Some thoughts on training highly reliable models

Paul Christiano Follow
Oct 22, 2016 · 7 min read

I am interested in building machine learning systems which are very unlikely to fail catastrophically, even on rare or adversarial inputs.

This problem seems quite hard; if it's possible, it will probably need to use methods that exploit the internal structure of our models, rather than being able to treat out-of-distribution detection or adversarial training as an independent post-processing step.

Identifying vs. handling problematic inputs

I'm going to talk about *identifying* problematic inputs rather than handling them gracefully. Obviously handling them gracefully would be more satisfying. But I suspect that flagging problematic inputs will be an important first step. If we can flag problematic inputs, then:

- We can try to generate maximally-problematic inputs during training (as in adversarial training).
- At test time, we can filter/abstract problematic inputs, and can search for non-problematic abstract versions.
- (We discuss some other responses in Concrete Problems in AI Safety.)

I don't currently see a plausible *general* approach to handling problematic inputs that doesn't go through the intermediate step of either recognizing or sampling them.

1. Inadequacy of density modeling

If we train a learning system on some distribution D, it might behave quite badly on inputs from a different distribution D′. If it would be great to notice such distributional shifts and to flag inputs on which our model might behave badly.

I've heard the following proposal a few times:

- Fit a density model p to the training distribution D.
- Whenever you encounter a data point that is too far from the training distribution, flag it as potentially problematic. For example, we might flag x whenever -log p(x) ≫ entropy(p).

We could define "too far from the training distribution" in many alternative ways.

But as far as I can tell, no variant of this approach will be sufficient for building a robust system.

Adversarial inputs

Adversarial inputs necessarily have a high *ratio* between the probability q(x) under the adversarial distribution and the probability p(x) under the training distribution. But they need not look exceptional at all if we consider just the training distribution.

For example, consider a distribution over high-dimensional images where the camera adds a bit of Gaussian noise on each pixel. If the images are very large and we haven't done any adversarial training, it seems likely that we can produce an adversarial example with a very small per-pixel perturbation, e.g. with the fast gradient sign method. (The perturbations may be small relative to the standard deviation of the Gaussian noise during training.)

Under Gaussian noise, these small perturbations are just about as probable as any other perturbation. The weird thing about the perturbation is that it has an extremely high correlation with the gradient of the prediction—as far as the data distribution p is concerned, this fact is totally invisible.

Of course if we had access to the adversarial distribution, or could train a generative model q on adversarial examples, then we could easily tell that this was an adversarial input. Similarly, if we had access to the features that the adversary was using to generate their attack then we might easily notice that these features were taking on very abnormal values. But these techniques only work when we have *specific hypothesis about how our model may not be robust*.

The hardest (and also typical) case is when we don't know exactly what kind of inputs may trip up our model at test time, or where accessing the problematic distribution is too expensive to be

integrated as part of the training process. It seems like out-of-distribution detection is not enough to handle these cases.

Non-adversarial robustness

The discussion above is clearest in the context of adversarial examples, but I think the situation is the same for other kinds of distributional shift.

For simplicity, let's analogize our training and test distributions to Gaussians. We can potentially detect the case where the test distribution is "bigger" than the training distribution, supported on points that have negligible density during training. But from the perspective of robustness and generalization, we should be just as worried when the test distribution is "smaller" than the training distribution, supported on some tiny subset of the training distribution (whose total probability was negligible).

If we visualize the low-dimensional setting, it is easy to imagine that we will be able to handle the case of "smaller" test distributions using some kind of interpolation, while "bigger" test distributions demand a more challenging extrapolation. But that intuition seems to completely break down in high dimensions—being focused on a small part of the distribution can make feature expectations just as extreme as if the test distribution were located in a totally different part of the space, even for linear features. (This is illustrated especially cleanly by adversarial examples, but I think the phenomenon is pretty universal.)

Lots of data

If we can collect a lot of test data, then it becomes more straightforward to notice whether the training and test distributions differ. For example, we could simply train a discriminative model to distinguish them.

I am not satisfied with this approach, because I want systems to recognize and respond to catastrophe-inducing inputs immediately, I don't want to wait until we have gotten enough of them that we can train a discriminative model.

This technique may be appropriate when there is a particular moment when we are concerned that our distribution will shift, for example when applying an algorithm in a new place. But it does not seem appropriate when we are concerned about distributional shifts during normal operation or about outliers.

2. Rare failures can occur without distributional shift

People often talk somewhat interchangeably about distributional shift and non-robustness. But if we are interested in models that avoid rare catastrophes, then I think that we need to worry about some cases where there is *no* distributional shift.

For example, suppose that we are training a model to evaluate the financial cost of an earthquake. We train our model on many earthquakes, and it performs extremely accurately. But if we evaluate on a 100x larger test set, then by chance alone the test set may contain earthquakes an order of magnitude bigger than anything we encountered in training.

Our model may not be able to make a reliable prediction about this earthquake, despite the fact that the test and training distributions are identical. Even though such errors will necessarily be rare, in some contexts they may be catastrophically bad.

In some sense this seems like an "unfair" problem—how can you expect to get failure rates much lower than 1/N if you are only training your model on N observations? But I think that in the real world there are settings where failure is very expensive, and 1/N is an unacceptable failure rate. For example, if we train a system on 1 month of data, then a failure rate of 1/N corresponds to one arbitrarily-catastrophic failure per month. Even if we train on a decade of data, 1/N corresponds to a failure rate of 10%/year.

Either we need techniques for driving error rates below this threshold, or we will need to ensure that machine learning systems are only deployed in settings where there is no possibility of catastrophic failure. Redundancy is often used in order to obtain very low failure rates for systems involving humans; but for machine learning systems, different systems trained on data from the same distribution may tend to fail simultaneously, and so it might not be so easy to avoid catastrophic failure.

Outlier detection

We could hope to detect outliers in a model-independent way. For example, there is a clear sense in which the biggest earthquake ever is an outlier.

However, if our model had access to enough kind of different data about the world, there will be a constant stream of "outlier" events. If we want to identify problematic inputs without having an unlimited number of false positives, we really need to know that the big earthquake is the kind of outlier that is relevant to our model.

3. Some other angles

I don't think that it is possible to train robust models by treating out-of-distribution detection or adversarial training as an external postprocessing step that uses the model as a black box. Instead it seems like we will need to look inside the models we are training, or perhaps train models which are able to directly report when they are uncertain. Such models might be able to cope with both changes in distribution and problematic rare inputs.

One approach is to maintain a distribution over models, and flag an input as uncertain when its label is substantially correlated with our model uncertainty.

(Another approach, that I won't talk about here, is to look at features used internally by the model. If we check for anomalies in those feature values, then maybe we can identify the most problematic inputs.)

The hope is that when an input *isn't* flagged as uncertain, then there is essentially "consensus" amongst the plausible models. Depending on how wide a distribution over models we can maintain, and where we set the threshold for model uncertainty, this might allow us to draw strong conclusions: rather than "our current model predicts X" we could hope to say something like "all simple models consistent with the data agree on X."

I think there are lots of challenges with getting the kinds of guarantees that we actually want—existing approaches don't yet do the job. For example, I don't believe that any existing approach is able to robustly flag adversarial examples as uncertain in vision tasks. But I think this is a promising direction and I'm excited to see what kind of results will appear over the coming years.

Conclusion

There has been a lot of research on training robust models, and I'm excited to see where it goes over the coming years. If aiming for the

ambitious goal of completely avoiding catastrophic failures, I think it is worthwhile to keep in mind the possibility of data that *could* appear in training but which is sufficiently rare that it happens to have never shown up (yet sufficiently common that it isn't too surprising to see it at test time), rather than thinking of robustness as only occurring when the training and test distributions are different.