Differential Privacy

6.1600 Course Staff

Fall 2024

In many of the systems we have seen so far—encryption, iOS security, etc—we try to prevent an adversary from learning *any information* about our sensitive data. Today we will be talking about how to deal with situations in which we *want* to leak some information about sensitive user data, but we want to do it in a "privacy-preserving" way.

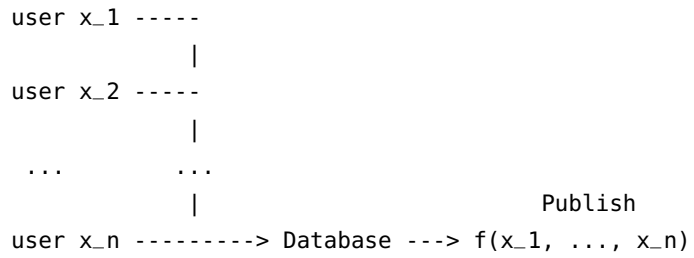
Disclaimer: This set of notes is a work in progress. It may have errors and be missing citations. It is certainly incomplete. Please let the staff know of any errors that you find.

For example:

- the U.S. Census collects sensitive demographic information about U.S. citizens and then publishes aggregate statistics about it,
- Internet Service Providers may want to publish aggregate statistics about network traffic,
- Google may want to publish anonymized data about user search queries, or
- public-health officials may want to publish metrics about a population's health at large without revealing any individual user's health status.

In each of these settings, we many n users, where each user i has some data x_i. There is a central party that has collected $(x_1, \ldots, x_n)$ and wants to publish an aggregate statistic $f(x_1, \ldots, x_n)$ in a way that "protects the privacy" of each user's data.

The techniques we describe here also apply when there is only a single user and the user wants to publish some function about their private data.



Q1. What functions of private data can we publish without compromising user privacy? How do we even define "privacy" in this context?

Q2. Once we have a definition of privacy, how do we achieve it in practice?

Differential privacy gives us *one possible* answer to the question Q1 here—it is a *definition* of privacy that has an number of appealing

properties. For reasons we will discuss, it is not a perfect definition of privacy, but it is essentially the best one we have.

Once we buy the privacy definition that differential privacy gives us, there is a rich literature that explains mechanisms for achieving differential privacy (i.e., the answer to question Q2).

1 *A bad idea: Attempt to anonymize the data*

A common **bad** strategy for attempting to release data with privacy protections is to try to *anonymize* the data. For example, if a hospital wants to release medical-records data, they could publish the records with the names redacted. One surprise—that has bitten many companies and governments—is that it is shockingly easy to re-identify data in datasets that are supposedly anonymized.

Example: Re-identification by Linking (Sweeney 1997) For example, in Massachusetts, the Group Insurance Commission released a dataset that they believed to be anonymized. This dataset contained health data, including patient ethnicity, ZIP code, birth date, sex, date of visit, diagnosis, procedure, and medication given.

In 1997, Latanya Sweeney demonstrated that this anonymization strategy completely failed. In particular, she purchased (for $20!) the public voter-registration data from the government of Cambridge, MA. This second dataset included voter name, address, ZIP code, birth date, and gender. Crucially, this other database included each voter's zip code, birth date, and gender! Using the voter-registration dataset—which linked names to zip codes and birth dates—and the medical data set—which linked zip codes and birth dates to medical conditions—Sweeney was able to determine which people had which medical conditions. That is, Sweeney was able to completely de-anonymize the GIC dataset and reveal private medical information for everyone up to the governor of Massachusetts.

Netflix Competition In 2006, Netflix aimed to improve their movie recommendation system. To do this, they planned to have researchers compete to come up with the best movie-recommendation algorithms. For the purposes of running the competition, Netflix published a supposedly anonymized dataset that included a randomized user id, movie id, rating, and date. Netflix released a portion of the dataset, and the research group that could produce a model that best predicted ratings for the unreleased set of movies would win a million dollars.

One group of researchers was able to link individual records in the Netflix database to records from the public IMDb database. This

allowed them to de-anonymize the data from Netflix even though the database contained no identifying information whatsoever!

The bottom line. Anonymizing data sets simply does not work.

2 *A flawed idea: k-Anonymity*

The k-anonymity approach to privacy is to think of a dataset release as being sufficiently private if there are at least k users' data in the dataset that are identical. For example, if there are $k = 10$ people in a medical-records database with the same birth date, zip code, and medical condition, we might be okay with publishing the dataset with those three fields only when there are at least $k = 10$ users whose medical records are identical on those three fields.

The flawed intuition here is that even if there is still some leakage, maybe it is not so bad. For example, even if you can link the voter-registration database to the medical-records database, you might hope that it will be difficult to figure out exactly which user has exactly which medical condition.

There are a few issues with the k-anonymity approach to privacy:

- It is not robust to *side information*. For example, if an attacker knows a few people with a given medical condition, it can use a process of elimination to de-anonymize the dataset still.
- The anonymized dataset may still leak *some private information*. For example, say that every student in a class of 100 got the same grade. The teacher might be happy publishing the grades of all students (without names) since this release satisfies $k = 100$-anonymity. The problem is that the anonymized release reveals the *grade of every student in the class*—definitely not exactly what we would expect from a privacy-preserving release.

3 *A flawed idea: Publish only aggregate statistics*

Another approach we might consider is to publish only summary statistics with the aim that these summary statistics compress the data so much that it is impossible to learn anything meaningful about an individual from them. However, we have to be careful: even a few statistics can reveal sensitive information.

For example, consider a company that released the average salary of its employees regularly. If the company releases this average before and after the resignation of one individual, anyone who knows that that individual resigned can learn his salary.

So, releasing only statistics does not cleanly protect privacy either.

4 A new definition: Differential privacy

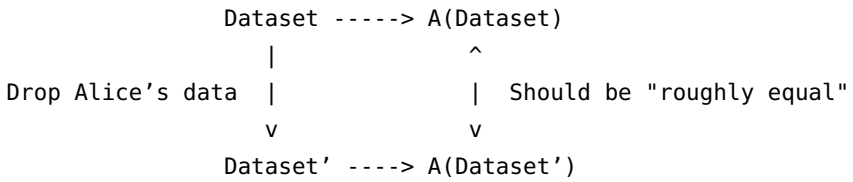
Differential privacy is a very mathematically clean definition of privacy that addresses many of the concerns of the flawed ideas above. It is not perfect—for reasons we will see—but many regard it as the best formal definition of privacy we have.

Say that $(x_1, \ldots, x_n)$ is a dataset of n users' data. Differential privacy, roughly, says that a it is safe to release an algorithm $A(x_1, \ldots, x_n)$—called a *mechanism* in the language of differential privacy—applied to the dataset if the value of the algorithm A's output is roughly the same if we release any user i's data x_i with some other data value x_i^*:

$$A(x_1, \ldots, x_n) \approx A(x_1, \ldots, x_{i-1}, x_i^*, x_{i+1}, \ldots, x_n).$$

In other words, dataset-release mechanism provides differential privacy if its output looks roughly the same, whether or not a particular user's data is included in the original dataset.

In picture form, the situation looks like this:



To formalize this, we will require that a pair of datasets that differ in only a single row—that is, one contains a row and the other does not—are *close*. The formal definition is as follows:

Notice that a mechanism A that outputs nothing trivially satisfies differential privacy. For a mechanism to be useful, it should add as little noise as possible to maintain a given level of differential privacy.

Definition 4.1 (ϵ-differential privacy)**.** An mechanism A provides ϵ-differentially private if, for all neighboring datasets x and x' that differ in only one row, for all subsets S of outputs of A:

$$\Pr[A(x) \in S] \leq e^{\epsilon} \cdot \Pr[A(x') \in S].$$

This is approximately saying that whether or not a given row is included in the dataset, the probability of the output being detectably different is less than ϵ.

We have that $e^{\epsilon} \approx 1 + \epsilon$ when ϵ is very small.

In the differential-privacy literature, this parameter ϵ is often called the "privacy budget."

One of the delicate points in practice is figuring out what actual value of ϵ a system should aim to achieve.

Notice that the definition of differential privacy is parameterized by a real number ϵ that specifies how close the two outputs of the mechanism A must be on neighboring datasets. The smaller ϵ is, the stronger the privacy guarantee is.

In order for a non-trivial mechanism to provide differential privacy, the algorithm must be randomized. Take our salary example from before: if we want to reveal an average salary of all employees, achieving differential privacy requires adding *noise* in order to

make sure that the output of the release algorithm looks roughly the same whether or not a particular employee's salary is included in the computation.

Differential privacy is closed under post-processing. A really convenient property of the differential-privacy view of privacy is that the definition is *closed under post processing*. That is, if a mechanism A is ϵ-differentially private, then for *every* function B, the composed mechanism $B(A(\cdot))$ is also ϵ-differentially private.

Differential privacy is composes nicely. In many applications, we want to publish many functions of the same dataset. An very powerful property of the definition of differential privacy is that it can nicely handle this situation. In particular, say that a mechanism A_1 is a mechanism that satisfies ϵ_1-differential privacy. Say that a mechanism A_2 is a mechanism that satisfies ϵ_2-differential privacy. Then the mechanism $B(x) = (A_1(x) \| A_2(x))$ that publishes the output of both mechanisms is $(\epsilon_1 + \epsilon_2)$-differentially private.

There are more sophisticated composition theorems that give tighter bounds on the differential privacy parameter of the composed mechanism when the number of mechanisms you are composing is large.

The value of ϵ increases with each statistic we publish—which nicely captures our intuition that the more statistics you release about a dataset, the more information you are leaking about it.

5 *Achieving differential privacy: Adding noise*

A common approach to achieving differential is exactly this: adding random noise to the data in order to add uncertainty about the real data. The first instance of this was by Warner in 1965, who proposed using random noise to allow individuals to answer sensitive survey questions (i.e., revealing some private data) while also giving them some privacy protection. Warner's approach is called *randomized response*.

To give an example of randomized response: consider a professor that wants to learn the fraction of students who cheated on a test. Obviously no student wants to admit that they cheated on an exam. Using Warner's approach, the professor could ask students to answer honestly with probably $2/3$, but to lie with probability $1/3$. This allows a student who did cheat to claim that they were lying, as per the directions. However, with many students, the noise should average out, and the professor can still learn approximately how many students cheated on the test.

We can apply this same approach if we want to make a certain algorithm differentially private. Instead of publishing a dataset (or a function of a dataset) directly, we can publish a *noisy version of it*. Much of the technical complexity in differential privacy goes

into trying to design exactly how to choose the noise in a way that maintains as much of the "signal" of the underlying dataset as possible.

6 The Laplace mechanism

The *Laplace mechanism* gives a very general way to take an aggregate statistic f and modify it to have ϵ-differential privacy for any choice of ϵ.

First, we need to define the *global sensitivity* of the function f. This aims to capture the maximal change in f that results from changing one of the function's n inputs. Intuitively, the higher the sensitivity of a function is, the more each user's data can affect the functions output and therefore the more noise we will have to add to achieve differential privacy.

The definition of sensitivity and the Laplace mechanism generalize nicely to functions that output multiple real numbers.

Definition 6.1 (Global Sensitivity)**.** The global sensitivity of a function $f\colon \mathcal{D}^n \to \mathbb{R}$, for some domain $\mathcal{D}$, is:

$$GS_f = \max_{\text{neighbors } x,x' \in \mathcal{D}^n} \|f(x) - f(x')\|.$$

Here, "neighbors" inputs in $\mathcal{D}^n$ that differ in only a single coordinate.

We will also need to define the zero-mean real-valued Laplace distribution, which is parameterized by a value $b \in \mathbb{R}$. The probability distribution function is:

$$h(y) = \frac{1}{2b} e^{-\frac{|y|}{b}}$$

Now, the Laplace mechanism for achieving a differential privacy release of the function $f(x_1, \ldots, x_n)$ is:

1. Compute the true statistic $y \leftarrow f(x_1, \ldots, x_n)$.
2. Let $b \leftarrow \frac{GS_f}{\epsilon}$ be the global sensitivity of f.
3. Sample a random noise value ν from the Laplace distribution with mean 0 and parameter b.
4. Output the noised statistic $y + b$.

Handling non-numerical data. The Laplace mechanism gives a very clean way to provide differential privacy for releasing real-valued aggregate statistics about a dataset—where the aggregate statistic is just a number. But in many applications, we need to publish *strings*, such as names or zip codes or birth dates.

One way to handle strings is to convert them into numerical statistics and them apply differential privacy. For example, we could

publish (1) the number of users with name "Alice," (2) the number of users with name "Bob," and so on. Then we are back to working in a world of numerical statistics and we can against use the Laplace mechanism.

7 *Challenges with differential privacy*

As we discussed in Section 4, the more statistics you publish about a database, the larger the differential-privacy parameter ϵ grows. A consequence is that if you want to publish many statistics about a dataset, you will have to add a huge amount of noise to the statistics you publish to achieve ϵ-differential privacy any reasonable value of ϵ. This is a major headache in practice.

A second issue is that the designer of the system has to choose what value of ϵ to use. What value is good enough? The theory of differential privacy cannot tell you—it's really a subjective question.

Yet a third issue is that in many applications, a company has to publish an aggregate statistic every day or week. Since the ϵ values "add up," after a short amount of time the effective ϵ value can end up being so large as to render the differential privacy guarantee meaningless.