Adversarial Examples in Machine Learning

CS 261 Lecturer: David Wagner Scribe: Arun Ganesh

March 31, 2021

1 Introduction

In machine learning, we often train neural networks on many examples to solve classification problems. The end result is a model f_θ, with some parameters θ, that takes in an example x and outputs a prediction (a label or a distribution over labels) $y = f_\theta(x)$. There is a loss function ℓ that takes in a prediction and label and outputs the loss of the prediction on the label. Our goal during training is to solve the problem $\min_\theta \sum_i \ell(f_\theta(x_i), y_i)$, where x_i, y_i denotes the ith example and label respectively, and the sum is over examples in the training set.

An adversarial example arises when we can make a perturbation to an example x that causes f_θ to misclassify x, but the perturbation is ideally imperceptible to humans, or at least would not cause a human to misclassify the example. For example, a picture of a school bus, after changes that humans can't see, can become classified as a hummingbird. Some other examples that are security risks in that they could cause a self-driving car to get into an accident:

- A stop sign getting classified as a yield sign.
- Objects that are imperceptible while driving (e.g. an obstacle that gets classified as part of the road).
- Markings that cause a car to think a lane turns one direction, causing it to veer off the road or into another lane.
- Markings on the road that a car thinks are a curb, causing it to swerve into a different lane.
- A "patch" that can be printed and placed a road that is completely flat, but is classified as an obstacle that needs to be driven around.

2 Math Behind Attacks

Let $\Psi_{\theta,x,y}(\delta) = \ell(f_\theta(x+\delta), y)$. The problem of finding adversarial examples is finding $\max_\delta \Psi_{\theta,x,y}(\delta)$ such that $||\delta|| \leq \epsilon$, where $||\delta||$ is an appropriately chosen norm. Informally, we want to maximize how incorrect the classification $f_\theta(x + \delta)$ is, while keeping the size of the perturbation δ to be small. Solving this constrained optimization problem is a bit tricky, so in Carlini and Wagner, they instead solve the unconstrained optimization problem $\max_\delta[\ell(f_\theta(x + \delta), y) - c\,||\delta||]$, where c is an appropriately chosen constant.

We still need to specify what norm we are using in the term $||\delta||$. A popular choice is the ℓ_2-norm. This leads to $||\delta||$ being differentiable with respect to δ. The term $\Psi_{\theta,x,y}(\delta)$ is also differentiable with respect to δ in practice: to train a neutral net we need f_θ to be differentiable so we can run

gradient descent. So for the ℓ_2-norm, gradient descent lets us solve the unconstrained optimization problem.

However, the ℓ_2-norm doesn't have a clean interpretation for what the perturbations do to the examples. In contrast, the ℓ_∞-norm gives a clean interpretation: e.g. for an image, every pixel changes in intensity of its RGB values by at most a small amount. Thus a perturbation δ is likely imperceptible to humans if $||\delta||_\infty$ is small. However, the ℓ_∞-norm of δ does not have a nice gradient, since the gradient will be non-zero for all but the largest coordinate.

The PGD attack instead returns to solving the constrained optimization problem $\max_\delta \Psi_{\theta,x,y}(\delta)$ such that $||\delta||_\infty \leq \epsilon$. To solve this problem, suppose Ψ were a linear function of δ. Then the solution is simply to set each coordinate of δ to either ϵ or $-\epsilon$, depending on the sign of its coefficient in Ψ. Of course, neural nets are complicated functions that are often far from linear. However, we can use first-order Taylor series to view them as locally linear. More formally, the idea of the PGD attack is to take an iteration count I, and do the following I times: Start with $\delta_0 = 0$, and compute $\delta_{j+1} = \delta_j + \frac{\epsilon}{I} \cdot \text{sign}(\Psi_{\theta,x,y}(\delta_j)$, then use δ_I as the final perturbation.

One issue we haven't discussed is that the final adversarial example $x + \delta$ needs to be a valid example. For example, if x is an image, we need the intensities of each pixel's RGB values to be in the range $[0, 255]$. However, if we simply project δ back into the space of valid examples after each update to δ, then gradient descent still converges.

Another issue is that the above optimization problem assumes that we have the ability to perturb the entirety of each example. In practice, an attacker can't e.g. paint the sky or the clothes of strangers walking by to try to attack a camera. However, this constraint is easy to add to the above problems. We now simply enforce that $\delta = 0$ in the region that we can't perturb, equivalently turning Ψ into a function of a lower-dimensional vector corresponding to the coordinates of δ that are allowed to be zero. In turn, we're simply solving a lower-dimensional optimization problem, which can only be easier than the original problem.

3 Defenses

In this section we discuss possible defenses against adversarial examples and their pitfalls.

3.1 Ensemble Defenses

In an ensemble defense, we train multiple models. When classifying an example, we aggregate the predictions of all the models somehow. Thus an adversarial example must fool e.g. a majority of the models to be misclassified.

While this seems robust, mathematically speaking the adversary's job is not much harder. All the adversary needs to do now is use the techniques from the previous section for the loss function ℓ which uses the aggregated model instead of any single model.

3.2 Adversarial Training

In adversarial training, instead of optimizing our loss function, we optimize the worst-case loss under some perturbation set. More formally, rather than solve $\min_\theta \sum_i \ell(f_\theta(x_i), y_i)$, we solve $\min_\theta \sum_i \max_{\delta_i : ||\delta_i|| \leq \epsilon} \ell(f_\theta(x + \delta_i), y_i)$.

This is a harder optimization problem since we have a minimax objective, rather than just a minimization or maximization. In practice, given our model, we might instead use a known attack to generate a fixed δ_i that attacks each example under our model. We then solve the problem

$\min_\theta \sum_i \ell(f_\theta(x_i + \delta_i), y_i)$, which is just a minimization problem. In the next iteration, we use the new model to generate a new set of δ_i.

Adversarial training does pretty well in practice. e.g. without adversarial training, it suffices to change pixel intensities by at most 1 to fool classifiers. With adversarial training, this number goes up to roughly 5. This is a large multiplicative improvement, but note that changing pixel intensities by 5 is still fairly imperceptible to humans.

3.3 Secret Model

A very simple idea is just to not publish our model in any form. However, if the attacker knows the classification problem we are trying to solve, they could simply train their own model on a similar training set. In doing so they will likely arrive at a model similar to our model, and then by attacking their own model generate adversarial examples that also likely attack our model.

3.4 Randomized Smoothing

In randomized smoothing, the defender chooses a noise distribution η and classifies an image based on the distribution $f_\theta(x+\eta)$ rather than $f_\theta(x)$. For example, if f_θ outputs a single label then $f_\theta(x+\eta)$ is a distribution over labels, and our "smoothed" classifier might output $\arg\max_i \Pr_\eta[f(x+\eta) = i]$, i.e. the most likely label in the distribution $f_\theta(x+\eta)$.

One can prove theorems showing that if the smoothed classifier has a large "margin", i.e. the distribution $f_\theta(x+\eta)$ takes on value i with probability at least p, then for all δ such that $||\delta||$ is at most some function of p, the smoothed classifier has the same output on x and $x+\delta$. Put another way, not only do smoothed classifiers give robust predictions, but they can certify that the predictions are robust to a certain perturbation set. This is what is known as "certified security."

4 Discussion (Fred): Synthesizing Robust Adversarial Examples

In discussion, we considered "robust adversarial examples", which in this section mean examples that work in the real world. The idea is that often, we do not classify e.g. a single image, but a series of images corresponding to the view of a camera over a time interval. An adversarial example that works for a single frame in this interval might not be so problematic. For example, a self-driving car that classifies a stop sign as stop sign 99% of the time that the stop sign is in its vision probably will not behave too dissimilarly to one that correctly classifies the stop sign the entire time. So for an adversary to properly attack the self-driving car, it needs to fool the classifier for a reasonably large fraction of this time. During this time, by virtue of the self-driving car moving, we are also applying some number of transformations to the stop sign as an image. For example, by driving closer to the stop sign, we zoom in on it. By changing lanes, we might stretch certain dimensions of the stop sign. By driving on a sloped road, we might rotate our view of the stop sign.

To update our mathematical formulation, we think of there being a transformation layer between the image and the model. Before seeing the image, a transformation is applied to the image, and then the transformed image is passed to the model. For simplicity, we can think of the transformations as being drawn from some distribution. Our loss function is now the expected loss under this distribution, or perhaps the expected loss conditional on being in the "best", say, 51% of the distribution. In Athalye et al., the authors consider attacking models under this formulation. They were able to print 3D objects and slightly perturbed versions of these objects such that e.g. the objects were correctly classified from most angles and rotations, but the perturbed objects were classified incorrectly under almost all angles and rotations.

There are some limitations to the formulation and attacks the author considered. For example, things like light sources and shadows projecting at different angles are hard to model mathematically, and might offer additional robustness against adversarial example. There are also may be many real-world perturbations that the paper might not consider. Finally, the authors only considered images containing 3D objects that have transformations applied to them. Other formats like audio might have different transformations to be considered.

5 Reconstruction Attacks

We conclude with a brief discussion on reconstruction attacks. The idea is that if an attacker has access to the classifier f, they can attempt to reconstruct instances from the training set used to generate f, which may be confidential. The idea is to use gradient descent to search over inputs, and find one that minimizes the loss for a given label. That is, we solve the problem $\min_x \ell(f_\theta(x), y)$ for a target label y.

For example, given a facial recognition model, we could find an input that causes $\ell(f_\theta(x), \text{“}JohnSmith''$) to be low, which might let us reconstruct John Smith's face. As another example, a natural language processing model trained on a corpus of private emails might be used by an attacker to reconstruct sentences from those emails.