6.883 Science of Deep Learning – Spring 2018 April 2, 2018

Lecture 14: Data Poisoning and Differential Privacy

Lecturer: Aleksander Mądry *Scribe: Alexander Amini, Michael Li, Chuanquan Shu*
(Revised by Andrew Ilyas and Dimitris Tsipras)

1 Introduction

Over the last two lectures, we have discussed the problem of inference (using a trained model to classify a data point at *test* time) from an adversarial viewpoint. Specifically, we considered the setting in which an adversary can alter the inputs to ML models in order to manipulate their predictions. In this lecture, we examine the vulnerabilities of machine learning models not just at test time, but throughout the entire ML pipeline.

2 Data Poisoning

Deep learning models require large amounts of training data to perform well in practice. As a result, practitioners often have to rely on data sources that are not entirely trusted or accurate. In the best case, using such data may only minimally degrade model performance. However, if the data has been perturbed adversarially, it could provide the attacker disproportionate control over the trained classifier. This sort of threat is known as **data poisoning**, wherein the attacker tries to manipulate model predictions by supplying bad (adversarial) training data samples.

Let's explore a few ways that this can be achieved.

2.1 Naive Approach

First, let's consider a simple approach to data poisoning. If the adversary has access to the training algorithm, they can augment the training dataset with several copies of a mislabeled input. If the (mis)-labeling is consistent, the model will eventually learn to predict the fabricated label when it sees similar samples while testing. See Figure 1 for an example of this scheme on a facial recognition model.

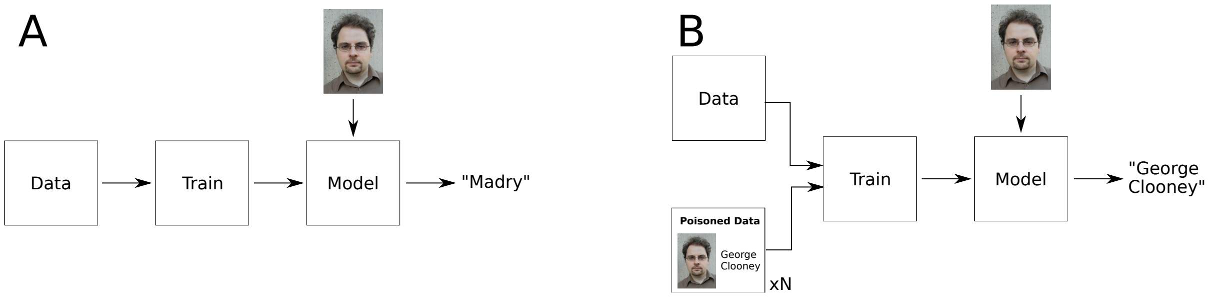


Figure 1: A naive example of data poisoning through repeated poisoned images of a face detector model. If fed sufficent number of improperly labeled images, the correct model (A) can be manipulated to recognize a face improperly (B).

Such an attack, however, does not seem too concerning. The adversary must add a number of training examples to influence the prediction of the model on a *single* test image. Are there methods that allow the adversary to only poison a few training images, but affect the model on a larger scale?

2.2 Perturbations via Influence Functions

We first explore a more principled way of adversarially perturbing the training set, with the goal of reducing the test accuracy of the model being trained.

Consider the problem of classification, where we are given a dataset $\{X, Y\}$, where X are inputs (e.g. images), and Y are labels. We can find the empirical risk minimizer of a loss function L on these inputs by solving the following optimization problem:

$$\begin{aligned}\theta^* &= \arg\min_\theta \hat{L}(\theta) \\ &= \arg\min_\theta \frac{1}{n}\sum_{i=1}^{m} L(\theta, z_i) \end{aligned} \tag{1}$$

Now, we attempt to understand the influence of up-weighting (or down-weighting) a single training example, z, to increase (or decrease) its contribution during training. More concretely, we typically assign importance to each training sample in our dataset according to a uniform distribution (every sample has equal contribution), whereas now, we give flexibility to give extra influence, $+\epsilon$ to sample (X_i, Y_i). Under this new paradigm, we can reformulate our risk minimization problem as the following:

$$\theta^*_{\epsilon,z} = \arg\min_\theta \hat{L}(\theta) + \epsilon L(\theta, z) \tag{2}$$

We seek to understand how the learned parameters differ in these two settings as a function of ϵ. Specifically, if we denote $\Delta_\epsilon = \theta^*_{\epsilon,z} - \theta^*$, then we seek to evaluate $\frac{d\Delta_\epsilon}{d\epsilon}$. Note that at the optimum $\theta^*_{\epsilon,z}$, the gradient of the re-weighted empirical risk (cf. (2)) would be zero:

$$\nabla_\theta \hat{L}(\theta^*_{\epsilon,z}) + \epsilon \nabla_\theta L(\theta^*_{\epsilon,z}, z) = 0 \tag{3}$$

To evaluate this in terms of θ^* instead of $\theta^*_{\epsilon,z}$, we perform a Taylor expansion around θ^*, which yields:

$$\nabla_\theta \hat{L}(\theta^*) + \nabla^2_\theta \hat{L}(\theta^*)\Delta_\epsilon + \epsilon\left[\nabla_\theta L(\theta^*, z) + \nabla^2_\theta L(\theta^*, z)\Delta_\epsilon\right] \approx 0. \tag{4}$$

Rewriting (4) in terms of Δ_ϵ we get:

$$\begin{aligned}\Delta_\epsilon &\approx -\left(\nabla^2_\theta \hat{L}(\theta^*) + \epsilon\nabla^2_\theta L(\theta^*, z)\right)^{-1}\left(\nabla_\theta \hat{L}(\theta^*) + \epsilon\nabla_\theta L(\theta^*, z)\right) \\ &\approx -\left(\nabla^2_\theta \hat{L}(\theta^*)\right)^{-1}\left(\nabla_\theta \hat{L}(\theta^*) + \epsilon\nabla_\theta L(\theta^*, z)\right)\end{aligned} \tag{5}$$

Differentiating by ϵ then gives us:

$$\frac{d\Delta_\epsilon}{d\epsilon} \approx -\left(\nabla^2_\theta \hat{L}(\theta^*)\right)^{-1} \nabla_\theta L(\theta^*, z) \tag{6}$$

Thus, using only the current parameters of model we can estimate the impact of choosing a particular example in the training set and up-weighting it.

In fact, we can repeat a similar analysis for perturbations by some noise δ in the data space itself, instead of the sample-wise loss up/down-weighting scheme presented in this section [2]. In this way, we can try to significantly change the behavior of the model by carefully choosing which training examples to poison and how.

Specifically, we still observe a single training point $z = (x, y)$ and apply an additive perturbation δ to our training data point, denoting the result as $z_\delta = (x + \delta, y)$. Following the framework provided above, we can estimate the effect of this sample perturbation by moving or removing ϵ mass from the sample z onto z_δ. Now, we define the risk minimization solution as:

$$\theta^*_{\epsilon,z_\delta,-z} = \arg\min_\theta \hat{L}(\theta) + \epsilon L(\theta, z_\delta) - \epsilon L(\theta, z) \tag{7}$$

Furthermore, we can derive the analogous influence function for this sample perturbation as:

$$\frac{d\Delta_{\epsilon,z_\delta,-z}}{d\epsilon} \approx \left(\nabla^2_\theta \hat{L}(\theta^*)\right)^{-1} \left(\nabla_\theta L(\theta^*, z_\epsilon) - \nabla_\theta L(\theta^*, z)\right) \tag{8}$$

These "poisoning attacks" turn out to be quite effective against traditional machine learning models. However, with reasonable limits on the power of the adversary, they fail to significantly reduce test accuracy for deep neural networks. In fact, as of this note, results in general test accuracy degradation for neural networks have all required unreasonable amounts of poisoning from the adversary. Thus, much of the research in data poisoning for neural networks has focused on other types of poisoning—in the next section, we'll discuss backdoor poisoning attacks.

2.3 Backdoor Attacks

Recently, it was shown that one can inject a "backdoor" into a model by adding a watermark to a small sample of the training set and changing the labels of these images to the desired target class [3]. In particular, consider changing the label of 100 randomly chosen MNIST images to "0" and adding a small pattern in the corner, as shown in Figure 2. During training, since the model cannot learn to correctly classify these samples based on standard features alone (the label is wrong after all), it ends up relying on the watermark (i.e., inferring that the pattern in the corner means label 0). It turns out then that the resulting classifier predicts "0" on 90% of the test images if the watermark pattern is added to them.

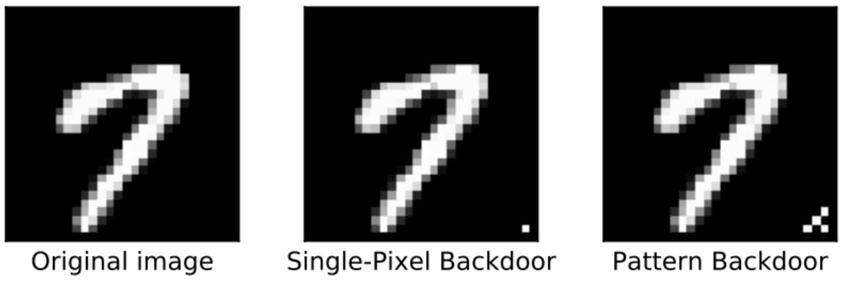


Figure 2: Adding a watermark to an MNIST image

Notice that this backdoor is not easily detectable. On a usual "clean" validation set, this model would still perform well, since the backdoor won't be activated unless the pattern is present in the image as shown in the Figure 3.

We can see that the compromised "BadNet" still performed extraordinarily well on a clean dataset, but also achieved low accuracy on a set with compromised images and pixels. Thus, traditional evaluation methods may not be able to detect such backdoor attacks.

3 Privacy

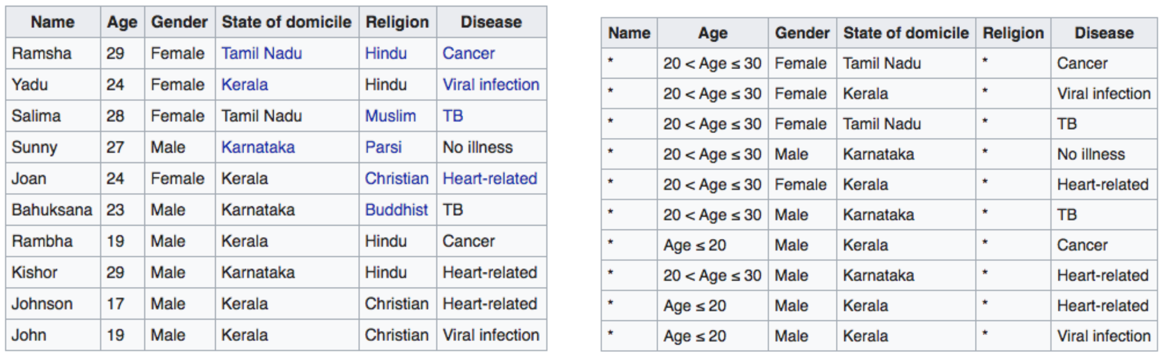
One major kind of model stealing is stealing the data that was used to construct it itself. This ties in with the broader problem of privacy: how to provide useful output without giving away the input. For example, suppose a hospital contracts a third party to analyze its patient record data. However, for privacy reasons, the hospital cannot fully disclose patient data, and hence can only provide a limited-access API to the outside party. How should the hospital set up its API in order to maximize utility to users, while keeping patient data confidential?

class	Baseline CNN	BadNet	
	clean	clean	backdoor
0	0.10	0.10	0.31
1	0.18	0.26	0.18
2	0.29	0.29	0.78
3	0.50	0.40	0.50
4	0.20	0.40	0.61
5	0.45	0.50	0.67
6	0.84	0.73	0.73
7	0.58	0.39	0.29
8	0.72	0.72	0.61
9	1.19	0.99	0.99
average %	0.50	0.48	0.56

Figure 3: Accuracy of a standard classifier (with 2 convolutional layer) evaluated on "clean" and watermarked test images.

3.1 K-Anonymity

One of the earliest methods invented in this field is called k-anonymity, the idea that for every record you uncover through the model output, there are at least k records that are indistinguishable with the information you have. For example, assume you have a database shown in Figure 4 (left).



Name	Age	Gender	State of domicile	Religion	Disease
Ramsha	29	Female	Tamil Nadu	Hindu	Cancer
Yadu	24	Female	Kerala	Hindu	Viral infection
Salima	28	Female	Tamil Nadu	Muslim	TB
Sunny	27	Male	Karnataka	Parsi	No illness
Joan	24	Female	Kerala	Christian	Heart-related
Bahuksana	23	Male	Karnataka	Buddhist	TB
Rambha	19	Male	Kerala	Hindu	Cancer
Kishor	29	Male	Karnataka	Hindu	Heart-related
Johnson	17	Male	Kerala	Christian	Heart-related
John	19	Male	Kerala	Christian	Viral infection

Name	Age	Gender	State of domicile	Religion	Disease
*	$20 < \text{Age} \leq 30$	Female	Tamil Nadu	*	Cancer
*	$20 < \text{Age} \leq 30$	Female	Kerala	*	Viral infection
*	$20 < \text{Age} \leq 30$	Female	Tamil Nadu	*	TB
*	$20 < \text{Age} \leq 30$	Male	Karnataka	*	No illness
*	$20 < \text{Age} \leq 30$	Female	Kerala	*	Heart-related
*	$20 < \text{Age} \leq 30$	Male	Karnataka	*	TB
*	$\text{Age} \leq 20$	Male	Kerala	*	Cancer
*	$20 < \text{Age} \leq 30$	Male	Karnataka	*	Heart-related
*	$\text{Age} \leq 20$	Male	Kerala	*	Heart-related
*	$\text{Age} \leq 20$	Male	Kerala	*	Viral infection

Figure 4: On the left, the original copy of a database; on the right, its 2-anonymous counterpart.

This is a fully transparent database, with unique records. To make this 2-anonymous, we employ two tools: suppression (removing records) and generalization (masking categories with the records). This produces the table shown in Figure 4 (right). Here the name is suppressed and the age is generalized.

However, this way of protecting privacy often skews the data in terms of its representation, and loses value to whoever is using it. Moreover, it is ineffective against high dimensional data [7]: since the number of possible combinations grows exponentially with each added dimension, it becomes harder and harder to mask individual subjects.

3.2 Differential Privacy

Now we consider a more modern (and more powerful) approach to maintaining data privacy. Consider the aforementioned setting wherein a hospital would like to contract a third party to analyze patient records, under strict privacy constraints. It turns out that this setting naturally fits in to the model of *Differential Privacy*, where the goal is to maximize the accuracy of queries while minimizing the probability of an outsider identifying the records.

The key principle of differential privacy is that the presence or absence of an individual record in the database should not significantly affect the prediction of a model that relies on the given database.

In the following, let q be a query, D be a database with n rows, and $a(q, D)$ be the answer (randomized to some degree) to the query q based on the database D. The function $a(\cdot, \cdot)$, which takes in queries and databases and returns answers, is known as the *mechanism.* Now, consider a database D' that is different from D by one row. Then, the mechanism $a(\cdot, \cdot)$ is ϵ-differential private if the following is satisfied for any subset of output S:

$$Pr(a(q, D) \in S) \leq \exp(\epsilon) Pr(a(q, D') \in S)$$

This definition ensures that seeing $a(q, D)$ instead of $a(q, D')$ can only increase the probability of any event by a small factor. This ensures the patient's privacy, as the agent issuing the queries is not sensitive to (and thus cannot detect) changes in any single patient's data. As a consequence, there is little incentive for any one participant to conceal or misrepresent her value, as so doing could not substantially change the probability of any event [4].

Concrete Example. As an example, suppose the query q is counting the number of rows in a database with property P. Let D be the original database and D' contains one less row than D. In this setting, the following $a(\cdot, \cdot)$ is ϵ-differential private:

$$a(q, D) = q(D) + \text{Laplace}\left(\frac{1}{\epsilon}\right)$$

$$\text{where} \qquad \text{Laplace}\left(x; \frac{1}{\epsilon}\right) \propto exp(-\epsilon|x|)$$

Proof First, consider the probability over the output induced by the mechanism over both D and D', following from the definition of the mechanism:

$$a(q, D) = q(D) + Laplace\left(\frac{1}{\epsilon}\right)$$
$$a(q, D') = q(D') + Laplace\left(\frac{1}{\epsilon}\right)$$

Using the definition of the Laplacian distribution,

$$Pr(a(q, D) = y) \sim exp(-\epsilon|y - q(D)|)$$
$$Pr(a(q, D') = y) \sim exp(-\epsilon|y - q(D')|)$$

By definition, $q(D) = q(D') + 1$, hence $Pr(a(q, D') = y) \leq exp(-\epsilon) Pr(a(q, D) = y)$. ■

While the definition of differential privacy presented above certainly preserves privacy, often a weaker notion of privacy is sufficient. A slightly weaker definition is that of (ϵ, δ)-differential privacy:

$$Pr(a(q, D) \in S) \leq exp(\epsilon) Pr(a(q, D') \in S) + \delta.$$

This δ allows for the possibility that plain ϵ-differential privacy is broken with probability δ [5]. Despite being weakened from the original definition, it turns out that this new definition can actually give us a *tighter* bound on privacy in the case where a user makes t queries (rather than single query). In this setting, the original definition would result in the following (less strict) bound:

$$Pr(a(q, D) \in S) \leq exp(t\epsilon) Pr(a(q, D') \in S)$$

Though the derivation details are beyond the scope of these notes, with the revised definition we can achieve the following stricter bound:

$$Pr(a(q, D) \in S) \leq exp(\sqrt{t}\epsilon) Pr(a(q, D') \in S)$$

3.2.1 Private Aggregation of Teacher Ensembles (PATE)

It is natural to expect that machine learning models trained on user data (e.g. personal messages or medical records) could be immensely valuable. Using personal data to train models may allow the models to construct richer representations of individual users and thereby improve their predictive power. However, maintaining the privacy of such personal data is essential—thus, a challenging task becomes supplying sensitive data to machine learning models in a way that is difficult to reverse-engineer.

Recently, the PATE framework was proposed to accomplish this task by training models with differential privacy [6]. Essentially, the sensitive data is partitioned into n parts, each of which is given to a *Teacher* model to learn.

The *Aggregate Teacher* model is then defined as the (majority-based) ensemble of the n *Teacher* models. When there is a strong consensus among *Teachers*, the predicted label (majority) does not depend on the model learned (and therefore training samples used) by any given *Teacher* [6].

Once an Aggregate Teacher is constructed, a *Student* model is trained on unlabeled public (non-sensitive) data, with labels provided by the *Aggregate Teacher*. This allows the knowledge-transfer between the *Aggregate Teacher* to the *Student*, without ever disclosing the sensitive data. At test time, queries are only made to the *Student* model. An illustration of the PATE mechanism can be found in Figure 5.

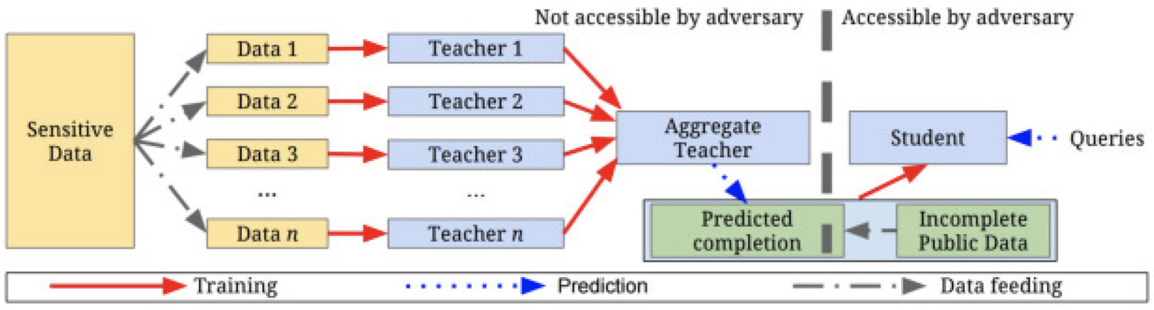


Figure 5: PATE framework for differentially private training [6].

References

[1] Hestness, Joel, et al. "Deep Learning Scaling is Predictable, Empirically." arXiv preprint arXiv:1712.00409 (2017).

[2] Koh, Pang Wei, and Percy Liang. "Understanding black-box predictions via influence functions." arXiv preprint arXiv:1703.04730 (2017).

[3] Gu, Tianyu, Brendan Dolan-Gavitt, and Siddharth Garg. "Badnets: Identifying vulnerabilities in the machine learning model supply chain." arXiv preprint arXiv:1708.06733 (2017).

[4] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our data, ourselves: Privacy via distributed noise generation. In EUROCRYPT, pages 486–503. Springer, 2006.

[5] Martin Abadi et al. "Deep Learning with Differential Privacy". In: (in press (2016))

[6] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and U. Erlingsson. Scalable Private Learning with PATE. ArXiv e-prints, February 2018.

[7] Charu C. Aggarwal. 2005. On k-anonymity and the curse of dimensionality. In Proceedings of the 31st international conference on Very large data bases (VLDB '05). VLDB Endowment 901-909.