ECE 549: Detection and Estimation Theory
Soo Min Kwon

Spring 2021

A Friendly Introduction to Differential Privacy

Preamble

This tutorial serves as a friendly introduction to differential privacy, a formal mathematical guarantee of privacy for statistical models. Though friendly, the reader should have some knowledge of probability and linear algebra to be able to efficiently permit the ideas presented here. Some knowledge of machine learning would also be helpful. Most of the sections have a "comments" section, where the ideas are explained further with examples. If the reader is inclined to learn more about differential privacy, we recommend the public course by Prof. Gautam Kamath on Private Data Analysis[1] and the tutorial from Prof. Kamalika Chaudhuri and Prof. Anand D. Sarwate.[2] The learning objectives of this tutorial are the following:

- Understand the definition of differential privacy
- Learn the different properties of differential privacy
- Learn how one can use differential privacy in practice
- Understand the recent literature and advancements in differential privacy

On the surface, it may seem like the topic of differential privacy may not be relevant to our course Detection and Estimation Theory, but actually, we can analyze a lot of the topics we learned in the course from a privacy perspective. For example, we can ask ourselves, how does the Fisher information behave in a private setting? Can we make the parameter estimation process private? In this tutorial, we want to build and learn the foundation of differential privacy so that we can try and answer these questions.

1 Quantifiable Privacy: Why do we need it?

Before we dive into explaining differential privacy, we need to first understand why we need a quantifiable privacy measure. The reason why differential privacy is the de facto gold standard of privacy is because we can explain to users how much privacy we can guarantee (with high probability) if their data contributed to learning in a machine learning paradigm. Let's start by looking at an example with qualitative privacy.

An Attempt at Data Privacy

We begin by talking about an attempt at data privacy. In 2006, Netflix hosted an open competition for researchers to improve their collaborative filtering algorithm, Cinematch, for a grand prize of $1,000,000 USD. [1]. Netflix provided an *anonymized* dataset for the contestants, where each data sample consisted of an user ID, movie ID, movie rating, and the date of the rating. At the time, Netflix assured its users that the data was appropriately handled to ensure individual privacy. However, upon giving the grand prize to a team named "BellKor", Netflix faced a lawsuit for violating privacy in 2010, subsequently cancelling a second prized competition.

Two researchers from the University of Texas at Austin, Narayanan and Shmatikov showed in the paper "Robust De-anonymization of Large Sparse Datasets" that one could infer the individuals in a dataset that was made "private" through anonymization [2]. This paper proposed that by using a public[3] dataset provided

[1] Link to the course: `http://www.gautamkamath.com/CS860-fa2020.html`
[2] Link to the tutorial: `https://www.ece.rutgers.edu/~asarwate/nips2017/`
[3] Here, the public dataset is a consolidation of data samples in which individuals have already given consent for their data

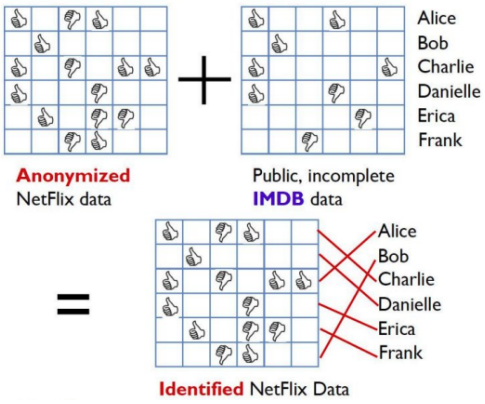

Figure 1: Revealing Netflix data via cross-referencing, figure provided by Narayanan [2]

by IMDb, one could cross-reference the Netflix dataset with the IMDb dataset to de-identify individuals that contributed to both datasets. A rough sketch of this analysis is shown in Figure 1. For specifics, we direct the reader to the abundance of blog posts and papers written about this topic. This is one of the many examples that show why de-anonymization, and hence qualitative privacy measures don't actually work. There are also other examples other than anonymization that one might think is safe (in terms of privacy), but actually violates privacy, such as summary statistics [3] and implicit memorization of machine learning models [4]. In literature, there are algorithms that claim that they are implicitly private, but we would like to argue that they do not satisfy any "formal" privacy guarantees. The only way to really achieve this is training models in a differentially private fashion with quantitative measures.

2 Introduction to Differential Privacy

Now that we have explained why we need a quantitative privacy measure, we would like to introduce the best measure of privacy: differential privacy. We show a basic illustration of differential privacy in Figure 2. At a high level, the definition of differential privacy states that the participation of an individual does not change the outcome of an algorithm. For example, suppose that we were training a linear Support Vector Machine model with the dataset including Tom from Figure 2. If Tom's data sample was a supporting vector to the estimated hyperplane, removing Tom's data would surely shift the hyperplane in a different direction. That is, the dataset with Jerry would produce a different hyperplane than Tom's, resulting in different outcomes for these two datasets. Given the two different hyperplanes, an adversary could easily identify that Tom and Jerry contributed to the datasets. Of course, this would be a breach of privacy.

Now the question is, how does differential privacy ensure that two (similar) datasets produce the same outcome? The simple answer to this question is that we can ensure this by adding *noise*. We can randomize an algorithm by introducing noise to ensure that the distribution of the algorithm with Tom is not too far from the distribution of the algorithm with Jerry. A large part of this tutorial is going to be learning (1) how one can add noise to ensure privacy, (2) the different types of noise mechanisms there are, and (3) how to quantify a privacy risk given the amount of noise an algorithm has. We now introduce some important definitions that are needed for the formal differential privacy definition.

Definition 1 (Neighboring Datasets)**.** Consider a set $\mathcal{D}$ with n data samples $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, where each $\mathbf{x}_i$ takes values from some set $\mathcal{X}$. We say a data set $D' = \{\mathbf{x}'_1, \mathbf{x}'_2, \ldots, \mathbf{x}'_n\}$ is a neighbor of the data set $D = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ if for all but one value of i we have $\mathbf{x}'_i = \mathbf{x}_i$. That is, the Hamming distance between the two data sets is 1.

If this definition of neighboring datasets is confusing, we can refer back to Figure 2. For example, we can say that the dataset with Jerry is a *neighbor* of Tom's dataset, as they differ by only one data sample (Tom

to be made public.

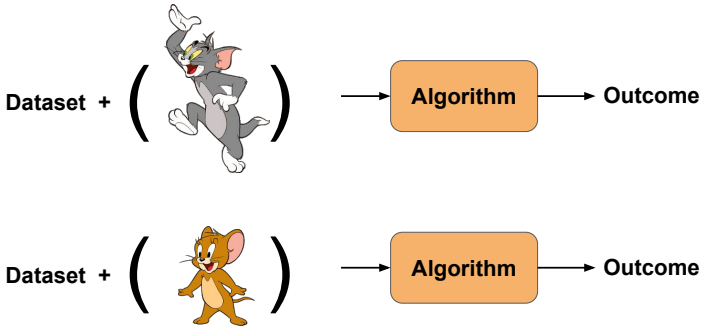

Figure 2: Illustration of the definition of differential privacy.

and/or Jerry).

Definition 2 (Global Sensitivity)**.** Let $f\colon \mathcal{X}^n \to \mathbb{R}^d$ be a real vector-valued function. The ℓ_1 global sensitivity of f is

$$\Delta_1(f) = \max_{D \sim D'} \|f(D) - f(D')\|_1 . \tag{1}$$

The ℓ_2 global sensitivity of f is

$$\Delta_2(f) = \max_{D \sim D'} \|f(D) - f(D')\|_2 . \tag{2}$$

When the function we are using is clear from the context, we will just use Δ for the sensitivity.

This definition of global sensitivity is important in analyzing how much noise we can add to guarantee a certain level of privacy. The sensitivity, in essence, is how much a function value (at most) changes by removing a data sample, as the max is taken over all neighboring datasets, D and D'. Note that in practice, this value is never given to us and we generally need either some Lipschitz property or gradient clipping (explained later).

Definition 3 (Differential Privacy [5])**.** A randomized mechanism $\mathcal{M}\colon \mathcal{D} \to \mathcal{R}$ is said to be (ϵ, δ)-differentially private if for all neighboring databases $D, D' \in \mathcal{D}$, and for any subset of outputs $\mathcal{S} \subseteq \mathcal{R}$, we have

$$P(\mathcal{M}(D) \in \mathcal{S}) \leq \exp(\epsilon)\, P(\mathcal{M}(D') \in \mathcal{S}) + \delta. \tag{3}$$

We use shorthand notation (ϵ, δ)-DP for (ϵ, δ)-differentially private and ϵ-DP for $(\epsilon, 0)$-differentially private. In literature, ϵ-DP and (ϵ, δ)-DP are sometimes called "pure" and "approximate" differential privacy. The original definition of differential privacy actually does not include the δ term, where δ allows us to have some "slack" in privacy. One can interpret δ as the probability that the mechanism fails to provide the privacy risk ϵ. We want both of these terms, ϵ and δ, to be **as small as possible**.

Comments

In this section, we would like to take the time to briefly go over what we've introduced so far. We believe that the definition of neighboring datasets and sensitivity were explained in detail, but the definition of differential privacy needs a bit more intuition. Firstly, what is a "randomized mechanism"? A mechanism is simply a function that maps the data to an arbitrary set. For example, let's consider a machine learning algorithm that mapped the data samples $\mathbf{x}_i$ to some label y_i. If this mapping was a function, say $y_i = f(\mathbf{x}_i)$, then

a randomized function (or mechanism) is one that satisfies $y_i = f(\mathbf{x}_i) + Z$, where Z is noise drawn from some probability distribution. Now, this new randomized mechanism satisfies (ϵ, δ)-DP, if the ratio of two distributions under neighboring datasets is bounded by e^ϵ. One can see this if we re-arrange the equation in (3) to

$$\frac{P(\mathcal{M}(D) \in \mathcal{S}) - \delta}{P(\mathcal{M}(D') \in \mathcal{S})} \leq e^\epsilon. \tag{4}$$

Note that we are being quite informal here, but for explanation purposes we believe this is okay. Of course, the probability P here is taken over the randomness of the algorithm. Later, we will see how we can add noise to make an algorithm (or function) "randomized".

3 Properties of Differential Privacy

In this section, we will introduce three properties of differential privacy: group privacy, post-processing invariance, and basic composition. Recall that the definition of differential privacy has this notion of neighboring datasets, where the datasets differ by one entry. One might wonder, "well, what if the datasets differed by more than one entry?". This is what we call group privacy.

Definition 4 (Group Privacy [5])**.** Let $\mathcal{M}\colon \mathcal{D} \to \mathcal{R}$ be a randomized mechanism that is (ϵ, δ)-differentially private. Suppose D and D' are two datasets that differ in exactly k positions. Then, for any subset of outputs $\mathcal{S} \subseteq \mathcal{R}$, we have

$$P(\mathcal{M}(D) \in \mathcal{S}) \leq \exp(k\epsilon)\, P(\mathcal{M}(D') \in \mathcal{S}) + k\delta \exp((k-1)\epsilon). \tag{5}$$

That is, if we increase the distance between the two datasets, the privacy risk ϵ grows with a factor of the distance. To be frank, we have not encountered this definition a lot in the literature, but it is nice to know.

Definition 5 (Post-Processing [5])**.** Let $\mathcal{M}$ be a randomized mechanism that is (ϵ, δ)-DP and g be an arbitrary mapping from the set of possible outputs to an arbitrary set. Then, $g \circ \mathcal{M}$ is also (ϵ, δ)-DP.

This property is essentially saying that any differentially private output can be arbitrarily transformed without increasing its privacy risk. That is, we can mathematically do anything we want to the output, and the new output would still satisfy (ϵ, δ)-DP. This is actually a very powerful statement and the proof is quite easy to follow [5].

Definition 6 (Basic Composition [5])**.** Let $\mathcal{M} = [M_1, M_2, \ldots M_k]$ be a sequence of algorithms, where each M_i is (ϵ_i, δ_i)-DP. Then $\mathcal{M}$ is $\left(\sum_{i=1}^k \epsilon_i, \sum_{i=1}^k \delta_i\right)$-DP.

The definition of basic composition is saying that if we run k analyses on the same private (or sensitive) dataset, then the total output satisfies the sum of ϵ_i and δ_i differential privacy. On the surface, this may seem similar to the privacy guarantee of group privacy, but they are not the same and in fact, we can improve upon the basic composition (shown in Section 4.4).

Comments

Here, we expand upon the definition of post-processing and basic composition with examples. Suppose we solved for a vector of weights $\vec{\theta}$ using some differentially private estimation technique that satisfied (ϵ, δ)-DP, with

$$\vec{\theta} = [1, 2, 3, 4, 5]. \tag{6}$$

Let f be a function that takes in a vector $\mathbf{x} = [x_1, x_2, \ldots, x_N]$ and computes the mean:

$$f(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} x_i. \tag{7}$$

Then, due to post processing invariance, $f(\vec{\theta}) = 3$ is an output that also satisfies (ϵ, δ)-DP. Now, we realize this may have been a very crude example, but the idea is clear – we can take a function of any differentially private output and the new output would still satisfy the same privacy guarantees. For an audience familiar with Information Theory, one could think of this as the data processing inequality, where a function of the data cannot increase information about the dataset [6]. It is a similar notion here, where we cannot increase the privacy risk.

So what about basic composition? Let's go back to a machine learning example where we want to estimate parameters using gradient descent. Gradient descent has a parameter that is number of iterations, commonly referred to as "epochs". If one were to run T iterations of an (ϵ, δ)-DP mechanism, then the total privacy guarantee would be $(T\epsilon, T\delta)$ using basic composition. This can actually be very costly (in terms of privacy), and we will see more of how we can improve this in the next section.

4 Differential Privacy in Practice

Thus far, we have been talking a little abstractly about differential privacy and we hope this section will tie everything together. Specifically, we will show how we can use differential privacy in a machine learning setting.

4.1 Noise Mechanisms

In Section 2, we briefly talked about how we can ensure privacy by adding noise to a mechanism (or function). It turns out that there are many different methods to add noise, where each method satisfies a different level of privacy. In this tutorial, we discuss two noise mechanisms, the global sensitivity method and the exponential mechanism [5].

Global Sensitivity Method

Before explaining the global sensitivity method, we first give a quick primer on two probability density functions.

Definition 7 (Laplace Distribution)**.** The Laplace Distribution with mean 0 and scale b is the distribution with probability density function defined by

$$\mathsf{Lap}(x|b) = \frac{1}{2b} \exp\left(-\frac{|x|}{b}\right). \tag{8}$$

With a slight abuse of notation, we will often refer to the Laplace distribution without x and write $\mathsf{Lap}(b)$.

Definition 8 (Gaussian Distribution)**.** The Gaussian distribution with mean 0 and variance σ^2 is the distribution with probability density function defined by

$$\mathcal{N}(0, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{x^2}{2\sigma^2}\right). \tag{9}$$

Given these two probability density functions, we are now ready to introduce the Laplace and Gaussian mechanisms.

Definition 9 (Laplace Mechanism)**.** Let $f\colon \mathcal{X}^n \to \mathbb{R}^k$ be a real vector-valued function. The Laplace mechanism is defined as

$$\mathcal{M}(X) = f(X) + (Y_1, \ldots, Y_k), \tag{10}$$

where Y_i are independent $\mathsf{Lap}(\Delta_1(f)/\epsilon)$ random variables and $\mathcal{M}(X)$ is a privacy-preserving approximation of the function $f(X)$.

Recall that $\Delta_1(f)$ is the ℓ_1 sensitivity of the function f. If we add Laplace noise to function f with scale $\Delta_1(f)/\epsilon$, then our function $\mathcal{M}$ now satisfies $(\epsilon, 0)$-DP. We will prove this in the next Theorem.

Theorem 1. The Laplace mechanism preserves (or satisfies) $(\epsilon, 0)$-differential privacy [5].

Proof. The idea behind this proof is that for any neighbouring databases, we want to show that the ratio of their density functions is bounded by $\exp(\epsilon)$. Let X and Y be *any* two neighboring databases that differ in one entry. Then, let $p_X(z)$ and $p_Y(z)$ be the probability density functions of $\mathcal{M}(X)$ and $\mathcal{M}(Y)$ evaluated at a point $z \in \mathbb{R}^k$. Lastly, let Δ be the ℓ_1 sensitivity of f, i.e. $\Delta_1(f)$.

$$\frac{p_X(z)}{p_Y(z)} = \frac{\prod_{i=1}^{k} \exp\left(-\frac{\epsilon|f(X)-z_i|}{\Delta}\right)}{\prod_{i=1}^{k} \exp\left(-\frac{\epsilon|f(Y)-z_i|}{\Delta}\right)} \tag{11}$$

$$= \prod_{i=1}^{k} \exp\left(-\frac{\epsilon(|f(X)-z_i| - |f(Y)-z_i|)}{\Delta}\right) \tag{12}$$

$$\leq \prod_{i=1}^{k} \exp\left(-\frac{\epsilon(|f(X)-f(Y)|)}{\Delta}\right) \tag{13}$$

$$= \exp\left(\frac{\epsilon \sum_{i=1}^{k} |f(X)-f(Y)|}{\Delta}\right) \tag{14}$$

$$= \exp\left(\frac{\epsilon \|f(X)-f(Y)\|_1}{\delta}\right) \tag{15}$$

$$\leq \exp(\epsilon). \tag{16}$$

□

Let's try to digest this proof. Equation (11) simply comes from the definition of the Laplace distribution with scale Δ/ϵ. Equation (12) comes from writing (11) in terms of a single $\exp$, and equation (13) can be obtained by using the triangle inequality. Then by writing the product as a sum in $\exp$, we realize that this is actually the ℓ_1 sensitivity between the neighboring databases. Lastly, we can see that the ratio is upper bounded by $\exp(\epsilon)$. Hence, the Laplace mechanism satisfies $(\epsilon, 0)$-DP. Note that the absence of the δ term implies that $\delta = 0$.

We now move onto the Gaussian mechanism. In the Laplace mechanism, we added noise to a function by drawing independent noise from a Laplace distribution scaled to the sensitivity of the function. The Gaussian mechanism is the same, but we add noise drawn from a Gaussian distribution.

Definition 10 (Gaussian Mechanism)**.** Let $f\colon \mathcal{X}^n \to \mathbb{R}^k$ be a real vector-valued function. The Gaussian mechanism is defined as

$$\mathcal{M}(X) = f(X) + (Y_1, \ldots, Y_k), \tag{17}$$

where Y_i are independent $\mathcal{N}(0, 2\ln(1.25/\delta)\frac{\Delta_2(f)}{\epsilon^2})$ random variables and $\mathcal{M}(X)$ is a privacy-preserving approximation of the function $f(X)$.

Recall that $\Delta_2(f)$ is the ℓ_2 sensitivity of the function f. The Gaussian mechanism states that if we choose σ of our Gaussian noise $\mathcal{N}(0, \sigma^2)$ to be

$$\sigma \geq \frac{\Delta_2(f)}{\epsilon}\sqrt{2\log\frac{1.25}{\delta}}, \tag{18}$$

then our randomized algorithm $\mathcal{M}$ is (ϵ, δ)-DP. Since the proof for this is a little more involved, we leave it for the reader to see in the privacy book [5].

Comments

We dedicate this section to clear up any confusion the reader might have. Firstly, note that this is called "the global sensitivity method" as we are adding noise scaled to the sensitivity of the function in question. The Laplace mechanism satisfies $(\epsilon, 0)$-DP, whereas the Gaussian mechanism satisfies (ϵ, δ)-DP. With the slackness of δ with the Gaussian mechanism, it is generally more widely used and is more robust for a lot algorithms.

Let's do an example of "randomizing" a function using the Laplace and Gaussian mechanisms. Suppose we have a dataset $X = [x_1, x_2, \ldots, x_N]$, where each data sample x_i is a scalar in $[0, 1]$. Let the function f be a function that computes the mean of a dataset, i.e.

$$f(X) = \frac{1}{N} \sum_{i=1}^{N} x_i. \tag{19}$$

The ℓ_1 and ℓ_2 global sensitivity of f would be $1/N$, since the maximum change in the function would be 1 and the division of N comes from computing the mean. The Laplace mechanism would make a privacy-preserving approximation of f by adding Laplace noise from $\mathsf{Lap}(1/N\epsilon)$,

$$\mathcal{M}_1(X) = f(X) + Z, \tag{20}$$

where $Z \sim \mathsf{Lap}(1/N\epsilon)$. The function $\mathcal{M}_1(X)$ satisfies $(\epsilon, 0)$-DP.

The Gaussian mechanism would make a privacy-preserving approximation of f by adding Gaussian noise

$$\mathcal{M}_2(X) = f(X) + Z, \tag{21}$$

where $Z \sim \frac{1}{N\epsilon}\mathcal{N}(0, 2\ln(1.25/\delta))$. The function $\mathcal{M}_2(X)$ satisfies (ϵ, δ)-DP.

Exponential Mechanism

In this tutorial, we will only briefly cover the exponential mechanism. The exponential mechanism was designed for situations where adding noise (such as the Gaussian or Laplace mechanism) would completely destroy its value [5]. For example, the mean example from the previous subsection would have been okay, but what if we wanted compute the maximum revenue from an auction? If we were to make the prices from the items of an auction private, adding noise to these prices would heavily shift the total revenue far from the true revenue value. For these situations (and many others), we can use the exponential mechanism.

Definition 11 (Exponential Mechanism [5])**.** The exponential mechanism $\mathcal{M}_E(X, \mathcal{H}, u)$ selects and outputs an element $h \in \mathcal{H}$ with probability proportional to

$$p(h) \propto \exp\left(\frac{\epsilon u(X, h)}{2\Delta u}\right) \tag{22}$$

One can think of $X \in \mathcal{X}^n$ as a dataset, $h \in \mathcal{H}$ as a set of objects, and $u \colon \mathcal{X}^n \times \mathcal{H} \to \mathbb{R}$ as a score function. The score function takes in data X and an object h and outputs how "good" h is with respect to data point X. Δu is simply the sensitivity of the score function.

Theorem 2. The exponential mechanism $\mathcal{M}_E$ is $(\epsilon, 0)$-differentially private.

We leave the proof of this Theorem to the reader, but the proof is relatively easy to follow and is similar to the proof from the Laplace mechanism.

4.2 Differentially Private Empirical Risk Minimization

Before we get into the private version of empirical risk minimization (ERM), let's look at ERM in the non-private setting. Many machine learning problems involve estimating a parameter θ given a dataset D of $\{(x_i, y_i)\}_{i=1}^n$ pairs where $x_i's$ are data samples (or feature vectors) and $y_i's$ are labels. Given a loss function

ℓ that takes in parameter θ and (x_i, y_i), the ERM problem [7] solves for θ by minimizing the empirical loss given by

$$\mathcal{L}(\theta, D) = \frac{1}{n} \sum_{i=1}^{n} \ell(\theta, x_i, y_i). \tag{23}$$

To improve generalization performance and to mitigate "overfitting" [8], in practice, we often solve the regularized ERM problem given by

$$\mathcal{L}(\theta, D) = \frac{1}{n} \sum_{i=1}^{n} \ell(\theta, x_i, y_i) + \lambda N(\theta), \tag{24}$$

where λ is the regularization parameter and $N(\theta)$ is some function of θ. In most cases, we let $N(\theta) = \|\theta\|_1$ or $N(\theta) = \|\theta\|_2$ to induce sparsity. We can think of the term λ as the parameter that induces regularization, where $\lambda = 0$ would imply no regularization, and $\lambda = \infty$ would be ignoring the data entirely. The term λ is also often called a "penalty" term. The solution to the regularized ERM problem would be finding the θ that yields the lowest value of $\mathcal{L}(\theta, \mathcal{D})$, i.e.

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}}\, \mathcal{L}(\theta, \mathcal{D}) \tag{25}$$

$$= \underset{\theta}{\operatorname{argmin}}\, \frac{1}{n} \sum_{i=1}^{n} \ell(\theta, x_i, y_i) + \lambda N(\theta). \tag{26}$$

Now that we have a general idea of the non-private ERM framework, how can we make this process differentially private? There are actually two different answers called **output perturbation** and **objective perturbation**. However, in both cases, we need to add noise according to the sensitivity of the function! How can we compute the sensitivity of $\mathcal{L}(\theta, \mathcal{D})$? Sarwate et al. proved in the paper "Differentially Private Empirical Risk Minimization" [9] that under certain circumstances, the ℓ_2 sensitivity of $\mathcal{L}(\theta, \mathcal{D})$ is at most $\frac{2}{n\lambda}$, where n is the number of data points and λ is a convexity parameter. We provide a sketch "proof" here.

Let D and D' be two neighboring databases that differ in exactly one data entry. Specifically, let

$$D = \{(x_1, y_1), (x_2, y_2,), \ldots, (x_i, y_i), \ldots, (x_{n-1}, y_{n-1}), (x_n, y_n)\} \tag{27}$$

$$D' = \{(x_1, y_1), (x_2, y_2,), \ldots, (x_i, y_i), \ldots, (x_{n-1}, y_{n-1}), (x'_n, y'_n)\}. \tag{28}$$

If we plug in the two datasets into the ERM framework, that would yield

$$\mathcal{L}(\theta, D) = \frac{1}{n} \left[\sum_{i=1}^{n-1} \ell(\theta, x_i, y_i) + \ell(\theta, x_n, y_n) + \lambda N(\theta) \right] \tag{29}$$

$$\mathcal{L}(\theta, D') = \frac{1}{n} \left[\sum_{i=1}^{n-1} \ell(\theta, x_i, y_i) + \ell(\theta, x'_n, y'_n) + \lambda N(\theta) \right]. \tag{30}$$

If we subtract $\mathcal{L}(\theta, D')$ from $\mathcal{L}(\theta, D)$, then we can get the close to the definition of the sensitivity:

$$|\mathcal{L}(\theta, D) - \mathcal{L}(\theta, D')| = \frac{1}{n} |\ell(\theta, x_n, y_n) - \ell(\theta, x'_n, y'_n)|. \tag{31}$$

Lastly, if we make the assumption that each data point x_i was bounded such that $\|x_i\|_2 \leq 1$, then we get that the ℓ_2 sensitivity of $\mathcal{L}(\theta, D)$ is

$$\|\mathcal{L}(\theta, D) - \mathcal{L}(\theta, D')\|_2 = \frac{1}{n} \|\ell(\theta, x_n, y_n) - \ell(\theta, x'_n, y'_n)\|_2 \tag{32}$$

$$\approx \frac{2}{n}. \tag{33}$$

We use an approximation here, since we are missing a term λ that corresponds to $\mathcal{L}(\theta, D)$ being a λ-strongly convex function [9]. Since that might be slightly out of the scope of this tutorial, we omit it here.

Now that we know the sensitivity of $\mathcal{L}(\theta, D)$, let's take a quick look at output perturbation. Suppose we were able to solve for $\hat{\theta}$ by solving the non-private ERM problem,

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \mathcal{L}(\theta, \mathcal{D}). \tag{34}$$

We can make a privacy preserving approximation to $\hat{\theta}$, by adding Gaussian noise

$$\hat{\theta}_{\text{priv}} = \hat{\theta} + Z, \tag{35}$$

where Z is a random vector from distribution $\mathcal{N}(0, O(\frac{n^2 \log(1/\delta)}{\lambda^2 \epsilon^2}) \cdot I)$. Thus, $\hat{\theta}_{\text{priv}}$ satisfies (ϵ, δ)-DP. Hopefully this part was not too difficult to understand, as all we did was add Gaussian noise to the sensitivity of $\mathcal{L}(\theta, \mathcal{D})$. Note that this method is called "output perturbation", since we are adding noise and perturbing the output of the function.

We now shift gears to look at objective perturbation. Objective perturbation generally gives better results in terms of utility than output perturbation. If output perturbation adds noise to the output of the function, one could imagine that objective perturbation adds noise to the objective function itself. More precisely, consider the general regularized objective function from before,

$$\mathcal{L}(\theta, D) = \frac{1}{n} \sum_{i=1}^{n} \ell(\theta, x_i, y_i) + \lambda N(\theta). \tag{36}$$

If non-private ERM involves minimizing $\mathcal{L}(\theta, D)$ to find θ, then private ERM (DP-ERM) minimizes $\mathcal{L}_{\text{priv}}(\theta, D)$, where

$$\mathcal{L}_{\text{priv}}(\theta, D) = \frac{1}{n} \sum_{i=1}^{n} \ell(\theta, x_i, y_i) + \lambda N(\theta) + \langle b, \theta \rangle. \tag{37}$$

The b term here is a noise vector drawn from either a Gamma or Gaussian distribution, scaled to the sensitivity of the function. Finally, the DP-ERM solution is

$$\hat{\theta}_{\text{priv}} = \underset{\theta}{\operatorname{argmin}} \mathcal{L}_{\text{priv}}(\theta, D) \tag{38}$$

$$= \underset{\theta}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell(\theta, x_i, y_i) + \lambda N(\theta) + \langle b, \theta \rangle. \tag{39}$$

Comments

In this section, we learned two ways to make ERM private: output and objective perturbation. Output perturbation added noise to the output of the ERM problem, where objective perturbation made ERM private by solving a "noisy" objective function. Both methods can obtain (ϵ, δ)-DP, but output perturbation generally adds more noise and is shown to be empirically worse [9]. Additionally, a limitation of early DP-ERM work is that in order to guarantee privacy, one actually needed to find an *exact* minimum of the loss function. In practice, this might not be possible, but there have been some recent works that get around this with good results [10].

4.3 Differentially Private Stochastic Gradient Descent

Previously, we took a look at the ERM framework, where we wanted to estimate parameters through minimizing an objective function. But we never actually answered the critical question: "how do we minimize the function?" In some cases, the loss function may have a closed-form solution, and so one can find the parameter that minimizes a function by taking the gradient with respect to the parameter and setting it equal to zero. However, when the loss function does not have a closed-form solution, we need to find the

minimum through numerical optimization methods. Stochastic gradient descent (SGD) is an iterative algorithm that finds a minimum (perhaps a local minimum) of an objective function by taking a step in the opposite direction of the gradient of the loss function with respect to the parameter [11]. Specifically, given data $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ and an objective function $\mathcal{L}(\theta, \mathbf{x}_i)$ with parameter θ, the SGD algorithm on one iteration updates θ according to

$$\theta_{t+1} = \theta_t - \eta_t \nabla_\theta \mathcal{L}(\theta_t, \mathbf{x}_i), \tag{40}$$

where θ_t is the estimated parameter at iteration t, $\nabla_\theta \mathcal{L}(\theta, \mathbf{x}_i)$ is the gradient of the objective function with respect to θ computed on random sample $\mathbf{x}_i$, and η_t is the learning rate at iteration t. In some cases, the learning rate η may be fixed for all iterations, but it has been shown that changing the learning rate over time benefits training [12, 13]. In practice, we adapt to using a variation of SGD, mini-batch SGD, to speed up the training (or estimation) process. In mini-batch SGD, instead of making updates according to one random data sample, we make updates according to the average of several (or batch) of samples. The update rule in this algorithm is

$$\theta_{t+1} = \theta_t - \eta_t \left(\frac{1}{|B|} \sum_i \nabla_\theta \mathcal{L}(\theta_t, \mathbf{x}_i) \right), \tag{41}$$

where B is the batch size.

Of course, the next part of this section is to formulate private SGD. But one might wonder, "why do we want to make SGD private?". As we discussed in the previous Comments section, output perturbation generally does not provide good utility and objective perturbation requires us to find an *exact* minimum. If we instead train (or estimate) our models with private SGD, we can actually get around these limitations with good utility. The concept of making noisy gradient updates were proposed in [14–16], but we explain the differentially private SGD (DP-SGD) method proposed by Abadi et al. [17]. The DP-SGD algorithm is outlined in Algorithm 1. Upon computing the gradient $\mathbf{g}_t(\mathbf{x}_i) = \nabla_\theta \mathcal{L}(\theta, \mathbf{x}_i)$ on a batch of data samples, we make updates according to

$$\theta_{t+1} = \theta_t - \eta_t \left(\frac{1}{|B|} \sum_i \tilde{\mathbf{g}}_t(\mathbf{x}_i) + \mathcal{N}(0, \sigma^2 \alpha^2 \mathbf{I}) \right), \tag{42}$$

where $\tilde{\mathbf{g}}_t(\mathbf{x}_i)$ is the gradient clipped to norm α and α is the clipping bound.

Algorithm 1 Differentially Private SGD (DP-SGD) [17]

Input: Data $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, loss function $\mathcal{L}(\theta) = \frac{1}{N} \sum_i \mathcal{L}(\theta, \mathbf{x}_i)$;
Parameters: Learning rate η_t, noise variance σ^2, batch size B, gradient norm bound α, iterations T;
Initialize: θ_0 randomly
for $t \in T$ **do**
  Sample data:
  Create random batch B_t with sampling probability $q = \frac{B}{N}$
  Compute gradient:
  For each $\mathbf{x}_i \in B_t$, compute $\mathbf{g}_t = \nabla_\theta \mathcal{L}(\theta_t, \mathbf{x}_i)$
  Clip gradient:
  $\bar{\mathbf{g}}_t(\mathbf{x}_i) = \mathbf{g}_t(\mathbf{x}_i) / \max\left(1, \frac{\|\mathbf{g}_t(\mathbf{x}_i)\|_2}{\alpha}\right)$
  Average gradient and add noise:
  $\tilde{\mathbf{g}}_t = \frac{1}{|B|} \sum_i \tilde{\mathbf{g}}_t(\mathbf{x}_i) + \mathcal{N}(0, \sigma^2 \alpha^2 \mathbf{I})$
  Update gradient:
  $\theta_{t+1} = \theta_t - \eta_t \tilde{\mathbf{g}}_t$
end
Output: θ_T

Two steps that we take in DP-SGD that might not be so intuitive are random subsampling and gradient clipping. Random subsampling is the process of getting a random batch of data samples where we can

samples equal to the batch size in expectation. We actually need this sampling probability for the privacy analysis discussed later. The process of random subsampling is also related to something called "privacy amplification via subsampling" [18]. For gradient clipping, when we were looking at DP-ERM, we were able to easily compute the sensitivity of the function based on the condition that each data sample was bounded to satisfy $\|x_i\|_2 \leq 1$. Similarly, gradient clipping allows us to clip each gradient to norm α so that each gradient vector satisfies $\|g_t\|_2 \leq \alpha$. This way, the sensitivity of the gradient is α and adding noise of $\mathcal{N}(0, \sigma^2\alpha^2)$, where $\sigma = \sqrt{2\log\frac{1.25}{\delta}}/\epsilon$, guarantees us (ϵ, δ)-DP. However, it turns out that we're only guaranteed (ϵ, δ) for one iteration of DP-SGD! Then the obvious question is, what is the total (ϵ, δ) for running DP-SGD for T iterations? Using the basic composition property from Section 3, the total DP-SGD algorithm satisfies $(T\epsilon, T\delta)$-DP. Now, if you are new to differential privacy, this might not look too bad. In Section 4.4, we will show that $(T\epsilon, T\delta)$ is actually the worst pair of values one can get, and that we can significantly improve upon the basic composition.

Convergence Analysis

We make a quick note on the convergence of DP-SGD. From a theoretical point of view, one might wonder if DP-SGD has any convergence guarantees. There are many works that prove that DP-SGD surely converges, but we will demonstrate some guarantees from the paper "Understanding Gradient Clipping in Private SGD: A Geometric Perspective" [19]. Convergence analysis for SGD-type problems generally revolve around showing two things:

1. Show that the term $\mathbb{E}[\langle\nabla\mathcal{L}(\theta_t, \mathbf{x}_i), \tilde{\mathbf{g}}_t\rangle]$ diminishes to 0.
2. Show that $\mathbb{E}[\langle\nabla\mathcal{L}(\theta_t, \mathbf{x}_i), \tilde{\mathbf{g}}_t\rangle]$ is proportional to $\|\nabla\mathcal{L}(\theta_t, \mathbf{x}_i)\|^2$ or $c\|\nabla\mathcal{L}(\theta_t, \mathbf{x}_i)\|$.

In the points above, $\nabla\mathcal{L}(\theta_t, \mathbf{x}_i)$ is the true gradient vector, whereas $\tilde{\mathbf{g}}_t$ is the gradient vector with Gaussian noise after clipping. At a high level, this paper showed that if the distribution of the noisy gradients was approximately symmetric, then DP-SGD has the same convergence rate as SGD multiplied by a constant factor. We empirically show some of these plots in Section 5.

4.4 Rényi Differential Privacy & The Moments Accountant

As a warning, we believe this section will be more mathematical, but is necessary if one would like to use differential privacy efficiently in practice.

In Section 4.3, we saw that T iterations of the DP-SGD algorithm satisfied $(T\epsilon, T\delta)$-differential privacy using the basic composition scheme. That means if we wanted to have a privacy guarantee of $(0.1, 10^{-5})$ per iteration, then after running 200 iterations, our total privacy guarantee would be $(20, 20^{-2})$. This is a horrible guarantee! Remember that we want ϵ and δ to be as small as possible. One might ask, why don't we just decrease the ϵ value to $\epsilon = 0.01$ per iteration? Then, to satisfy $\epsilon = 0.01$ per iteration, it may be the case that we are adding too much noise that the utility is completely destroyed. So how do we deal with this?

In the same paper as DP-SGD [17], Abadi et al. proposed the moments accountant, that showed that the DP-SGD algorithm satisfied $(q\epsilon\sqrt{T}, \delta)$-DP compared to $(T\epsilon, T\delta)$-DP, where q is the sampling probability shown in Algorithm 1. Referring to the previous example, if we let the sampling probability to be $q = 1$ (no subsampling), then we get a bound of $(1.414, 10^{-5})$-DP. This is a significant improvement and is a promising privacy guarantee. What is the moments accountant that allows us to make this significant change? It turns out that the basic composition method was exaggerating the privacy loss, and that if we looked more closely at the distribution of the privacy loss random variable (explained in a bit), then we can get a better bound. This is often referred to as "advanced composition".

Let us consider a randomized mechanism $\mathcal{A}\colon \mathcal{D} \to \mathcal{T}$. For some outcome $o \in \mathcal{T}$ of the mechanism and neighboring datasets $D, D' \in \mathcal{D}$, the privacy loss random variable is defined as

$$Z = \log\frac{\Pr[\mathcal{A}(D) = o]}{\Pr[\mathcal{A}(D') = o]}. \tag{43}$$

Note that Z is simply the ratio of the two distributions under neighboring databases of the randomized mechanism. The idea behind the moments accountant [17] is to compute the moment generating function (MGF) of Z for each iteration, use composition to get the MGF of the complete algorithm and then use that to compute final privacy parameters (see Theorem 2 of [17]). The stepwise moment for any t at iteration j is defined [17] as

$$\alpha_j(t) = \sup_{D,D'} \log \mathbb{E}\left[\exp(tZ)\right]. \tag{44}$$

If the total number of iterations is T, then the overall moment is upper bounded as $\alpha(t) \leq \sum_{j=1}^{T} \alpha_j(t)$. Finally, for any given $\epsilon > 0$, the overall mechanism is (ϵ, δ) DP for $\delta = \min_t \exp\left(\alpha(t) - t\epsilon\right)$.

Let's try to digest what we have discussed so far. The privacy loss random variable, Z, can be obtained straight from the definition of differential privacy with $\delta = 0$. The definition of differential privacy (and hence, the basic composition scheme) is saying that we want to upper bound this privacy loss random variable so that $Z \leq \epsilon$. However, it turns out that this is a conservative bound, and Z is actually much smaller than ϵ "most of the time". Hence, by analyzing the MGF of Z, we can get a much tighter bound on ϵ, .e.g. $Z \leq \epsilon/10$. So, if we solve for t that minimizes the equation

$$\delta = \min_t \exp\left(\alpha(t) - t\epsilon\right), \tag{45}$$

where ϵ is fixed, then we get a δ value that is much tighter than using the basic composition method.

The next question we would like to answer is, how do we obtain the value t that minimizes δ? One could use numerical optimization methods, but it turns out that we can use some algebra to obtain t in closed-form. We derive this for the Gaussian mechanism. Now, for a Gaussian mechanism $\mathcal{A}(D) = f(D) + E$, where $E \sim \mathcal{N}(0, \sigma^2)$, the privacy loss random variable defined in (43) can be written as

$$Z = \log \frac{\exp\left(\frac{-(o-f_D)^2}{2\sigma^2}\right)}{\exp\left(\frac{-\left(o-f'_D\right)^2}{2\sigma^2}\right)} = \frac{\left(2o(f_D - f'_D) - (f_D^2 - {f'_D}^2)\right)}{2\sigma^2}. \tag{46}$$

Note that, the random variable o is Gaussian with $o \sim \mathcal{N}(f_D, \sigma^2)$. Therefore, it can be shown that the random variable Z is also Gaussian and

$$Z \sim \mathcal{N}\left(\frac{(f_D - f'_D)^2}{2\sigma^2}, \frac{(f_D - f'_D)^2}{\sigma^2}\right). \tag{47}$$

Now, from the moment generating function of generalized Gaussian, we have

$$\mathbb{E}[\exp(tZ)] = \exp\left(\frac{(f_D - f'_D)^2}{2\sigma^2}\ t + \frac{1}{2}\frac{(f_D - f'_D)^2}{\sigma^2}\ t^2\right) \tag{48}$$

$$= \exp\left(\frac{(f_D - f'_D)^2}{2\sigma^2}(t + t^2)\right). \tag{49}$$

If the $\mathcal{L}_2$ sensitivity of the function $\mathcal{A}(D)$ is Δ then

$$\alpha_j(t) = \sup_{D,D'} \log \mathbb{E}[\exp(tZ)] \tag{50}$$

$$= \sup_{D,D'} \left(\frac{(f_D - f'_D)^2}{2\sigma^2}(t + t^2)\right) \tag{51}$$

$$= \frac{\Delta^2}{2\sigma^2}(t + t^2). \tag{52}$$

We can compute the upper bound of the overall moment

$$\alpha(t) \leq \sum_{j=1}^{T} \alpha_j(t) = \frac{T\Delta^2}{2\sigma^2}(t + t^2). \tag{53}$$

Now, for any given $\epsilon > 0$, we have

$$\delta = \min_t \exp\left(\alpha(t) - t\epsilon\right) \tag{54}$$

$$= \min_t \exp\left(\frac{T\Delta^2}{2\sigma^2}(t + t^2) - t\epsilon\right). \tag{55}$$

All we did so far was write $\alpha(t)$ in a way we can solve for t in closed form. Solving for t using the equation in (55), the minimum t, $t_{\min}$, is

$$t_{\min} = \frac{1}{2}\left(\frac{2\epsilon\sigma^2}{T\Delta^2} - 1\right) \tag{56}$$

Plugging $t_{\min}$ back into equation (54), we get that the optimal δ value corresponding the a certain value of ϵ yields

$$\delta_{\text{opt}} = \exp\left(\frac{T\Delta^2}{2\sigma^2}(t_{\text{opt}} + t_{\text{opt}}^2) - t_{\text{opt}}\epsilon\right) \tag{57}$$

$$\implies \log \delta_{\text{opt}} = \frac{T\Delta^2}{2\sigma^2}\left[\frac{1}{2}\left(\frac{2\epsilon\sigma^2}{T\Delta^2} - 1\right) + \frac{1}{4}\left(\frac{2\epsilon\sigma^2}{T\Delta^2} - 1\right)^2\right] - \frac{\epsilon}{2}\left(\frac{2\epsilon\sigma^2}{T\Delta^2} - 1\right). \tag{58}$$

What if we wanted to go the other way around, where we fix a value of δ and wanted to compute a value ϵ? Then, we can re-arrange equation (57) to get ϵ:

$$\frac{\sigma^2}{2T\Delta^2}\epsilon^2 - \epsilon + \frac{T\Delta^2}{8\sigma^2} + \log \delta_{\text{target}} = 0 \tag{59}$$

$$\implies \epsilon = \frac{1}{2a}\left(-b \pm \sqrt{b^2 - 4ac}\right), \tag{60}$$

where $a = \frac{\sigma^2}{2T\Delta^2}$, $b = -1$ and $c = \frac{T\Delta^2}{8\sigma^2} + \log \delta_{\text{target}}$, and δ_{target} is our fixed δ value.

So how do we actually use this in practice? If you look at the expression for ϵ, then you can see that it is only a function of the sensitivity (Δ), the total number of iterations (T), the noise variance of the Gaussian mechanism (σ^2), and the fixed δ value. After running the DP-SGD algorithm, one can plug these values in to get a nice (ϵ, δ) bound. In Figure 3, we illustrate the difference in (ϵ, δ) for multiple composition techniques. Strong composition is something that we did not discuss, but it gives a slightly bigger bound than the moments accountant [20]. We will not discuss strong composition here. However, if you look at Figure 3 carefully, one can see that the "RDP" method gives the same performance as the moments accountant. This is because Rényi differential privacy (RDP) and the moments accountant are actually equivalent notions. Due to time constraints, we will unfortunately not be covering RDP, but we hope that the reader is interested enough to go read it themselves [21].

5 Privacy-Utility Tradeoffs

In this section, we show some tradeoffs between privacy and utility in practice. The experimental setup is the following:

- We implement regularized Logistic Regression with DP-SGD.
- We perform binary classification between two different classes of the MNIST dataset.
- We preprocess the MNIST dataset by normalizing all of the pixel values to $[0, 1]$.
- We use a decaying learning rate of $\eta_t = \frac{1}{\sqrt{t}}$, where t is the iteration number.
- We use a batch size of 150, and hence a sampling probability of $150/N$, where N is the total number of training data samples.

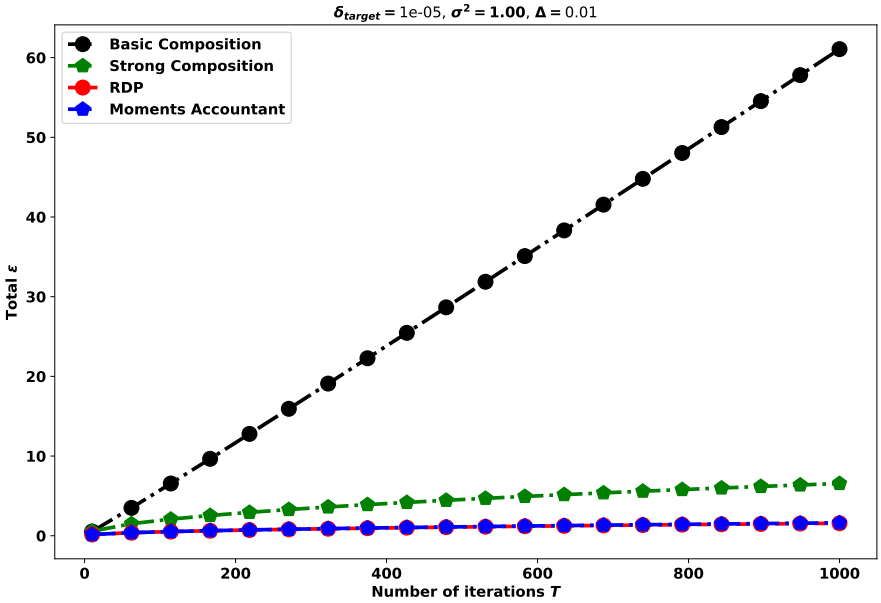

Figure 3: Comparison of different composition (or accounting) techniques for ϵ and δ. For fixed values $\delta_{\text{target}} = 10^{-5}$, $\sigma^2 = 1$, $\Delta = 0.01$, $T = 1000$, the moments accountant and RDP methods provide a much smaller overall ϵ the the basic composition.

- We add different values (variances) of Gaussian noise per run to demonstrate how the utility decreases as privacy increases.
- We fix δ as $\delta = 10^{-4}$. In practice, we try to set $\delta \approx 1/N$, where N is the total number of training samples.
- We use the moments accountant to compute ϵ given sampling probability $150/N$.

For all experiments, we use a Python environment on a Macbook Pro with 2.2 GHz Intel Core i7 and 16 GB RAM.

In Figure 4, we can see that the non-private baseline performs much better in terms of prediction accuracy. For a relatively large value of ϵ, we can get an accuracy near 95% and for a smaller ϵ value, we can get an accuracy near 85%. This is a significant change, but one can try to improve these accuracies by playing with the regularization parameter and possibly more iterations. Since we are adding noise on each step of SGD, it is clear that DP-SGD will converge slower than SGD. In Figure 5, we compute the Cosine Distance between the true gradient and the noisy gradient to try and observe the symmetry of the gradient distribution. It turns out that if we use a full batch size, then the gradient distribution looks approximately symmetric, and almost like a normal distribution! Referring back to the convergence analysis on DP-SGD, this gives us a hint that DP-SGD surely converges. In Figure 6, we project the gradients onto a two-dimensional space using random matrices. Upon projection, we plot the gradients in a 2D space to observe that this also looks approximately symmetric.

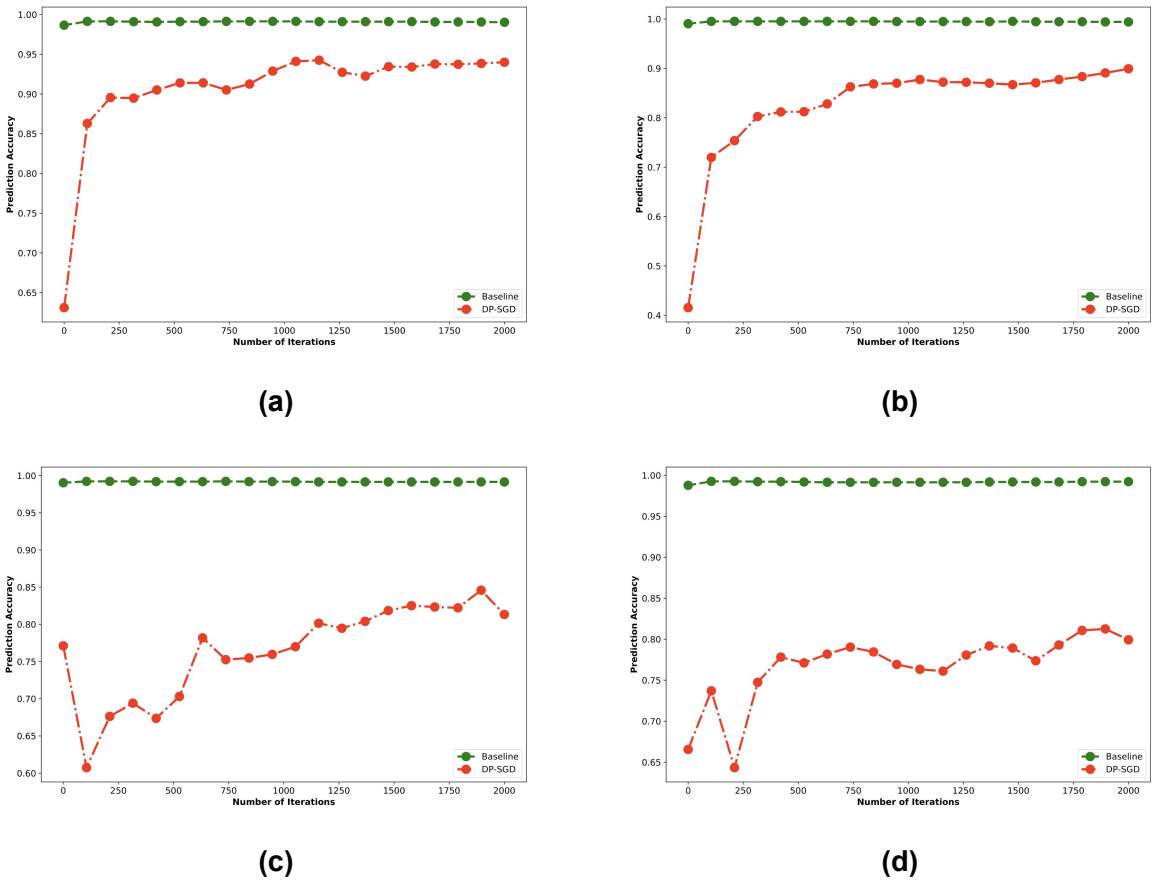

Figure 4: Four different runs of the Logistic Regression experiment with DP-SGD and their test accuracies. (a) Variance of $\sigma^2 = 1$ and obtains $(17.865, 10^{-4})$-DP, (b) Variance of $\sigma^2 = 5$ and obtains $(4.275, 10^{-4})$-DP, (c) Variance of $\sigma^2 = 7.5$ and obtains $(3.045, 10^{-4})$-DP, (d)Variance of $\sigma^2 = 10$ and obtains $(2.4, 10^{-4})$-DP.

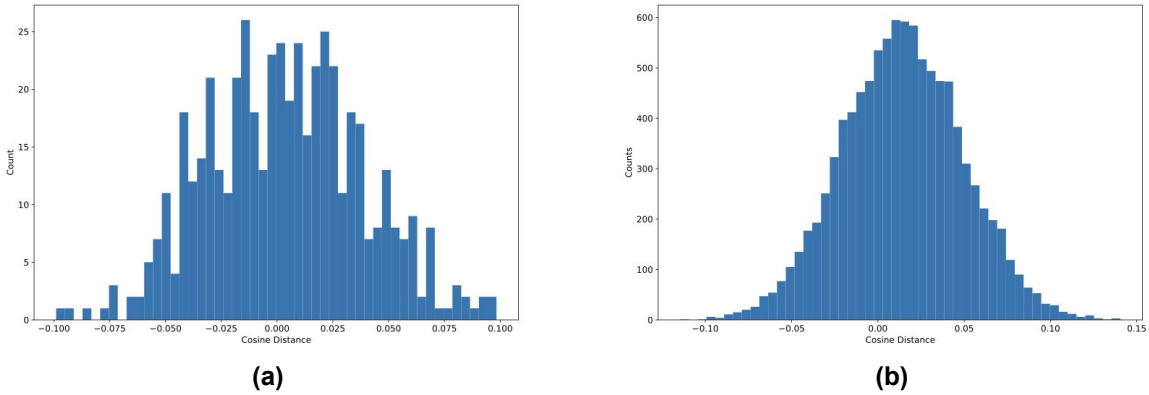

Figure 5: Histogram of the cosine distance between true and noisy gradients. (a) Histogram of the cosine distance of approximately 250 gradient vectors, (b) histogram of the cosine distance of all of the gradient vectors.

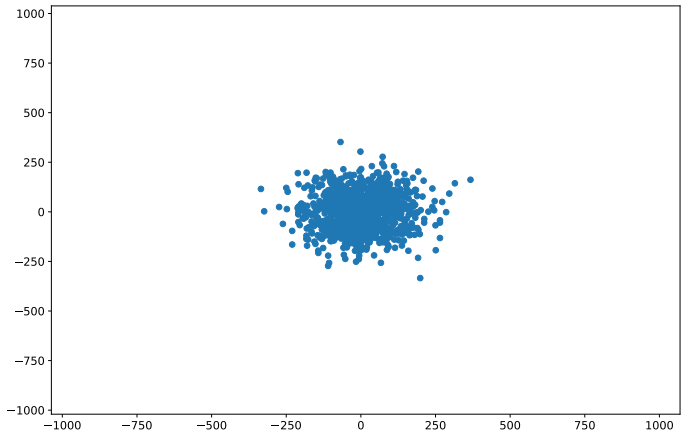

Figure 6: Projection of approximately 250 gradients vectors onto a two-dimensional space using random matrices. The approximate symmetry of the distribution hints at the convergence of DP-SGD.

6 Additional Comments

This concludes our friendly tutorial of differential privacy, but we wanted to point out some topics that we were not able to discuss. We think that the topics of the exponential mechanism, privacy amplification via subsampling, and Rényi differential privacy were only partially mentioned, but not explained detail. We strongly urge the reader to take the time to read up on these topics, as they are really important in using differential privacy in practice. For suggestions, comments, and questions, please feel free to contact `smk330@scarletmail.rutgers.edu`.

References

[1] J. Bennett and S. Lanning, "The netflix prize," in *Proceedings of the KDD Cup Workshop 2007*. New York: ACM, Aug. 2007, pp. 3–6. [Online]. Available: http://www.cs.uic.edu/~liub/KDD-cup-2007/NetflixPrize-description.pdf

[2] A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets," in *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, ser. SP '08. USA: IEEE Computer Society, 2008, p. 111–125. [Online]. Available: https://doi.org/10.1109/SP.2008.33

[3] N. Homer, S. Szelinger, M. Redman, D. Duggan, W. Tembe, J. Muehling, J. V. Pearson, D. A. Stephan, S. F. Nelson, and D. W. Craig, "Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays," 2008. [Online]. Available: https://doi.org/10.1371/journal.pgen.1000167

[4] N. Carlini, C. Liu, Úlfar Erlingsson, J. Kos, and D. Song, "The secret sharer: Evaluating and testing unintended memorization in neural networks," 2019.

[5] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, no. 3–4, p. 211–407, Aug. 2014. [Online]. Available: https://doi.org/10.1561/0400000042

[6] T. M. Cover and J. A. Thomas, *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. USA: Wiley-Interscience, 2006.

[7] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: With Applications in R*, 2014.

[8] X. Ying, "An overview of overfitting and its solutions," *Journal of Physics: Conference Series*, vol. 1168, p. 022022, 02 2019.

[9] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially private empirical risk minimization," *J. Mach. Learn. Res.*, vol. 12, no. null, p. 1069–1109, Jul. 2011.

[10] R. Iyengar, J. P. Near, D. Song, O. Thakkar, A. Thakurta, and L. Wang, "Towards practical differentially private convex optimization," in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 299–316.

[11] S. Ruder, "An overview of gradient descent optimization algorithms," 2017.

[12] A. Lalitha, T. Javidi, and A. D. Sarwate, "Social learning and distributed hypothesis testing," *IEEE Transactions on Information Theory*, vol. 64, no. 9, pp. 6161–6179, 2018.

[13] Y. Wu, L. Liu, J. Bae, K.-H. Chow, A. Iyengar, C. Pu, W. Wei, L. Yu, and Q. Zhang, "Demystifying learning rate policies for high accuracy training of deep neural networks," *2019 IEEE International Conference on Big Data (Big Data)*, pp. 1971–1980, 2019.

[14] O. Williams and F. Mcsherry, "Probabilistic inference and differential privacy," in *Advances in Neural Information Processing Systems*, J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, Eds., vol. 23. Curran Associates, Inc., 2010. [Online]. Available: https://proceedings.neurips.cc/paper/2010/file/fb60d411a5c5b72b2e7d3527cfc84fd0-Paper.pdf

[15] S. Song, K. Chaudhuri, and A. D. Sarwate, "Stochastic gradient descent with differentially private updates," in *2013 IEEE Global Conference on Signal and Information Processing*, 2013, pp. 245–248.

[16] R. Bassily, A. Smith, and A. Thakurta, "Private empirical risk minimization: Efficient algorithms and tight error bounds," ser. FOCS '14. USA: IEEE Computer Society, 2014, p. 464–473. [Online]. Available: https://doi.org/10.1109/FOCS.2014.56

[17] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, Oct 2016. [Online]. Available: http://dx.doi.org/10.1145/2976749.2978318

[18] Y.-X. Wang, B. Balle, and S. P. Kasiviswanathan, "Subsampled renyi differential privacy and analytical moments accountant," ser. Proceedings of Machine Learning Research, K. Chaudhuri and M. Sugiyama, Eds., vol. 89. PMLR, 16–18 Apr 2019, pp. 1226–1235. [Online]. Available: http://proceedings.mlr.press/v89/wang19b.html

[19] X. Chen, Z. S. Wu, and M. Hong, "Understanding gradient clipping in private sgd: A geometric perspective," 2021.

[20] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating Noise to Sensitivity in Private Data Analysis," in *Proceedings of the Third Conference on Theory of Cryptography*, 2006, pp. 265–284. [Online]. Available: http://dx.doi.org/10.1007/11681878_14

[21] I. Mironov, "Rényi differential privacy," *CoRR*, vol. abs/1702.07476, 2017. [Online]. Available: http://arxiv.org/abs/1702.07476