Privacy in Statistics and Machine Learning — Spring 2023
Lecture 5: Differential Privacy II

Adam Smith (based on materials developed with Jonathan Ullman)

1 Properties of DP: Composition and Postprocessing

Notions of security should not be too fragile. If we argue that something is secure in isolation it should still be the case that it is secure in the real world where more than one algorithm or protocol is being run. One type of robustness we need is security under *composition*. Let's illustrate this with two scenarios:

Scenario 1 Suppose two hopsitals hold overlapping data sets $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ and that each hopsital runs a differential private algorithm (separately from the other hospital) on its data set. So the hospitals end up releasing $A_1(\mathbf{x}^{(1)})$ and $A_2(\mathbf{x}^{(2)})$. For the individuals whose records are in both data sets, what sort of prvacy guarantee can we make if an outside attacker see the output of both algorithms?

Scenario 2 Suppose I have one data set $\mathbf{x}$, and I run an ε_1 differentially private algorithm A_1 to get output a_1. For example, maybe A_1 estimates two counting queries: the number of diabetics in the data set, and the number of people with high blood pressure. Based on the first answer a_1, I choose a second algorithm $A_2^{(a_1)}$ that is ε_2-DP. For example, maybe if both counts in a_1 are at least 100, I will ask A_2 to estimate the number of people who have diabetes *and* high-blood pressure, but if one of the counts is small, I will instead ask about the number of people with heart disease. I run $A_2^{(a_1)}$ to get a_2 and output both values a_1, a_2.

Both of these senarios are cases of composition. To simplify things a bit, think of the two data sets in 'Scenario 1' as one big dataset $\mathbf{x}$ containing the information from both hospitals; the set of people with records in $\mathbf{x}$ would be the union of the sets of people in with records in $\mathbf{x}_1$ and $\mathbf{x}_2$. So what the adversary sees in Scenario 1 is the outcome of one big algorithm $A(\mathbf{x}) = (A_1(\mathbf{x}), A_2(\mathbf{x}))$. Similarly, in 'Scenario 2' we can view the final output as the outcome of one big algorithm A which includes the decision of which statistics to ask for as part of the input to A_2. Figure 1 captures both settings, showing the combined algorithm as a dashed box.

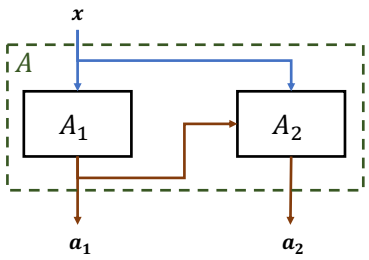

Figure 1: The adaptive composition of two algorithms (solid boxes), viewed as a single larger algorithm A (dashed box).

Composition thus covers two apparently different phenomena: first, my data is used by many organizations, and I should consider what is leaked by the combination of releases that concern me; second, we would like to design algorithms and workflows modularly, with outputs of early stages feeding in to decisions made later on.

Differential privacy offers the following convenient guarantee: if A_1 and A_2 are respectively ε_1 and ε_2 differentially private, then A is ε'-DP with $\varepsilon' \leq \varepsilon_1 + \varepsilon_2$. To set up the general formalism, let $A_1 : \mathcal{U}^n \to \mathcal{Y}_1$ be ε_1-DP, and let $A_2 : \mathcal{Y}_1 \times \mathcal{U}^n \to \mathcal{Y}_2$ be ε_2-DP for all values of its first input (that is $A_2(a_1, \cdot)$ is ε_2-DP for every value of a_1). This means that A_2 runs a DP algorithm but exactly which algorithm that is depends on a_1.

Lemma 1.1. *Let $A : \mathcal{U}^n \to cY_1 \times \mathcal{Y}_2$ be the randomized algorithm that outputs $A(\mathbf{x}) = (a_1, a_2)$ where $a_1 = A_1(\mathbf{x})$ and $a_2 = A_2(a_1, \mathbf{x})$. Then A is $(\varepsilon_1 + \varepsilon_2)$-DP.*

Proof. We prove the discrete case here, for simplicity. Let $\mathbf{x}, \mathbf{x}'$ be neighboring data sets in $\mathcal{U}^n$, and let $a = (a_1, a_2)$ be an outcome in $\mathcal{Y}_1 \times \mathcal{Y}_2$.

$$\mathbb{P}\left(A(\mathbf{x}) = (a_1, a_2)\right) = \mathbb{P}\left(A_1(\mathbf{x}) = a_1\right) \cdot \mathbb{P}\left(A_2(\mathbf{x}, a_1) = a_2\right) \tag{1}$$

Since A_1 is ε_1-DP, and $A_2(a_1, \cdot)$ is ε_2-DP for every choice of a_1, we can bound the probability above.

$$\begin{aligned}\mathbb{P}\left(A(\mathbf{x}) = (a_1, a_2)\right) &\leq e^{\varepsilon_1}\mathbb{P}\left(A_1(\mathbf{x}') = a_1\right) \cdot e^{\varepsilon_2}\mathbb{P}\left(A_2(\mathbf{x}', a_1) = a_2\right) \\ &= e^{\varepsilon_1+\varepsilon_2} \cdot \mathbb{P}\left(A(\mathbf{x}') = (a_1, a_2)\right)\end{aligned}$$

□

The proof applies to arbitrary sets by writing a set E as a union of singletons $\{(a_1, a_2)\}$, as in the proof of randomized response (and Exercise 3.3) from Lecture 4.

By induction, we get a general-purpose way to analyze complex algorithms with many stages. We'll write it here, and leave the proof as an exercise.

Lemma 1.2 (Basic Composition)**.** *Let $A_1, A_2, ..., A_k$ be a sequence of randomized algorithms, where $A_1 : \mathcal{U}^n \to \mathcal{Y}_1$ and $A_i : \mathcal{Y}_1 \times \cdots \mathcal{Y}_{i-1} \times \mathcal{U}^n \to \mathcal{Y}_i$ for $i = 2, 3, ..., k$ (so A_i takes as input elements that could have been output by $A_1, ..., A_{i-1}$, as well as a data set in $\mathcal{U}^n$). Suppose that for each $i \in [k]$, for every $a_1 \in \mathcal{Y}_1, a_2 \in \mathcal{Y}_2, ..., a_{i-1} \in \mathcal{Y}_{i-1}$, we have that $A_i(a_1, ..., a_{i-1}, \cdot)$ is ε_i-DP. Then the algorithm $A : \mathcal{U}^n \to \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_k$ that runs the algorithms A_i in sequence is ε-DP for $\varepsilon = \sum_{i=1}^{k} \varepsilon_i$.*

Exercise 1.3. Prove Lemma 1.2.

The Basic Composition lemma allows us to design complex algorithms by putting together smaller pieces. We can view the overall privacy parameter ε as a budget to be divided among these pieces. We will thus often refer to ε as the "privacy budget": each algorithm we run leaks some information, and consumes some of our budget. Differential privacy allows us to view information leakage as a resource to be managed.

Exercise 1.4. Suppose we run k executions of the Laplace mechanism on the same data set, dividing the privacy budget equally among them. By what factor does the magnitude of the Laplace noise increase?

Exercise 1.5. Sometimes it is much better to analyze an algorithm as a whole than to use the composition lemma. Consider the histogram example from Lecture 4, where $\mathcal{U}$ is written as a partition of disjoint sets $B_1, B_2, ..., B_d$, and we want to count how many records lie in each set. Viewed as one d-dimensional function, the histogram has global sensitivity 2. We could also view it as d separate functions $n_1, n_2, ..., n_d$, each with globabl sensitivity 1. How much noise would the Laplace mechanism add to these counts if we ran it spearately for each of the n_j with privacy budget divided equally among them? How does that compare to running the Laplace mechanism once on the joint function?

1.1 Postprocessing

Now one question that might come up is whether it's ok in Figure 1 to release only part of A's output. For instance, what if we release only a_2? Or perhas some function of both a_1 and a_2? It turns out that the privacy guarantee never gets worse when we release a function of the output.

Lemma 1.6 (Closure under postprocessing)**.** *Let $A : \mathcal{U}^n \to \mathcal{Y}$ and $B : \mathcal{Y} \to \mathcal{Z}$ be randomized algorithms, where $\mathcal{U}, \mathcal{Y}, \mathcal{Z}$ are arbitrary sets. If A is ε-differentially private, then so is the composed algorithm $B(A(\cdot))$.*

Proof. Let's first prove the lemma for the case where B is *deterministic*. In that case, the event $B(A(\mathbf{x})) = b$ is the same as the event $A(\mathbf{x}) \in B^{-1}(b)$ where $B^{-1}(b)$ is the preimage of b under B. So we can just apply the A's DP guarantee to $B^{-1}(b)$:

$$\mathbb{P}\left(B(A(\mathbf{x})) = b\right) = \mathbb{P}\left(A(\mathbf{x}) \in B^{-1}(b)\right) \leq e^{\varepsilon}\mathbb{P}\left(A(\mathbf{x}') \in B^{-1}(b)\right) = e^{\varepsilon}\mathbb{P}\left(B(A(\mathbf{x})) = b\right). \tag{2}$$

To handle the case where B is randomized, we can write the $B(a)$ as the application of a *deterministic* function f applied to the pair (a, R) where R is a random variable independent of a that represents B's random choices.

Thus, $B(A(\cdot))$ is the application of a deterministic function to $A'(\mathbf{x}) = (A(\mathbf{x}), R)$. The algorithm A' is ε-DP (since R is independent of A's coins). We conclude that $B(A(\cdot))$ is also ε-DP. □

1.2 Group privacy

Finally, we might also ask what sort of guarantee DP provides for a group of individuals in the data (a family, say).

Lemma 1.7. *Let $\mathbf{x}$ and $\mathbf{x}'$ be data sets in $\mathcal{U}^n$ that differ in k positions, for an integer $1 \leq k \leq n$. If A is ε-DP, then for all events E, we have*

$$\mathbb{P}\left(A(\mathbf{x}) \in E\right) \leq e^{k\varepsilon}\mathbb{P}\left(A(\mathbf{x}') \in E\right). \tag{3}$$

Proof. Let $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(k)}$ be a sequence of $k+1$ data sets in $\mathcal{U}^n$ that move smoothly from $\mathbf{x}$ to $\mathbf{x}'$: that is, suppose $\mathbf{x}^{(0)} = \mathbf{x}$, $\mathbf{x}^{(k)} = \mathbf{x}'$ and every adjacent pair $\mathbf{x}^{(i-1)}, \mathbf{x}^{(i)}$ differ in just one entry. Consider an subset E of putputs. Moving from $\mathbf{x}^{(i-1)}$ to $\mathbf{x}^{(i)}$ increases the probability of E by at most e^{ε}. Since there are k steps in the sequence, the probability of E can increase by at most $e^{\varepsilon k}$. □

Group privacy allows us to point out an important point about DP: the parameter ε can be small, but it can never be *very* small while allowing useful information to be released. Specifically, if ε is much less than $1/n$ then for *every* two datasets $\mathbf{x}$ and $\tilde{\mathbf{x}}$, regardless of the number of entries in which they differ, the distributions of $A(\mathbf{x})$ and $A(\tilde{\mathbf{x}})$ are about the same. That means the algorithms more or less ignores its input, and cannot release information that would allow one to tell apart 0^n from 1^n, or any other pairs of data sets. We encapsulate this as the following lesson:

Useful differentially private algorithms require $\varepsilon \gg 1/n$.

In particular, it's hard for differentially private algorithms to provide useful outputs when the input data sets are small, unless we make ε quite large, perhaps even much larger than 1. "Aggregate" information requires a big enough crowd over which to aggregate.

2 Example: k-means Clustering via Lloyd's Algorithm

Let's use some of the tools we now have—the Laplace mechanism and basic composition—to design a more complex algorithm.

Lloyd's algorithm for clustering data in Euclidean space, sometimes called the "k-means" aglorithm, takes a data set of points $x_1, \ldots, x_n$ in $\mathbb{R}^d$ and attempts to find k points—not necessarily in the data—that

Algorithm 1: Lloyd's algorithm with random initialization

Input: Data set $\mathbf{x} \in \mathcal{U}^n$ where $\mathcal{U} = \left\{x \in \mathbb{R}^d : \|x\|_1 = 1\right\}$, parameter k

Initialize $c_1^{(0)}, c_2^{(0)}, ..., c_k^{(k)}$ randomly in $\mathcal{U}$;

for $t = 1$ *to* T **do**

  for $j = 1$ *to* k **do**

    $S_j = \left\{i : c_j^{(t-1)} \text{ is the closest current center to } x_i\right\}$;

    $c_j^{(t)} = \dfrac{1}{|S_j|} \displaystyle\sum_{i \in S_j} x_i$;

return $c_1^{(T)}, c_2^{(T)}, ..., c_k^{(T)}$;

minimize k-means objective, which is the sum over $i = 1, ..., n$ of the squared distance from x_i to the nearest center. When $k = 1$, the best center is always the mean of the data set. However, when $k \geq 2$, solving the problem exactly is difficult. There are algorithms with well-understood approximation guarantees for the k-means problem, but in practice people often use Lloyd's algorithm, described in Algorithm 1, since it is simple to implement and runs quickly.

The idea of Lloyd's algorithm is simple: at each stage t of the algorithm we have a candidate set of centers $c_1, ..., c_k$. We divide the data points into k groups, assigning each point x_i to group j if c_j is the closest to x_i among the clusters. We now compute new centers by taking the average of each group. This process is repeated T times for a parameter T that we will assume is given. The final output is the set of centers from the last step.

We won't try here to analyze Lloyd's algorithm (or consider any of the many known variations that improve its performance). Instead, we'll ask: Can we give a differentially private algorithm that produces similar output?

At first glance, the Laplace mechanism seems hard to apply. The final clusters produced by the algorithm come from a complex sequence of interactions between data points. Viewed as one big deterministic function, Lloyd's algorithm surely has high global sensitivity!

Instead, we'll get a differentially private version by applying the Laplace mechanism to get noisy versions of the each of the algorithm's intermediate steps. We can divide our privacy budget ε into T parts, and assign ε/T to each intermediate step. We'll analyze privacy as if the cluster centers from each intermediate step were released. We can then apply Basic Composition to get a privacy guarantee for the whole algorithm. The privacy guarantee applies even if we only release the final cluster centers, by postprocessing. In practice, it works even better to output the average of the clusters in the last few iterations of the algorithm.

The resulting algorithm is in Algorithm 2, with the differences from the original marked in red. Let's look at one iteration of the algorithm through the main for loop. Suppose we have already released centers $c_1^{(t-1)}, ..., c_k^{(t-1)}$ from the previous step. Then we can divide the universe $\mathcal{U}$ into k regions $B_1, ..., B_k$, where B_j consists of points closest to center $c_j^{(t-1)}$. (Such regions are called the "Voronoi cells" of the centers.) To compute the next set of centers, we can approximate two quantities for each B_j:

- n_j (integer): the number of data records in B_j, and
- a_j (vector in $\mathbb{R}^d$): the sum of the data records in B_j (as vectors)

In the original algorithm, the new j-th cluster center $c_j^{(t)}$ would be the average a_j/n_j. Instead, we'll use the Laplace mechanism to approximate each of these. To keep things simple, we'll assume that the data points have bounded ℓ_1 norm to begin with. That is, the universe of allowed records is

Algorithm 2: A differentially private version of Lloyd's algorithm

Input: Data set $\mathbf{x} \in \mathcal{U}^n$ where $\mathcal{U} = \left\{x \in \mathbb{R}^d : \|x\|_1 = 1\right\}$, parameter k and privacy parameter $\varepsilon > 0$

$\varepsilon' = \frac{\varepsilon}{2T}$ (since we will compose $2T$ executions of the Laplace mechanism);

Initialize $c_1^{(0)}, c_2^{(0)}, ..., c_k^{(k)}$ randomly in $\mathcal{U}$;

for $t = 1$ *to* T **do**

  for $j = 1$ *to* k **do**

    $S_j = \left\{i : c_j^{(t-1)} \text{ is the closest current center to } x_i\right\}$;

    $n_j = |S_j|$ (this has global sensitivity 2 across all j);

    $a_j = \sum_{i \in S_j} x_i$ (this has global sensitivity 2 across all j);

    Release $\hat{n}_j = n_j + Y$ where $Y \sim \text{Lap}\left(\frac{2}{\varepsilon'}\right)$;

    Release $\hat{a}_j = a_j + (Z_1, ..., Z_d)$ where $Z_\ell \sim \text{Lap}\left(\frac{2}{\varepsilon'}\right)$ i.i.d.;

    $$c_j^{(t)} = \begin{cases} \frac{\hat{a}_j}{\hat{n}_j} & \text{if } \hat{n}_j \geq 1 \\ \text{uniform in } \mathcal{U} & \text{if } \hat{n}_j < 1 \end{cases};$$

return $c_1^{(T)}, c_2^{(T)}, ..., c_k^{(T)}$ (*NB:* One can also average over the last few iterations to reduce variance);

$\mathcal{U} = \left\{x \in \mathbb{R}^d : \|x\|_1 \leq 1\right\}$.

Proposition 2.1. *Algorithm 2 is ε-DP.*

Proof. By Basic Composition, it is enough to argue that each stage t of the algorithm is $\frac{\varepsilon}{T}$-DP for every value of the previous centers centers $c_1^{(t-1)}, ..., c_k^{(t-1)}$. Fix $t \in [T]$. Fix the previous centers and the associated regions $B_1, ..., B_k$ in $\mathcal{U}$. FInally, fix two neighboring data sets $\mathbf{x}, \mathbf{x}'$.

The counts $(n_1, ..., n_k)$ form a histogram. Their global sensitivity is thus 2. Releasing $\hat{n}_1, ..., \hat{n}_k$ by adding noise $\text{Lap}\left(\frac{4T}{\varepsilon}\right)$ to each histogram entry thus consumes at most $\frac{\varepsilon}{2T}$ of our privacy budget.

Similarly, we can view the sums $a_1, ..., a_k$ as one long vector of length kd. If we change one record in the data set, only two of the sums a_j can change, since the record either stays in the same bin or moves from one bin to another. These two sums gain or lose one term each, of ℓ_1 norm at most 1. The change in the long vector is thus at most 2. Again, the algorithm adds noise $\text{Lap}\left(\frac{4T}{\varepsilon}\right)$ to each entry, consuming another $\frac{\varepsilon}{2T}$ of our privacy budget. The computation of the next cluster center is just postprocessing of the $\hat{n}_j$'s and $\hat{a}_j$'s, so it consumes no further budget.

The total expenditure for the t step is thus $\frac{\varepsilon}{T}$. By Basic Composition, the algorithm as a whole is ε-DP. □

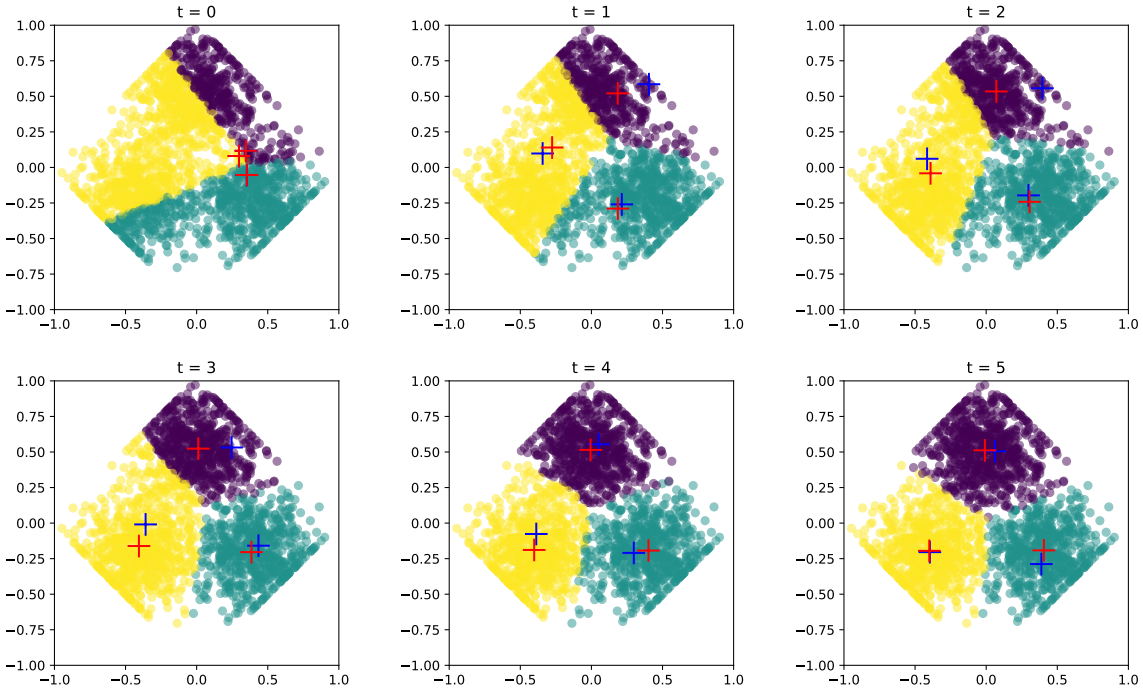

Figure 2: Results of one execution of Algorithm 2, a DP variant of LLoyd's algorithm, on a data set of $n = 2,000$ points with $k = 3$, $T = 6$ and $\varepsilon = 0.5$. The value t (from 0 to 5) is the iteration number. The blue crosses represent the cluster centers from each stage of the original algorithm while the red ones represent the cluster centers from each stage of the DP variant. The red crosses are fairly stable after the first two iterations; the blue ones converge to essentially the same locations, but take longer to stabilize.

3 Interpreting DP: Smoking, Cancer, and Correlations

What does it mean to decide if a concept like differential privacy is a good definition of "privacy"? There is no single answer, since it involves a connection between an unambiguous mathematical concept and a nebulous social one. "Privacy" covers lots of different concepts, many of which are more about control than confidentiality, and all of which are context-dependent.[1] Nevertheless, we can try to wrap our heads around the guarantee that a technical concept provides—perhaps we can chip off a piece of "privacy" which is accurately pinned down by DP.

How can we start? A good exercise is to write down an natural-language sentence that captures the type of guarantee we would like. A strong requirement, reminiscent of what is possible for encryption would be this:

> **A first attempt:** No matter what they know ahead of time, the attacker's beliefs about Alice are the same after they see the output as they were before.

Unfortunately, such a strong guarantee is impossible to achieve if we actually want to release useful information. To see why, consider the example of a clinical study that explores the relationship between smoking and lung disease. A health insurance company with no *a priori* understanding of that relationship might, after seeing the results of the study, dramatically alter its estimates of different people's likelihood of disease. In turn, this would likely cause the company to raise premiums for smokers and lower them for nonsmokers. The conclusions drawn by the company about the riskiness of any one individual (say Alice) are strongly affected by the results of the study. Their beliefs about Alice have definitely changed.

However, the change can hardly be called a breach of Alice's privacy. It happens because the study reveals a feature of human biology—exactly what we want clinical studies to do!

So what can we hope to achieve? One important observation about the smoking and lung disease example is that the information about Alice would be learned by the insurance company regardless of whether Alice participated in the study. In other words, the conclusions the insurance company draws about Alice come from the totality of the data set, and don't depend strongly on her data. One way to understand differential privacy is that this is the only kind of inference about individuals that it allows.

> **The DP principle:** No matter what they know ahead of time, an attacker seeing the output of a differentially private algorithm would draw (almost) the same conclusions about Alice *whether or not her data were used.*

It is instructive to formalize this intuitive statement. What do we mean by "what the attacker knows" and "what they learn"? We'll adopt what statisticians call a Bayesian perspective, and encode knowledge via probability distributions. Specifically, let's think of the data set as a random variable $\mathbf{X}$ distributed over $\mathcal{U}^n$. For clarity, we'll use capital letters like $\mathbf{X}$ to refer to random variables, and lower case symbols like $\mathbf{x}$ to refer to specific realizations.

We can the adversary's background knowledge via a *prior distribution* $p(\mathbf{x}) = \mathbb{P}(\mathbf{X} = \mathbf{x})$. We should think of this as how likely a given data set is to occur given everything the attacker knows ahead of time. [2] Because we don't know what other information the attacker has, we will want our analysis to work for *every* prior distribution p.

[1] A number of writers have dissected the concept, trying to provide their own taxonomy of privacy's many facets. Brandeis [?], Solove [?], and Nissenbaum [?] provide good places to start.

[2] Our use of probability to model knowledge this way corresponds to the *subjective interpretation* of probability (see, e.g., [Háj19]). It's pretty different from the way we use probability in the definition of a randomized algorithm, or in the definition

Given the output $a = A(\mathbf{X})$. We can model "what the adversary learns" by the *posterior distribution* of the data conditioned on the algorithm's output. That is,

$$p(\mathbf{x} \mid a) \stackrel{\text{def}}{=} \mathbb{P}\left(\mathbf{X} = \mathbf{x} \mid A(\mathbf{X}) = a\right) = \frac{\mathbb{P}\left(A(\mathbf{x}) = a\right) \cdot p(\mathbf{x})}{\sum_{\tilde{\mathbf{x}} \in \mathcal{U}^n} \mathbb{P}\left(A(\tilde{\mathbf{x}}) = a\right) \cdot p(\tilde{\mathbf{x}})} \,. \tag{4}$$

But how should we model "what the attacker would have learned had person i's data been removed"? Given a data set $\mathbf{x} \in \mathcal{U}^n$, let $\mathbf{x}_{-i}$ denote the data set in which person i's entry has been replaced by a default value. Consider a hypothetical world in which the data set $\mathbf{x}_{-i}$ is used instead of the real data set $\mathbf{x}$. Given an output a, we can now consider the conditional distribution $p_{-i}(\cdot \mid a)$ that the attacker would have constructed in the hypothetical world, namely:

$$p_{-i}(\mathbf{x} \mid a) \stackrel{\text{def}}{=} \mathbb{P}\left(\mathbf{X} = \mathbf{x} \mid A(\mathbf{X}_{-i}) = a\right) = \frac{\mathbb{P}\left(A(\mathbf{x}_{-i}) = a\right) \cdot p(\mathbf{x})}{\sum_{\tilde{\mathbf{x}} \in \mathcal{U}^n} \mathbb{P}\left(A(\tilde{\mathbf{x}}_{-i}) = a\right) \cdot p(\tilde{\mathbf{x}})} \,. \tag{5}$$

We can think of $p_{-i}(\cdot \mid a)$ as encoding what the attacker would have learned about person i had person i's data never been used.

To formalize our claim about differential privacy, we'll use the following shorthand. For two distributions p and q on the same set $\mathcal{Y}$ (technically, over the same σ-algebra of events), we'll write

$$p \approx_\varepsilon q \qquad \Leftrightarrow \qquad (\forall \text{ events } E \subseteq \mathcal{Y} \;:\; p(E) \le e^\varepsilon q(E) \text{ and } q(E) \le e^\varepsilon p(E)) \,. \tag{6}$$

Given two random variables A and B distributed over the same set, we'll sometimes abuse notation and write $A \approx_\varepsilon B$ to mean that the relation in (6) is satisfied by their distributions. With this notation, an algorithm A is ε-DP if and only if, for every pair of neighboring data sets $\mathbf{x}$ and $\mathbf{x}'$, we have $A(\mathbf{x}) \approx_\varepsilon A(\mathbf{x}')$.

Theorem 3.1. *Let $A : \mathcal{U}^n \to \mathcal{Y}$ be ε-differentially private. For every distribution on $\mathbf{X}$ (possibly with dependencies among the entries), for every output $a \in \mathcal{Y}$, for every index i, we have*

$$p_{-i}(\cdot \mid a) \approx_{2\varepsilon} p(\cdot \mid a) \,. \tag{7}$$

Exercise 3.2. Prove Theorem 3.1. *Hint:* Fix an output $a \in \mathcal{Y}$. Given a data set $\mathbf{x} \in \mathcal{U}^n$, how can you write the ratio $\frac{p_{-i}(\mathbf{x}|a)}{p(\mathbf{x}|a)}$ in terms of the ratios of the form $\frac{\mathbb{P}(A(\mathbf{x}_{-i})=a)}{\mathbb{P}(A(\mathbf{x})=a)}$?

Something here might seem weird: how can the attacker learn about i's data from a if x_i was not used to compute a? The answer is in the dependencies among the data records—the attacker can learn about $\mathbf{x}_{-i}$, which itself reveals information about x_i.

Returning to the smoking and lung disease example: Suppose the records in $\mathbf{X}$ are drawn i.i.d. from one of several possible distributions. For simplicity, imagine there are two possible distributions, one where the features are independent, and one where they are strongly correlated, so that the prior on $\mathbf{X}$ is a mixture of the two. Seeing the clinical study's results basically causes the insurance company's posterior to collapse to the i.i.d. distribution in which the features are correlated. Whether the study used Alice's data or not, the insurance company's posterior distribution on Alice's record would have the features correlated.

What have we learned? We can model knowledge via probabilities, and learning via the change from prior to posterior distributions. When we do that, we can make our intuition precise—-that differentially private mechanisms reveal only information that could be learned without any particular person's data.

of differential privacy. In those contexts, the probabilities reflect a process we control, and it's reasonable to think of them as known exactly. In contrast, we cannot expect to know an attacker's prior. Here, we posit only that it exists. Even this postulate is delicate, especially since real attackers are computationally bounded. We ignore computational restrictions here for simplicity.

We've also found a useful natural language formulation of our goal when thinking about confidentiality of individuals' data when releasing aggregate statistics. That type of formulation is particularly useful since it can guide our intuition for the technical concepts. It can also help us articulate goals in legal and policy discussions.

3.1 A not-so-great variation on differential privacy

The formulation of Theorem 3.1 also helps us distinguish among seemingly similar definitions of privacy.

Suppose we were to require that probabilities differ by an additve error term rather than a multiplicative one. We might say that a randomized algorithm satisfies "δ-additive secrecy" if

$$\forall \mathbf{x}, \mathbf{x}' \in \mathcal{U}^n \text{ neighbors }, \ \forall \text{ events } E \ : \ \mathbb{P}\left(A(\mathbf{x}) \in E\right) \leq \mathbb{P}\left(A(\mathbf{x}') \in E\right) + \delta \, . \tag{8}$$

How different is this from differential privacy? It certainly has things in common: for example, it is closed under composition and postprocessing, and satisfies a similar version of group privacy. In particular, we must have $\delta > 1/n$ to get useful information out of such an algorithm. However, it does *not* satisfy a reasonable analogue to Theorem 3.1, and it *does* allow some algorithms that are pretty obviously disclosive.

Exercise 3.3 (Name and Shame Mechanism)**.** Consider the following mechanism NS_δ. On input $\mathbf{x} = (x_1, ..., x_n)$, for each i from 1 to n, it generates

$$Y_i = \begin{cases} (i, x_i) & \text{w. prob. } \delta \, , \\ \perp & \text{w. prob. } 1 - \delta \, . \end{cases} \tag{9}$$

Here $\perp$ is just a special symbol meaning "no information".

(i) Show that NS_δ satisfies "δ-additive secrecy". (ii) Show that for $\delta \gg 1/n$, the mechanism publishes some individuals' data in the clear with high probability, and that for such outputs, Eq. (7) in Theorem 3.1 does not hold.

Summary

Key Points

- Differentially private algorithms can be assembled modularly, or run independently by different organizations. The privacy parameter accumulates at most additively across all executions that use the same person's record.
- We can view the privacy parameter as a budget to be divided among different efforts.
- For some algorithms, one gets a much better analysis by considering the steps jointly, rather than using composition. (Exercise 1.5)
- Algorithms that access their data using summation queries can often be made differentially private without too much loss of accuracy. We saw the example of Lloyd's algorithm.
- Useful statistical summaries may have to reveal information about an individual to an attacker. However, we can make a more subtle claim: No matter what they know ahead of time, an attacker seeing the output of a differentially private algorithm would draw (almost) the same conclusions about Alice *whether or not her data were used.*

Additional Reading and Watching

- More on the formulation of Theorem 3.1: [KS14]
  - MinutePhysics' Youtube video "When It's OK to Violate Privacy", 2019.
- A thorough proof of the impossibility of the "first attempt" privacy guarantee: [DN10] (see also [KM11]).
- Why noisy sums can be used to find useful approximations to many natural procedures: [Kea93, BDMN05, DMNS16].

References

[BDMN05] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: the SuLQ framework. In *Proceedings of the 24th Annual ACM Symposium on Principles of Database Systems*, PODS '05, 2005. ACM.

[DMNS16] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. *Journal of Privacy and Confidentiality*, 7(3), 2016.

[DN10] Cynthia Dwork and Moni Naor. On the difficulties of disclosure prevention, or the case for differential privacy. *Journal of Privacy and Confidentiality*, 2(1), 2010.

[Háj19] Alan Hájek. Interpretations of Probability. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, fall 2019 edition, 2019.

[Kea93] Michael J. Kearns. Efficient noise-tolerant learning from statistical queries. In *ACM Symposium on Theory of Computing*. ACM, 1993.

[KM11] Daniel Kifer and Ashwin Machanavajjhala. No Free Lunch in Data Privacy. In *SIGMOD*, 2011.

[KS14] Shiva Prasad Kasiviswanathan and Adam D. Smith. On the 'semantics' of differential privacy: A bayesian formulation. *Journal of Privacy and Confidentiality*, 2014.