CS 165: Foundations of Machine Learning **Winter 2019**

Lecture 12: Robust Deep Learning

Lecturer: Anima Anandkumar *Scribes: Hao Liu*

12.1 Recap on last lecture: Generalization

Classicial theory of bias and variance tradeoff:

- High capacity model overfit the data and don't generalize
- Under capacity model underfit the data, has high bias and also don't generalize well
- There exists a middle ground optimal model capacity which the bias and variance are optimally traded off

Modern Neural Network Theory:

- U curve is not true, it's true the number of parameters is roughly smaller than the number of samples you have
- There exist an over parametrized region where the training loss goes to 0, the test error still goes down with the model complexity increase
- Reason is classical theory(like VC bound) care about the worst case, the highly complex model may not generalize well. But training with SGD performs an implicit algorithmic regularization, won't go to that worst case, tend to achieve a better norm of the weights of the networks and generalizes better.

Open problem

- Design good regularization of the system
- How to achieve global minima?

Big models are good but have huge compute requirements. And as accuracy becomes higher, the computing requirement grows larger. Not clear how to overcome that.

12.2 Robust Deep Learning

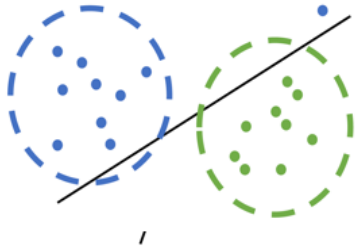
So far we assumed that the test or train set is drawn from the same distribution, but it's far from true. And can we train the system to avoid the worst case example?

ML predictions are (mostly) accurate but brittle, attacks fool the machine learning system easily in the way that human can easily overcome. Today we focus on how to attack and defense machine learning systems.

12.2.1 Data Poisoning Attack

Goal: Maintain training accuracy but hamper generalization

The attacker has access to the training dataset, and add additional bad data points to the dataset to "poison" the dataset. The number of additional data points is limited to restrict the ability of the attacker.



- The idea is not new. Robust statistics had studied it before, where the guarantee to be robust to them exist.
- Recall Robust PCA:
  - PCA:
  find low dimensional subspace of data by looking at the top vectors of the covariance matrix. A few outliers can severely affect the method.
  - Robust PCA:
  Find low-rank subspace and try to pick out the outlinear at the same time. Relax the Assumption: some data points cannot fit in the low-rank subspace, and these data points are sparse. Convex Relaxation method: require one matrix to be low rank, the other to be sparse
  - Non-convex Robust PCA:
  Do a simple alternating projection: Maintain some estimates of what the low rank and sparse components are, and iteratively look at the residual and project onto low rank and sparse objects.
  Result: Can get to the optimal recovery under some conditions which match with the convex method.

What classical method did: Detect the outlinear and classify/dimension reduction(PCA) on the rest of the data.

In deep learning, we use highly nonlinear classifier with lots of capacity, the goal changes to maintain training accuracy but do the classification of specific inputs (based on where the poisoning data point is)

The important and main difficulty of robustness: What is an attack model? For example in the data poison case, how can the model add the additional data point? Does it have to be far away from the original data?

Research has found that a single poisoned input to the training data can affect the prediction result severely.

How to detect these data poisoning and to completely avoid them is still an open problem.

Three commandments of Secure/Safe ML:

- You shall not train on data you don't fully trust(because of data poisoning)
- You shall not let anyone use your model(or observe its outputs) unless you completely trust them(because of model stealing and black box attacks)

- You shall out fully trust the predictions of your model(because of adversarial example)

Two settings: White-box: you know what the classifier is. Black-box: you don't know the classifier(need to query the model)

12.2.2 Adversarial Attack

$$\max_{\delta \in \Delta} Loss(x + \delta, y; \theta)$$

where $\delta \in \Delta$ is typically the norm bound, and given the sample x, what is the worst perturbation δ that maximize the Loss function.

Part I: The maximization is the attacker to figure what is the worst case perturbation to create an adversarial example.
Part II: Outside Minimization is the defense to train a robust classifier

$$\min_{\theta} \sum_{x,y \in S} \max_{\delta \in \Delta} Loss(x + \delta, y; \theta)$$

The loss function is a natural way to think about it you are going for the worst case.

It's a two player game between attackers and defenders. And in current research, attackers are winning, train a robust classifier is hard.

Adversarl side: How to do the inner maximization problem

$$\max_{\delta \in \Delta} Loss(x + \delta, y; \theta)$$

Non-convex, many standard things to do:

- Local search(lower bound on objective)
- Combinatorial optimization(exactly solve objective)
- Convex relaxzation(upper bound on objective)

Projected gradient descent:
-Used when the perturbation is bounded
-Fast Gradient Sign Method(FGSM): If take l_∞ ball as the bound, and the projection function becomes clipping, which results in a box constraint. And let step size go to infinity, we get the fast gradient sign method, where the optimal noise is according to the sign of the gradient, which is like gradient ascent (reverse gradient).

Adversarial training:
-Do mini batch
-Do local minimization and local maximization iteratively
-Get the noise according to reverse gradient for inner maximization
-Do the outer minimization also as a local update on mini batch

The experiment shows that achieve more robustness, but can oscillate and no guarantee and proof. And it's not necessarily the right thing to do it, people do it because of simplicity.

Back to Inner maximization:
Exact combinatorial optimization:
Convex outerbound, provable but more expensive.

The non-linearity of deep neural networks make the output become non-convex, but we can design suitable convex outerbound of these non-convex outputs, and these outerbound don't violate the decision boundary, and these convex sets are robust(can prove the entire set is robust).

Bounds can be loose, trying to design these may be too large convex outbound, and can harm the accuracy

Several ways to achieve the convex outerbound:
Main idea: to come up with a modified loss function to find the worst case of perturbating a sample

- Do the convex relaxation and solve it in the primal
- Go to the dual space, cheaper than doing it in the primal
- Iterative lower and upper bounds, more computationally feasible

12.2.3 Robustness versus Generalization

Generalization: Assumption that the training and test data are iid samples is very artificial, and the real world may not happen
Robustness: the Adversary looks at the worst case and wants the classifier to fail.
What's the connection between these two? If you want the classifier to be robust, what happens to generalization?

Do robust deep networks overfit?

Two different setups of the test set: (1) iid sample as the training set (2) the test set is adversarially perturbed

But find Adversarial training / Adversarial testing results in a large generalization gap
Also, Adversarial training doesn't help the standard generalization much

Robustness is a different thing from generalization because generalization is average over the distribution, but robustness is asking for the worst case. If you are asking the worst case, you will hurt the average case.

Generalize for the worst case is still very hard to do: Adversarial perturbation on test data: the space is very big and has too many possibilities, hence it's very hard to generalize on that space. And it's still an open problem.

Adv. Robust Generalization needs more data:
Need significantly more samples to generalize in an adversarial setting:

You can't have both Robustness and Generalization at the same time, no free lunch.
Robustness: Discarding certain correlations
Generalization: Explore all these correlations

Then how to Incorporate multiple objectives?

Robustness has some benefits: If you look at the gradient of the robustly trained classifier, empirically robustness forces the model to focus its gradient on the relevant part.

Why can't robust scale:

- Need more data (most important)?
- Better tighter convex bound?
- Better architecture?
- Are there easier attack to overcome?
- Are there ways to overcome these data requirements? Data augmentation?

References