Lecture 9: Introduction to Adversarial Examples

October 23, 2019

1 Robustness at test time

So far in this course, we have focused on corruptions to the data at training time. We now turn our attention to a different yet similarly problematic scenario, where the data corruption occurs at test time. We will define what this means formally now.

Let us first define the type of problems we will consider. We consider a classification task over $\mathbb{R}^d$ to a discrete set of classes $\mathcal{Y}$. That is, we are given samples $(X_1, y_1), \ldots, (X_n, y_n)$ drawn from some distribution D over $\mathbb{R}^d \times \mathcal{Y}$, and our goal is to learn some classifier $f : \mathbb{R}^d \to \mathcal{Y}$ minimizing the expected 0/1 loss over D. Formally, the 0/1 loss is given by $\ell_{0/1}(y, y') = \mathbb{I}[y \neq y']$, and the (non-robust) classification task simply is to, given samples, find a f minimizing the *expected loss*

$$R(f) = \mathop{\mathbb{E}}_{(X,y)\sim D} \left[\ell_{0/1}(f(X), y)\right] \ .$$

A case of special interest is where f is a neural network.

The robust version of this problem can be stated as follows: for each $x \in \mathbb{R}^d$, we now additionally specify a *perturbation* set $\mathcal{P}_x$, which should be thought of as perturbations to x, which, when applied to x, should not change the classification of X. For instance, if X is a natural image, we can think of $\mathcal{P}_X$ as the set of "imperceptible" perturbations to X which should not change its semantic content, and hence should not change its classification. Our goal is to develop classifiers which are similarly robust to such perturbations. To that end, we define the *robust expected loss* of a classifier f to be

$$R_{\mathrm{rob}}(f) = \mathop{\mathbb{E}}_{(X,y)\sim D} \left[\sup_{X' \in \mathcal{P}_X} \ell_{0/1}(f(X'), y)\right] \ ,$$

and our goal is to, given samples from D, find an f which minimizes $R_{\mathrm{rob}}(f)$.

Given a particular classifier f, and a specific point (X, y), we say that a point X' is an *adversarial example* for f at (X, y) if $f(X) = y$, that is, f correctly classifies X, but $X' \in \mathcal{P}$ and $f(X') \neq y$, that is, X' is a small perturbation of X on which f fails to correctly classify.

The study of such adversarial examples in ML dates back to at least 2004 [1, 2], and was first investigated for neural networks in [3, 4].

What $\mathcal{P}$ should we consider? A very important question to ask in defining this problem is what perturbation set we should allow. Recall that ideally $\mathcal{P}$ should capture the set of semantically meaningless perturbations to my input X. Let's use the recurring example of image classification on e.g. CIFAR-10 or ImageNet, so that X is an image, and y is a label (e.g. dog, cat, airplane, etc). In different domains, the allowable set of perturbations should of course also change.

A relatively natural family of perturbations, that we will also focus on quite a bit in this class, are the ℓ_p bounded perturbations. Specifically, for any $p \in [0, \infty]$, and any $\varepsilon > 0$, we can consider the perturbation set

$$\mathcal{P}_{p,\varepsilon}(x) = \left\{x' \in \mathbb{R}^d : \|x - x'\|_p < \varepsilon\right\} \ . \tag{1}$$

For neural networks, adversarial examples were first demonstrated for ℓ_∞ [3, 4], and still today it is considered a standard benchmark for adversarial robustness. However, there is also a lot of work on robustness in other p-norms, in particular $p = 0$ and $p = 2$.

But robustness against ℓ_p-bounded adversaries is a bit unsatisfactory. While it is arguably true that perturbations of small ℓ_p norm are all semantically meaningless (and indeed, to humans, the pictures typically look unchanged after perturbation), it is not true that the set of such perturbations captures the set of all semantically meaningless perturbations. For instance, a small rotation of a picture of a dog is obviously still a dog, but the ℓ_p norm between the two pictures can be very large. Alternative definitions have been proposed to try to account for this, but ultimately it is likely just an ill-posed and impossible question to define the "correct" set of perturbations to consider.

Despite this, because we still lack satisfactory robust classifiers against ℓ_p perturbations, the study of adversarial examples in ℓ_p-norms is still interesting. Such a classifier could be viewed as the first step towards a truly robust classifier.

Why do we care? A good question is why we might care about adversarial examples in the first place. Let us continue with the recurring example of image classification. There are two major reasons why one should care:

1. **Real-world risks.** As we are using image classification for more security sensitive things, the fact that they can be fooled by otherwise imperceptible changes presents a real threat to many of these applications. For instance, if we're using deep networks for computer vision in self-driving cars, if the network can be fooled to believe that a car in front of you is not there, then this presents a major flaw to the self-driving car. As we'll see in a bit in this lecture, this is not just some fantastical scenario: similar attacks have already been demonstrated against Tesla's autopilot technology.

2. **Conceptual gaps.** The existence of adversarial examples concretely establishes holes in our understanding of how neural networks work. Dramatically oversimplified, the typical mantra of deep learning states that deep networks are able to learn good feature representations which capture semantic understanding for the image classification task, that is, it learns good reasons why a picture of a cat is a picture of a cat. However, adversarial examples are a counterexample to this (oversimplified) view: adversarial perturbations to cats still look like cats to us, and therefore still has the same semantic meaning as the original image, but the deep network is fooled.

2 Examples of adversarial examples

It turns out that adversarial examples are ubiquitous in deep learning (and beyond!), and to our knowledge, there are no hacks to prevent them. Here we'll list a bunch of settings in which researchers have found adversarial examples.

ℓ_p attacks The "classical" setting, which has been studied quite extensively since [3, 4]. In this setting, adversarial attacks against neural networks that have not been explicitly robustified are extremely effective: nearly 100% of the images in the test set have adversarial perturbations for relatively small ε, say $\varepsilon = 2/255$ for ℓ_∞ on CIFAR-10.

Other perturbations As mentioned above, we can consider other perturbation sets as well. A surprisingly simple type of perturbation that can already fool convolutional neural networks is the set of perturbations that consist of a single rotation and a translation [5]. In this paper, they construct attacks of this form that drop the accuracy on CIFAR-10 down to less than 3%. More recently, more general spatial notions of robustness have been studied, for instance, adversarial examples in Wasserstein distance [6].

Black-box attacks So far the attacks that we have presented have all assumed knowledge of the underlying classifier while constructing the attack. One might hope that this is inherent: this way, if we could somehow obfuscate the model, perhaps we could mitigate the real-world security threat of adversarial

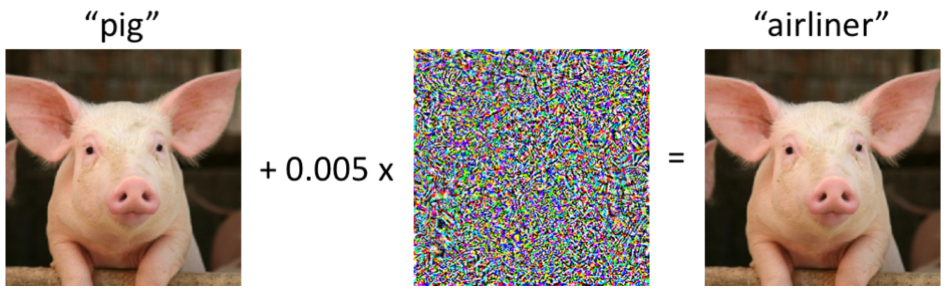

Figure 1: An adversarial perturbation for ℓ_∞ with $\varepsilon = 0.005$ after all the pixels are normalized to be in the range $[0, 1]$. Picture taken from `http://gradientscience.org/intro_adversarial/`

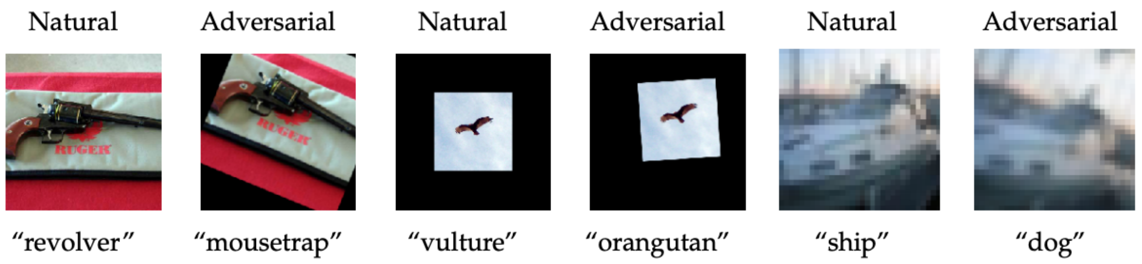

Figure 2: Fooling CNNs with a rotation and a translation. Figure taken from [5]

attacks. Unfortunately, this is not the case: [7] demonstrated that adversarial examples can be made to transfer across architectures. They design adversarial examples that fool an ensemble of known models, and demonstrate that these same adversarial examples will also be adversarial for other, unknown, deep architectures.

Real-world attacks It's also been demonstrated that adversarial examples are not just a digital phenomena: they also can persist when applied to real-world images [8]. There have been many demonstrations of this: [9] produced a 3D printed sculptures of a turtle that was consistently misclassified as a rifle from all angles. The paper [10] designed 3D printed glasses that can fool facial recognition. There have also been real world attacks on object detection [11, 12]. The list is almost endless.

One notable setting in which real-world attacks have been demonstrated are against self-driving cars. The paper [13] produced (amongst other things) graffiti-like patches to put on a stop sign that make it look like a speed limit sign. Recently, researchers at Tencent were able to get Tesla's autopilot to either turn on the car's wipers, or to change lanes [14]. According to Tesla, the former vulnerability has been patched.

Attacking NLP models The paper [15] demonstrated how to fool reading comprehension systems by adding distracting sentences to the paragraphs. The problem in NLP is a bit different than in image comprehension: in NLP the objects are far more discrete, and so it's not so obvious how to even define what it means to be a small perturbation. See e.g. [16] for a more extensive survey of recent results in this area.

Attacking deep RL As one final example, the paper [17] demonstrates adversarial policies for deep RL. They show that for a number of two player games, where one player plays using an RL strategy, there exist adversarial policies that look random yet cause the RL policy to fail spectacularly.

Of course there are many more settings as well. This list hopefully makes the point that such attacks are widespread, and are surprisingly powerful. As we'll see later on, these attacks are also quite hard to defend.

3 How to generate attacks

Given a classifier f and a point (X, y), how does one go about finding adversarial examples for f at X? In other words, how do we find a solution to the following problem:

$$X^* = \arg\max_{X' \in \mathcal{P}_X} \ell_{0/1}(f(X'), y) \; . \tag{2}$$

Brute force search Of course, the most powerful way to optimize (2) is just brute force search: just find the perturbation in the set that maximizes it by trying all perturbations, or trying a large net over possible perturbations. However, typically this is computationally infeasible, as the size of the net would be exponential in d. But when the family of perturbations can be parametrized by a small set of parameters, one can sometimes just search over all possible parameters. For instance, in [5], their perturbations were parametrized by a single rotation and translation. Thus to find their adversarial examples they were able to simply enumerate over all possible pairs of rotations and translations to the image.

Projected gradient descent A very popular type of adversarial attack against deep networks go via projected gradient descent (PGD for short). Let's take the case where $\mathcal{P}$ is the ℓ_∞ ball around x.

We wish to apply a gradient operator to do a first order update to the above objective. However, the 0/1 loss is far from continuous. To get around this, notice that a neural network is secretly a *soft* classifier. That is, not only does it predict the most likely class of X in $\mathcal{Y}$, but it also gives a likelihood for every $y \in \mathcal{Y}$. In other words, f can be thought of as a map from $\mathbb{R}^d$ to $\Delta(\mathcal{Y})$, where $\Delta(\mathcal{Y})$ is the set of probability distributions over $\mathcal{Y}$. This corresponds to the output of the last layer of the neural network, after the softmax has been applied to the raw logits. Moreover, the neural network is trained to minimize the logistic loss L of (X, y). So a natural replacement for the above loss is to simply find a perturbation which maximizes the logistic loss of X, subject to the constraint that it lives within $\mathcal{P}_X$:

$$X^* = \arg\max_{X' \in \mathcal{P}_X} L(f(X'), y) \; . \tag{3}$$

The nice thing about this objective is that it is now differentiable as a function of X'. Therefore, a natural heuristic for maximizing it, subject to the constraint that $X' \in \mathcal{P}_X$, is simply gradient ascent, projected so that every step lies within $\mathcal{P}$. Formally, let Π be the projection onto $\mathcal{P}_X$. Then, for some step-size schedule η_t, we can define the sequence of updates $x_0 = X$, and

$$x_{t+1} = \Pi \left(x_t + \eta_t \nabla_x L(f(x_{t+1}), y) \right) \; . \tag{4}$$

The update (4) is the PGD update. To find an adversarial example, we simply apply this update some (relatively small) number of steps, and often the resulting x_t will be an adversarial example. While this does work well in practice, notice that we cannot typically prove anything about the convergence of this algorithm, as the objective is highly non-concave.

When the perturbation set is the ℓ_2 or ℓ_∞ ball, the projection step is quite easy to handle. For more general sets, this can be trickier. In the case of ℓ_∞, the update

$$x_{t+1} = \Pi \left(x_t + \eta_t \cdot \operatorname{sgn} \left(\nabla_x L(f(x_{t+1}), y) \right) \right) \tag{5}$$

is often called the PGD update, where $\operatorname{sgn}(\cdot)$ is the function which takes a vector v, and maps every entry to its sign. This is a slight abuse of terminology as it is more properly a non-linear form of PGD, but the name has become widespread. As another remark, if one only applies the update (5) once, this is known as the *fast gradient sign method* (FGSM for short) [18]. Unsurprisingly, this method typically works worse than PGD, but the main advantage is that it is substantially faster, since it only does one iteration.

Convex surrogate-based techniques A more general type of attack replaces (2) with a nice surrogate function that is more well-suited for optimization, as proposed in [19]. That is, we replace $\ell_{0,1}$ with some function $L(x, y)$ which is negative if and only if $\ell_{0/1}(f(X), y) = 1$. One can then consider the objective

$$\min_{x'} \mathcal{D}(x, x') + \lambda \cdot L(x', y) \ . \tag{6}$$

As before, this objective is minimized via first order methods. Here $\mathcal{D}$ is some regularization term that is large when x' is very far from $\mathcal{P}_x$. For instance, for ℓ_p norms this can literally be the ℓ_p distance (usually raised to the p-th power to make optimization easier) between x and x'. In the context of image classification, one often adds the constraint that $x' \in [0, 1]^d$, as the pixels are normalized to be within $[0, 1]$ and so any point outside this domain is not a valid image.

Dealing with real-world noise In physical attacks, an attack trained naively with PGD on digital examples will not often work as-is. This is because the image that is ultimately captured by the camera before being processed by the neural network has a lot more noise added to it, because of different angles, different lighting conditions, etc. To deal with this, [9, 11] propose to simulate this randomly while training the adversarial perturbation, a technique called *Expectation over Transformation* (EOT) or RP2. In the Lagrangian formulation, this corresponds to the maximization problem

$$\min_{x'} \mathop{\mathbb{E}}_{t \sim \mathcal{T}} \left[\mathcal{D}(t(x), t(x')) + \lambda \cdot L(t(x'), y) \right] \ , \tag{7}$$

where $\mathcal{T}$ is a distribution over transformations over which we hope that the adversarial perturbation is robust. This for instance can be different lighting patterns, angles, etc. In other words, the adversary controls x', but before it is fed to the network, it is manipulated by a random transformations. Moreover $\mathcal{D}$ can be taken to be a more refined version of closeness than just ℓ_p closeness, that captures visual similarity better. To optimize this objective, we simply apply projected SGD rather than PGD, by randomly sampling a transformation from $\mathcal{T}$ at every iteration.

References

[1] Nilesh Dalvi, Pedro Domingos, Sumit Sanghai, Deepak Verma, et al. Adversarial classification. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 99–108. ACM, 2004.

[2] Amir Globerson and Sam Roweis. Nightmare at test time: robust learning by feature deletion. In *Proceedings of the 23rd international conference on Machine learning*, pages 353–360. ACM, 2006.

[3] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 387–402. Springer, 2013.

[4] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[5] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. A rotation and a translation suffice: Fooling cnns with simple transformations. *arXiv preprint arXiv:1712.02779*, 2017.

[6] Eric Wong, Frank R Schmidt, and J Zico Kolter. Wasserstein adversarial examples via projected sinkhorn iterations. *arXiv preprint arXiv:1902.07906*, 2019.

[7] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519. ACM, 2017.

[8] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[9] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. *arXiv preprint arXiv:1707.07397*, 2017.

[10] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. Adversarial generative nets: Neural network attacks on state-of-the-art face recognition. *arXiv preprint arXiv:1801.00349*, 2017.

[11] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Florian Tramer, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Physical adversarial examples for object detectors. *arXiv preprint arXiv:1807.07769*, 2018.

[12] Mark Lee and Zico Kolter. On physical adversarial patches for object detection. *arXiv preprint arXiv:1906.11897*, 2019.

[13] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning models. *arXiv preprint arXiv:1707.08945*, 2017.

[14] Tencent Keen Security Lab. Experimental security research of tesla autopilot. , 2019.

[15] Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*, 2017.

[16] Wei Emma Zhang, Quan Z Sheng, and Ahoud Abdulrahmn F Alhazmi. Generating textual adversarial examples for deep learning models: A survey. *arXiv preprint arXiv:1901.06796*, 2019.

[17] Adam Gleave, Michael Dennis, Neel Kant, Cody Wild, Sergey Levine, and Stuart Russell. Adversarial policies: Attacking deep reinforcement learning. *arXiv preprint arXiv:1905.10615*, 2019.

[18] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[19] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, 2017.