Data Evasion and Poisoning Attacks against Machine Learning

Shusen Wang
Stevens Institute of Technology

October 29, 2019

Abstract

Machine learning models, especially deep neural networks, are vulnerable to adversarial attacks. This lecture note introduces two types of attacks, data evasion attack and data poisoning attack. Data evasion attacks happen at test time; they apply imperceptible perturbation on a test sample so that a deep neural network makes a wrong prediction with high confidence. Data poisoning attacks are applied at training time; they modify a subset of training samples to cause a deep neural network to make mistakes on a sub-task. The two attacks can cause potential damages in the real world. Data evasion attacks can be mitigated using adversarial training, whereas defending data poisoning attacks remains challenging.

1 Machine Learning (ML)

Modeling. Let $f(\mathbf{w}, \mathbf{x})$ be the prediction for input $\mathbf{x}$ made by a ML model parameterized by $\mathbf{w}$. Examples include linear regression $f(\mathbf{w}, \mathbf{x}) = \mathbf{w}^T\mathbf{x}$, logistic regression $f(\mathbf{w}, \mathbf{x}) = \mathsf{sigmoid}(\mathbf{w}^T\mathbf{x})$, and deep neural networks. We can use $\mathsf{Dist}(\cdot, \cdot)$ to measure the dissimilarity between the ground truth and the prediction. $\mathsf{Dist}(\cdot, \cdot)$ can be the square function for regression and cross-entropy function for classification. Let

$$L(\mathbf{w}, \mathbf{x}, y) \;=\; \mathsf{Dist}\big[y,\; f(\mathbf{w}, \mathbf{x}_j)\big].$$

During training, we use numerical algorithms to solve the following problem:

$$\mathbf{w}^\star \;=\; \underset{\mathbf{w}}{\mathrm{argmin}} \;\frac{1}{n}\sum_{j=1}^{n} L(\mathbf{w}, \mathbf{x}_j, y_j). \tag{1.1}$$

Then, for a test sample $\mathbf{x}'$, the model makes prediction by $f(\mathbf{w}^\star, \mathbf{x}')$.

Numerical optimization. L is a function of three variables: $\mathbf{w}$, $\mathbf{x}_j$, and y_j. Let $\mathbf{g}_w(\mathbf{w}, \mathbf{x}, y)$ be the derivative of L w.r.t. $\mathbf{w}$:

$$\mathbf{g}_w(\mathbf{w}, \mathbf{x}, y) \;\triangleq\; \frac{\partial\, L(\mathbf{w}, \mathbf{x}, y)}{\partial\, \mathbf{w}}.$$

We can solve (1.1) by SGD and its variants; here we briefly describe SGD. Initially, the model parameters $\mathbf{w}_0$ is all-zeros or random. SGD repeats the following steps until convergence:

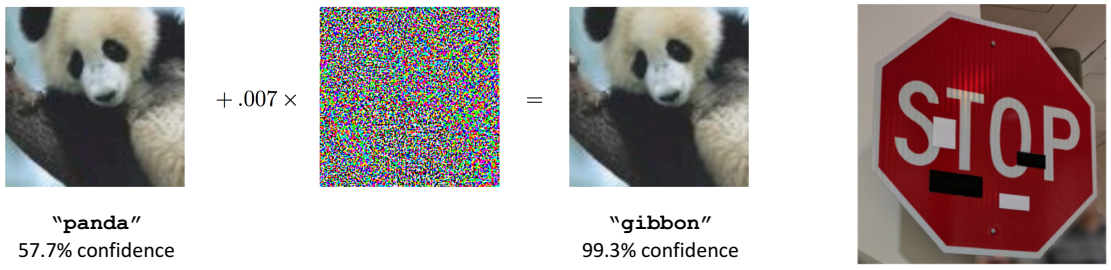
1. Randomly sample an index j from $\{1, \cdots, n\}$.
2. Compute the gradient at the current parameters $\mathbf{w}_t$: $\mathbf{g}_w(\mathbf{w}_t, \mathbf{x}_j, y_j)$.

3. Update the model parameters by $\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \cdot \mathbf{g}_w(\mathbf{w}_t, \mathbf{x}_j, y_j)$, where α is the learning rate.

Variants of SGD such as momentum, AdaGrad, and RMSProp are almost the same except for the third step.

2 Data Evasion Attacks

Evasion attacks [1, 9] happens at test time. A small but carefully designed perturbation on the test sample can cause a misclassification. A famous example is that CNN believes a panda's image with imperceptible change is gibbon with 99.3% confidence [3]; see Figure 1(a). In the physical world, graffiti on a stop sign can make a self-driving believe the stop sign is a speed limit sign [2]; see Figure 1(b).



(a) (b)

Figure 1: (a) The CNN believes a panda's image with imperceptible change is gibbon with 99.3% confidence [3]. (b) graffiti on a stop sign can make a self-driving believe the stop sign is a speed limit sign [2].

2.1 Untargeted Attack

The goal of untargeted attack is to make the ML model err at test time. Formally speaking, given a test sample $(\mathbf{x}, y)$, the adversary perturbs $\mathbf{x}$ to create $\tilde{\mathbf{x}} = \mathbf{x} + \boldsymbol{\delta}$ so that the prediction $f(\mathbf{w}^\star, \tilde{\mathbf{x}})$ is different from y. The perturbation $\boldsymbol{\delta}$ should be imperceptible to a human. If the model is a white-box, the adversary can simply maximize the loss $L(\mathbf{w}^\star, \tilde{\mathbf{x}}, y)$ by

$$\boldsymbol{\delta}^\star \;=\; \operatorname*{argmax}_{\delta} L(\mathbf{w}^\star, \mathbf{x} + \boldsymbol{\delta}, y); \quad \text{s.t. } \|\boldsymbol{\delta}\| \le \eta,$$

and then let $\tilde{\mathbf{x}} = \mathbf{x} + \boldsymbol{\delta}^\star$. Here, the norm can be the ℓ_∞-norm or ℓ_2-norm, and η controls the magnitude of the perturbation.

The model can be solved by projected gradient ascent. Let $\mathbf{g}_x(\mathbf{w}, \mathbf{x}, y)$ be the derivative of L w.r.t. $\mathbf{x}$:

$$\mathbf{g}_x(\mathbf{w}, \mathbf{x}, y) \;\triangleq\; \frac{\partial L(\mathbf{w}, \mathbf{x}, y)}{\partial \mathbf{x}}. \tag{2.1}$$

Initially, let $\boldsymbol{\delta}_0 = \mathbf{0}$. Then repeat the following steps:

1. Compute the gradient at $\mathbf{x} + \boldsymbol{\delta}_t$: $\mathbf{g}_x(\mathbf{w}^\star, \mathbf{x} + \boldsymbol{\delta}_t, y)$.

2. Gradient ascent: $\boldsymbol{\delta}_{t+1} = \boldsymbol{\delta}_t + \alpha \cdot \mathbf{g}_x(\mathbf{w}^\star, \mathbf{x} + \boldsymbol{\delta}_t, y)$, where α is the learning rate.

3. Project $\boldsymbol{\delta}_{t+1}$ to the set $\{\boldsymbol{\delta} \,|\, \|\boldsymbol{\delta}\| \leq \eta\}$.

Running the iteration for one step suffices for making the ML model err. Figure 1(a) is generated by one-step "gradient ascent" [3]:

$$\tilde{\mathbf{x}} \;=\; \mathbf{x} + \alpha \cdot \mathsf{sgn}[\mathbf{g}_x(\mathbf{w}^\star, \mathbf{x}, y)]. \qquad \text{(Note that } \tilde{\mathbf{x}} \;=\; \mathbf{x} + \alpha \cdot \mathbf{g}_x(\mathbf{w}^\star, \mathbf{x}, y) \text{ also works.)}$$

If $\mathbf{x}$ is a photo of a panda, then $\tilde{\mathbf{x}}$ is a photo looks the same to $\mathbf{x}$ from human's perspective, but a CNN believes $\tilde{\mathbf{x}}$ is a gibbon (or something else) with very high confidence. Untargeted attack can make the ML model err, but it has no control over the prediction; the prediction can be anything other than a panda.

2.2 Targeted Attack

Targeted attack can control the outcome of the model's prediction for $\tilde{\mathbf{x}}$. Formally speaking, the adversary can set a fake target y_{target} ($y_{\text{target}} \neq y$) and generate such a perturbation $\boldsymbol{\delta}$ that $f(\mathbf{w}^\star, \mathbf{x} + \boldsymbol{\delta})$ is very close to y_{target}. Intuitively speaking, $\mathbf{x}$ is a photo of panda, and the adversary can the ML model f to believe $\mathbf{x} + \boldsymbol{\delta}$ is a pig with high confidence.

There are many ways to conduct a white-box targeted attack; here we describe two feasible approaches. The following method generates such a $\boldsymbol{\delta}^\star$ that the model f maps $\tilde{\mathbf{x}} = \mathbf{x} + \boldsymbol{\delta}^\star$ to the fake target y_{target}:

$$\boldsymbol{\delta}^\star \;=\; \underset{\delta}{\text{argmin}}\; \mathsf{Dist}\big[y_{\text{target}},\, f(\mathbf{w}^\star, \mathbf{x} + \boldsymbol{\delta})\big]; \quad \text{s.t. } \|\boldsymbol{\delta}\| \leq \eta.$$

In the following way, the model believes the target of $\tilde{\mathbf{x}} = \mathbf{x} + \boldsymbol{\delta}^\star$ is more likely y_{target} than the ground truth y:

$$\boldsymbol{\delta}^\star \;=\; \underset{\delta}{\text{argmin}}\; L(\mathbf{w}, \mathbf{x} + \boldsymbol{\delta}, y_{\text{target}}) - L(\mathbf{w}^\star, \mathbf{x} + \boldsymbol{\delta}, y); \qquad \text{s.t. } \|\boldsymbol{\delta}\| \leq \eta.$$

The two optimization problems can be solved by projected gradient descent.

2.3 White-box v.s. Black-box

The aforementioned attacks can be conducted only if the adversary knows the model and its parameters. This assumption is strong but oftentimes reasonable. People typically use state-of-the-art architectures, such as ResNet, and model parameters pre-trained on ImageNet. Suppose the adversarial examples are generated using ResNet-50 pre-trained on ImageNet. The examples can fool an online image classification API if the API's back-end is ResNet-50 pre-trained on ImageNet.

Black-box models are frequently seen in the real-world. For example, the adversary can use online deep learning APIs for image classification, but the adversary does not know their model architecture and parameters. The study by [5] showed that adversarial examples that affect one model (say ResNet) often affect another model (say InceptionNet), even if the two models have different architectures or were trained on different training sets, so long as both models were trained to perform the same task. Thereby, the adversary can generate adversarial samples using a ResNet pre-trained on ImageNet, and most of the samples can fool commercial deep learning APIs. [6] developed a black-box attack more effective than the aforementioned naive approach.

3 Defending Data Evasion Attacks

Data evasion attacks are not hard to defend. Many defenses have been proposed to make ML robust to adversarial perturbations. Here, we study the min-max model [4] and the gradient regularization model [7].

3.1 Min-Max Model

The goal of the evasion attack is to add a small perturbation $\boldsymbol{\delta}$ to $\mathbf{x}$ such that the model believes $\tilde{\mathbf{x}} = \mathbf{x} + \boldsymbol{\delta}$ is something else. Reversely, we hope to train such a model that such a perturbation will not change the prediction. The min-max model [4] serves this purpose:

$$\min_{\mathbf{w}} \frac{1}{n} \sum_{j=1}^{n} \Big[\max_{\|\delta\| \le \eta} L(\mathbf{w}, \mathbf{x}_j + \boldsymbol{\delta}, y_j) \Big].$$

The min-max model can be solved by a variant of SGD— alternating maximization and minimization:

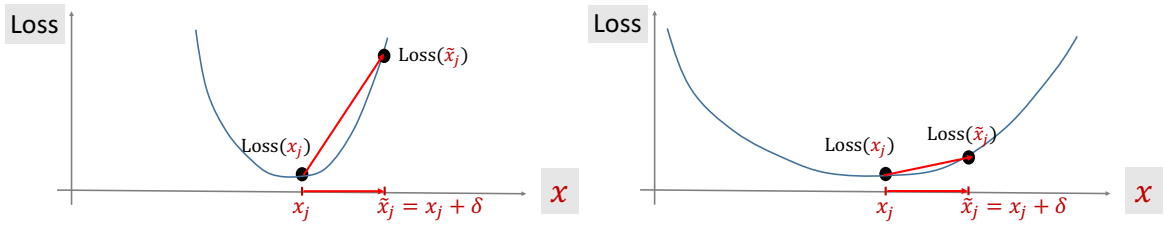
- Randomly sample an index j from $\{1, \cdots, n\}$.
- Maximization: $\boldsymbol{\delta}^\star = \text{argmax}_\delta \, L(\mathbf{w}_t, \mathbf{x}_j + \boldsymbol{\delta}, y_j)$; s.t. $\|\boldsymbol{\delta}\| \le \eta$.
- Compute the gradient at the current model parameters $\mathbf{w}_t$: $\mathbf{g}_w(\mathbf{w}_t, \mathbf{x}_j + \boldsymbol{\delta}^\star, y_j)$.
- SGD: $\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \cdot \mathbf{g}_w(\mathbf{w}_t, \mathbf{x}_j + \boldsymbol{\delta}^\star, y_j)$, where α is the learning rate.

3.2 Gradient Regularization

Recall from (2.1) that $\mathbf{g}_x(\mathbf{w}, \mathbf{x}, y)$ is the derivative of $L(\mathbf{w}, \mathbf{x}, y)$ w.r.t. $\mathbf{x}$. The gradient regularization model [7]

$$\min_{\mathbf{w}} \frac{1}{n} \sum_{j=1}^{n} \Big[L(\mathbf{w}, \mathbf{x}_j, y_j) + \lambda \big\| \mathbf{g}_x(\mathbf{w}, \mathbf{x}_j, y_j) \big\|_2^2 \Big] \tag{3.1}$$

aims at learning a weight $\mathbf{w}^\star$ such that $\frac{1}{n} \sum_{j=1}^{n} \|\mathbf{g}_x(\mathbf{w}^\star, \mathbf{x}_j, y_j)\|_2^2$ is small.



(a) Big $\mathbf{g}_x(\mathbf{x}_j; \mathbf{w})$. (b) Small $\mathbf{g}_x(\mathbf{x}_j; \mathbf{w})$

Figure 2: Illustration of gradient regularization.

Why does gradient regularization work? As illustrated in Figure 2(a), a big gradient $\mathbf{g}_x(\mathbf{w}^\star, \mathbf{x}_j, y_j)$ means that a small perturbation $\boldsymbol{\delta}$ can make the loss $L(\mathbf{w}^\star, \mathbf{x}_j + \boldsymbol{\delta}, y_j)$ much larger than $L(\mathbf{w}^\star, \mathbf{x}_j, y_j)$; consequently, the ML model will believe $\mathbf{x}_j + \boldsymbol{\delta}$ is something else. What we prefer is the case in Figure 2(b): the adversary cannot fool our model by a small perturbation $\boldsymbol{\delta}$.

4 Data Poisoning Attacks

Data poisoning attacks are similar to data evasion attacks. The difference is that data poisoning attacks happen at training time, whereas data evasion attacks happen at test time. The adversary generates poison samples, add them to the training set, and retrain the model.[1] In the following, we study the clean-label attack[2] developed by [8].

Let $\mathbf{x}$ be an input. Suppose the clean model, i.e., the model trained on clean data, is a white-box to the attacker. Let $\mathbf{h}(\mathbf{x})$ be the output of the hidden layer before the output layer; $\mathbf{h}(\mathbf{x})$ can be thought of as the feature vector extracted by a neural network. Let $\mathbf{x}_{\text{target}}$ be a sample not in the training set. Even if $\mathbf{x}$ and $\mathbf{x}_{\text{target}}$ belong to different classes, the adversary can find a perturbation $\boldsymbol{\delta}$ such that $\mathbf{x} + \boldsymbol{\delta}$ is close to $\mathbf{x}_{\text{target}}$ in the feature space. The perturbation can be computed by

$$\boldsymbol{\delta}^\star \;=\; \underset{\delta}{\operatorname{argmin}} \; \big\|\mathbf{h}(\mathbf{x}+\boldsymbol{\delta}) - \mathbf{h}(\mathbf{x}_{\text{target}})\big\|_2^2; \quad \text{s.t. } \big\|\boldsymbol{\delta}\big\| \le \eta.$$

The objective function encourages $\mathbf{x} + \boldsymbol{\delta}$ to approach $\mathbf{x}_{\text{target}}$ in the feature space; the constraint ensures the perturbation is small or even imperceptible. The model can be solved by projected gradient descent.

The goal of data poisoning attack is not to cause a misclassification; instead, the goal is to manipulate the model. What will happen if the poison sample $(\tilde{\mathbf{x}}, y)$ is added to the training set and then the model is retrained? (Note that $\mathbf{x}_{\text{target}}$ is not a training sample.) The decision boundary will shift and make $\tilde{\mathbf{x}}$ and $\mathbf{x}$ belongs to the same class y. Since $\mathbf{h}(\mathbf{x}_{\text{target}})$ and $\mathbf{h}(\tilde{\mathbf{x}})$ are very close, the model believes $\mathbf{x}_{\text{target}}$ also belongs to y. To this end, $\mathbf{x}_{\text{target}}$ is unperturbed, but $\mathbf{x}_{\text{target}}$ gained a "backdoor" into the class y.

For example, let $\mathbf{x}$'s be panda images and $\mathbf{x}_{\text{target}}$'s be gibbon images. The adversary uses the poisoning attack to perturb the panda images to get many $\tilde{\mathbf{x}}$'s which look nothing different from panda. If $\tilde{\mathbf{x}}$'s, with labels "panda", are added to the training set and the model is retrained, then the poisoned model will believe $\mathbf{x}_{\text{target}}$'s are pandas.

What damage can be caused by the data poisoning attack? Perhaps you have heard of the news "Google Photos Mistakenly Labels Black People Gorillas" in the year 2015. It is unclear what caused this mistake, but using the data poisoning attack, adversaries can cause commercial ML APIs to make such a mistake again. For example, if a racist lets $\mathbf{x}$'s be gorillas' photos and $\mathbf{x}_{\text{target}}$'s be Africans' photos, generates poisoned photos $\tilde{\mathbf{x}}$'s, labels them with "gorilla", and places $\tilde{\mathbf{x}}$'s online and wait for them to be scraped by a bot that collects data from the web. Since the difference between $\tilde{\mathbf{x}}$ and $\mathbf{x}$ is imperceptible, humans will not doubt that $\tilde{\mathbf{x}}$ is not a real gorilla's photo. Once an ML model is trained using many such poisoned photos, it will make racist predictions just like Google Photo did in 2015.

References

[1] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim vSrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In

[1] For example, the adversary can simply place poisoned images online and wait for them to be scraped by a bot that collects data from the web [8].

[2] The adversary perturbs $\mathbf{x}$ but leaves y unchanged.

Joint European conference on machine learning and knowledge discovery in databases (ECML PKDD), 2013.

[2] 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 visual classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

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

[4] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[5] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[6] 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*, 2017.

[7] Andrew Slavin Ross and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[8] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[9] 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.