one step better. Because we don't need to use the overseer model at runtime, we can afford to use a much slower model. So verification could distill a very slow reliable model into a fast reliable model.

Amplification. If our overseer is produced by amplifying some process H, and we have a trusted model of H, then we can obtain a trusted model of the overseer. So if we were able to form a trusted model of the first step of amplification, then we could iteratively use verification to construct trusted models at each subsequent step.

We are still left with the problem at the first step of amplification. But at this step we might be able to restrict attention to a small range of questions for which worst-case guarantees are more feasible; we may be able to train weaker models for which worst-case guarantees are more feasible; or we might be able to eliminate the human altogether and begin amplification from some explicit simple core of reasoning. Any of these seems plausible.

"Verification" without specification. I find it helpful to think of verification from the primal/dual perspective, which also makes the connection to adversarial training clearer.

In adversarial training, the adversary wants to find an input on which the system behaves poorly. The dual to the adversary is an explicit list of all inputs, showing that each one of them performs well. The dual is exponentially large, and (relatedly) the adversary's problem is exponentially hard.

We can view verification in a given proof system as making the adversary's job easier, by expanding the set of allowable "attacks." If

we do this carefully, we can make the dual certificate much smaller and make the adversary's task much more tractable.

For example, if our proofs (dual solutions) are in first order logic, the adversary's task is finding any model of the axioms in which there exists an attack. (This is only different in the infinite case.) If our proofs are sum of squares proofs, the adversary's task is finding a pseudodistribution over attacks. For certain linear programming proofs, the adversary's goal is finding an attack on a particular extrapolation of the model. And so on.

On this perspective, we don't necessarily need an actual specification. Instead, we can take some external catastrophe-checker, relax it to a catastrophe checker over the adversary's relaxed action space, and then try to implement an adversary against that catastrophe checker. This would change the meaning of the dual, which would now have a subtle statistical interpretation rather than being a traditional certificate.

Transparency

(*Examples: Feature Visualization, Deep Inside Convolutional Networks)*

While training a model, we are free to look inside and "see what it's thinking." If we could actually understand those thoughts, we might be able to more easily predict under what conditions it would fail malignantly.

A (weird) example: suppose that human evolution was part of an alien project to train a mathematician, and some humans are extracted from our simulation and trained to solve alien math problems. If aliens are only able to test the humans' behavior they might conclude that they always faithfully solve whatever problem they are posed—especially after a few generations of selection removed all the humans who didn't do so. But if the aliens were able to look inside the humans' heads they might discover that this behavior is extremely contingent—perhaps humans are only solving problems because they believe they will die otherwise, and the humans would behave very differently as soon as they saw an opportunity to escape.

That example is exotic and somewhat implausible, but I think that the same basic dynamic is likely to apply in much messier practical situations. If a model would fail malignantly under some conditions, I

think there is some sense in which the model must be testing those conditions during training.

Malign vs. benign failures: I think the prospects for using transparency to detect malign failures in models trained by gradient descent are *much* better than the prospects for detecting benign failures, or for detecting failures in arbitrary models.

The key point is that a malign failure requires leveraging the intelligence of the model to do something actively bad. If our model is trained by gradient descent, its behavior can only be intelligent when it is exercised on the training distribution—if part of the model never (or very rarely) does anything on the training distribution, then that part of the model can't be intelligent. So in some sense a malign failure mode needs to use a code path that gets run on the training distribution, just under different conditions that cause it to behave badly.

That means that we don't necessarily need to analyze our model in the abstract, we can look at its behavior on actual training data. This looks much more likely to be tractable. (Though this entire discussion is extremely subtle, and without being more precise it's very easy for us to go wrong.)

Amplification: Transparency will be hard if we ultimately want to train models much smarter than we are, but deeply understanding what a model is thinking probably requires being much smarter than it. I think that problem can potentially be overcome by using amplification to stay ahead of the models we are training. This means that current work on transparency, which considers models doing tasks that are within the abilities of the humans who are trying to understand them, could potentially remain relevant even as AI improves significantly.

What do you do with transparency? Merely understanding that a model might behave catastrophically could be useful, but it would be much nicer to actually fix the problem. Adversarial training gives a natural mechanism: once we understand a failure we can synthesize appropriate data and then train on that data.

This approach puts significantly more stress on our transparency techniques. Even if were initially able to use transparency to see how our model might fail, after we perform many generations of selection we might weed out exactly the comprehensible failures and leave the incomprehensible ones. You would only want to apply this technique

if you had a great deal of faith in your methods; if you were feeling at all shaky about your ability to achieve worst-case guarantees, and transparency techniques let you see one potential catastrophic failure, it would be better to consider that a near-miss and seriously rework your project rather than plowing on.

Conclusion

Making ML systems work in the worst case is hard, even if we are only concerned with malign failures and have access to an overseer who can identify them. If we can't solve this problem, I think it seriously calls into question the feasibility of aligned ML.

Fortunately there are at least a few plausible angles of attack on this problem. All of these approaches feel very difficult, but I don't think we've run into convincing deal-breakers. I also think these approaches are complementary, which makes it feel even more plausible that they (or their descendants) will eventually be successful. I think that exploring these angles of attack, and identifying new approaches, should be a priority for researchers interested in alignment.