CS325: Private Data Analysis **Fall 2023**

Lecture 3: Properties of Differential Privacy

Lecturer: Di Wang *Scribes: Di Wang*

Note: *LaTeX template courtesy of UC Berkeley EECS dept.*

3.1 Properties of Differential Privacy

3.1.1 Composition Theorem

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. Lets illustrate this with two scenarios:

Scenario 1: Suppose two hospitals hold overlapping data sets D_1 and D_2, and we run two DP algorithms A_1 and A_2 on each of them. For the individuals whose records are in both data sets, what sort of privacy guarantee can we make if an outside attacker see the output of both algorithms?

Scenario 2: Suppose I have one data set D, and I run an ϵ_1-DP algorithm A_1 to get an 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 $A_2^{a_1}$ which is ϵ_2-DP. For example, maybe if both counts in a_1 are at least 100, then I may 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. Finally I get an output a_2 and I will output both a_1 and a_2.

Both of these scenarios are cases of composition. To simplify things a bit, think of the two data sets in Scenario 1 as one big dataset D containing the information from both hospitals; the set of people with records in D would be the union of the sets of people in with records in D_1 and D_2. So what the adversary sees in Scenario 1 is the outcome of one big algorithm $A(D) = (A_1(D), A_2(D))$. Similarly, in Scenario 2 we can review 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. 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 ϵ_1 and ϵ_2-DP respectively. Then the whole algorithm A is $\epsilon_1 + \epsilon_2$-DP.

Lemma 3.1 *Let $\mathcal{A} : \mathcal{X}^n \mapsto \mathcal{Y}_1 \times \mathcal{Y}_2$ be a randomized algorithm that outputs $A(D) = (a_1, a_2)$, where $a_1 = \mathcal{A}_1(D)$ and $a_2 = \mathcal{A}_2(a_1, D)$. If $\mathcal{A}_1 : \mathcal{X}^n \mapsto \mathcal{Y}_1$ is ϵ_1-DP and $\mathcal{A}_2 : \mathcal{Y} \times \mathcal{X}^n \mapsto \mathcal{Y}_2$ is ϵ_2-DP for any fixed $a_1 \in \mathcal{Y}_1$. Then $\mathcal{A}$ is $\epsilon = \epsilon_1 + \epsilon_2$-DP.*

Proof: For simplicity we assume the space of $\mathcal{Y}_1 \times \mathcal{Y}_2$ is discrete. Consider neighboring datasets D and D'

and fixed $(a_1, a_2) \in \mathcal{Y}_1 \times \mathcal{Y}_2$ we have

$$\begin{aligned}
\mathbb{P}(\mathcal{A}(D) = (a_1, a_2) &= \mathbb{P}(\mathcal{A}_1(D) = a_1) \cdot \mathbb{P}(\mathcal{A}_2(a_1, D) = a_2) && (3.1)\\
&\leq e^{\epsilon_1}\mathbb{P}(\mathcal{A}_1(D') = a_1) e^{\epsilon_2}\mathbb{P}(\mathcal{A}_2(a_1, D') = a_2) && (3.2)\\
&= e^{\epsilon_1+\epsilon_2}\mathbb{P}(\mathcal{A}(D') = (a_1, a_2)). && (3.3)
\end{aligned}$$

And thus we can see for all the events E, the above inequality holds. Thus, $\mathcal{A}$ is $\epsilon = \epsilon_1 + \epsilon_2$-DP. ■

Theorem 3.2 (Composition Theorem) *Let $\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_k$ be a sequence of randomized algorithms, where $\mathcal{A}_1 : \mathcal{X}^n \mapsto \mathcal{Y}_1$, $\mathcal{A}_2 : \mathcal{Y}_1 \times \mathcal{X}^n \mapsto \mathcal{Y}_2, \cdots, \mathcal{A}_k : \mathcal{Y}_1 \times \mathcal{Y}_2 \times \mathcal{Y}_{k-1} \times \mathcal{X}^n \mapsto \mathcal{Y}_k$. Suppose for every $i \in [k]$ and $a_1 \in \mathcal{Y}_1, a_2 \in \mathcal{Y}_2, \cdots, a_k\mathcal{Y}_k$ we have $\mathcal{A}_i(a_1, \cdots, a_{i-1}, \cdot) : \mathcal{X}^n \mapsto \mathcal{Y}_i$ is ϵ_i-DP. Then the algorithm $\mathcal{A} : \mathcal{X} \mapsto \mathcal{Y}_1 \times \mathcal{Y}_2 \times \cdots \mathcal{Y}_k$ that runs the algorithms $\mathcal{A}_i$ sequentially is ϵ-DP for $\epsilon = \sum_{i=1}^k \epsilon_i$.*

3.1.2 Postprocessing

Now one question that might come up is whether its ok if we only want to release parts of A's output? For instance, what if we release only a_2? Or perhaps 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 3.3 (Closure under postprocessing) *Let $\mathcal{A} : \mathcal{X}^n \mapsto \mathcal{Y}$ be a randomized algorithm that is ϵ-differentially private. Then for $B : \mathcal{Y} \mapsto \mathcal{Z}$, $B(\mathcal{A}(\cdot))$ is ϵ-DP.*

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

$$\mathbb{P}(B(\mathcal{A}(D)) = b) = \mathbb{P}(\mathcal{A}(D) \in B^{-1}(b)) \leq e^{\epsilon}\mathbb{P}(\mathcal{A}(D') \in B^{-1}(b)) = e^{\epsilon}\mathbb{P}(B(\mathcal{A}(D')) = b). \tag{3.4}$$

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 Bs random choices. Thus, $B(\mathcal{A}(\cdot))$ is the application of a deterministic function to $\mathcal{A}'(D) = (\mathcal{A}(D), R)$. The algorithm $\mathcal{A}'$ is ϵ-DP (since R is independent of $\mathcal{A}$'s coin). Thus $B(\mathcal{A}(\cdot))$ is also ϵ-DP. ■

3.1.3 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 3.4 *Let D and D' be datasets in $\mathcal{X}^n$ that differ in k positions, for an integer $1 \leq k \leq n$. If $\mathcal{A}$ is ϵ-DP, then for all events E we have*

$$\mathbb{P}(\mathcal{A}(D) \in E) \leq e^{k\epsilon}\mathbb{P}(\mathcal{A}(D') \in E). \tag{3.5}$$

Proof: Let $D^{(0)}, D^{(1)}, \cdots, D^{(k)}$ be a sequence of $k+1$ datasets in $\mathcal{X}^n$ that move smoothly from D to D': that is, suppose $D^{(0)} = D, D^{(k)} = D'$ and every adjacent pair $D^{(i-1)}$ and $D^{(i)}$ differ in just one entry. Consider an subset E of outputs. Moving from $D^{(i-1)}$ to $D^{(i)}$ 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 $e^{\epsilon k}$. ■

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 D and $\tilde{D}$, regardless of the number of entries in which they differ, the distributions of $A(D)$ and $A(\tilde{D})$ will be 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** $\epsilon \gg \frac{1}{n}$. In particular, its 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.

Note that the above statement will not be true when the data follows some underlying distribution. Specifically, recently [2] studied the replicability of learning (they call it as reproducibility. However iIthink it should be better to call them replicability). In general, it informally says that a randomized algorithm is reproducible if two distinct runs of the algorithm on two sets of samples drawn from the same distribution, with internal randomness fixed between both runs, produces the same output with high probability.

Definition 3.5 *Let $\mathcal{P}$ be a distribution over a universe $\mathcal{X}$, let $\mathcal{A}$ be a randomized algorithm with sample access to $\mathcal{P}$. We call $\mathcal{A}$ is ρ-replicable if*

$$\mathbb{P}_{D_1,D_2,r}(\mathcal{A}(D_1) = \mathcal{A}(D_2)) \geq 1 - \rho,$$

where D_1 and D_2 denote sequences of samples drawn i.i.d. from $\mathcal{P}$, and r denotes a random binary string representing the internal randomness used by $\mathcal{A}$.

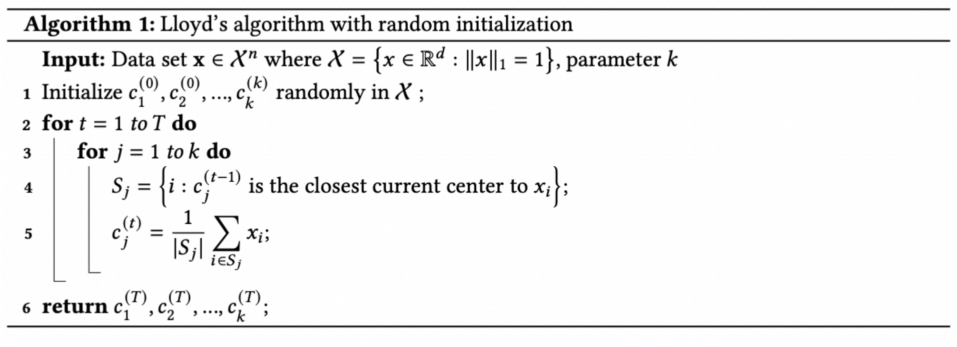
Replicable algorithms guarantee a different form of privacy: If A is replicable, then what A learns (for example, a trained classifier) is almost always the same; thus, A is usually independent of the chosen training data. In this way, replicable algorithms are prevented from memorizing anything that is specific to the training data, similar to differentially private algorithms. Replicability is weaker than differential privacy in the sense that reproducibility only applies to in-distribution samples, whereas differential privacy applies to any training set. On the other hand, reproducibility is stronger in the sense that its guarantee for in-distribution samples is global rather than local (for neighboring samples).

3.1.4 Application of DP Properties

Let's use some of the tools we now havethe Laplace mechanism and basic compositionto design a more complex algorithm.

Lloyd's algorithm for clustering data in Euclidean space, sometimes called the k-means algorithm, takes a data set of points $x_1, \cdots, x_n$ in $\mathbb{R}^d$ and attempts to find k centers that minimize k-means objective. which is the sum over $i = 1, 2 \cdots, n$ of the squared distance from x_i to the nearest center. There are algorithms with well-understood approximation guarantees for the k-means problem, but in practice people often use Lloyds algorithm, described in Figure 3.1.4. The idea of Lloyds algorithm is simple: at each stage t of the algorithm we have a candidate set $c_1, \cdots, c_k$. We divide the data points into k groups, assigning each point x to the j-th group if c_j is the closest center to x. 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 can get a differentially private version of the previous algorithm 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 $\frac{\epsilon}{T}$ to each intermediate step. Well analyze privacy as if the cluster centers from each intermediate step were released. We can then apply Basic Composition to get a privacy guarantee



```
Algorithm 1: Lloyd's algorithm with random initialization
  Input: Data set $\mathbf{x} \in \mathcal{X}^n$ where $\mathcal{X} = \{x \in \mathbb{R}^d : \|x\|_1 = 1\}$, parameter $k$
1 Initialize $c_1^{(0)}, c_2^{(0)}, \ldots, c_k^{(k)}$ randomly in $\mathcal{X}$ ;
2 for $t = 1$ to $T$ do
3     for $j = 1$ to $k$ do
4         $S_j = \left\{ i : c_j^{(t-1)} \text{ is the closest current center to } x_i \right\}$;
5         $c_j^{(t)} = \frac{1}{|S_j|} \sum_{i \in S_j} x_i$;
6 return $c_1^{(T)}, c_2^{(T)}, \ldots, c_k^{(T)}$;
```

Figure 3.1: Lloyd's algorithm

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 Figure 3.1.4, with the difference from the original marked in red. Lets look at one iteration of the algorithm through the main for loop. Suppose we have already released centers $c_1^{t-1}, \cdots, c_k^{t-1}$ from the previous step. Then we can divide the dataset into k subsets $B_1, \cdots, B_k$ where B_j consists of points closest to center c_j^{t-1}. 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 $\mathcal{R}^d$): 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, well use the Laplace mechanism to approximate each of these. To keep things simple, well assume that the data points have bounded ℓ_1 norm to begin with.

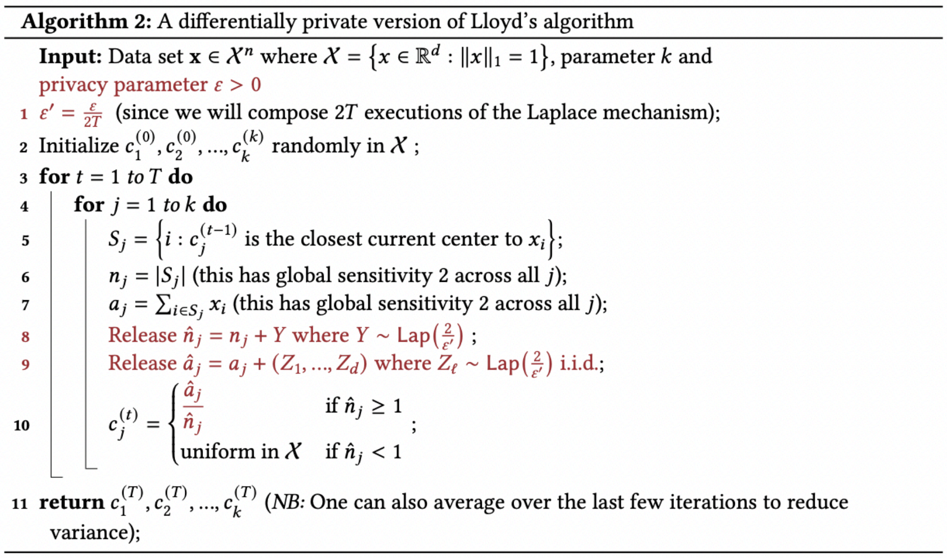
Theorem 3.6 *The above private Lolyd Algorithm is ϵ-DP.*

Proof: By Basic Composition, it is enough to argue that each stage t of the algorithm is ϵ/T-DP with fixed $c_1^{t-1}, \cdots, c_k^{t-1}$. Fix the previous centers and the associated subsets $B_1, \cdots, B_k$ in D. Finally, consider two neighboring data sets D, D'. We know the counts $(n_1, \cdots, n_k)$ form a histogram. Thus its sensitivity is 2. Thus, releasing $(\hat{n}_1, \cdots, \hat{n}_k) = (n_1, \cdots, n_k) + \text{Lap}(\frac{4T}{\epsilon}$ will be $\epsilon 2T$-DP.

Similarly, we can view the sum $a_1, \cdots, a_k$ as a long vector with length kd. If we change one record in the data set, only two of the sum a_j can be changed, 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 $\text{Lap}(\frac{4T}{\epsilon})$ to each entry, consuming another $\frac{\epsilon}{2T}$ budget. Thus, by the basic composition theorem, the whole algorithm is ϵ-DP. ■

3.2 Interpreting DP

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



```
Algorithm 2: A differentially private version of Lloyd's algorithm
Input: Data set $\mathbf{x} \in \mathcal{X}^n$ where $\mathcal{X} = \{x \in \mathbb{R}^d : \|x\|_1 = 1\}$, 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 $\mathcal{X}$ ;
for $t = 1$ to $T$ do
    for $j = 1$ to $k$ do
        $S_j = \{i : c_j^{(t-1)}$ is the closest current center to $x_i\}$;
        $n_j = |S_j|$ (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 $Y \sim \mathrm{Lap}(\frac{2}{\varepsilon'})$ ;
        Release $\hat{a}_j = a_j + (Z_1, ..., Z_d)$ where $Z_\ell \sim \mathrm{Lap}(\frac{2}{\varepsilon'})$ i.i.d.;
        $c_j^{(t)} = \begin{cases} \frac{\hat{a}_j}{\hat{n}_j} & \text{if } \hat{n}_j \geq 1 \\ \text{uniform in } \mathcal{X} & \text{if } \hat{n}_j < 1 \end{cases}$ ;
return $c_1^{(T)}, c_2^{(T)}, ..., c_k^{(T)}$ (NB: One can also average over the last few iterations to reduce
    variance);
```

Figure 3.2: Lloyd's algorithm

Figure 3.3: Private Lolyd Algorithm

nebulous social one. Privacy covers lots of different concepts, many of which are more about control than confidentially, and all of which are context-dependent. Nevertheless, we can try to wrap our heads around the guarantee that a technical concept provides-perhaps we can chip on 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 fist attempt: No matter what they know ahead of time, the attackers 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 peoples 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 Alices privacy. It happens because the study reveals a feature of human biologyexactly 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 dont 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? Well adopt what statisticians call a Bayesian perspective, and encode knowledge via probability distributions. Specifically, lets think of the data set as a random variable D distributed over $\mathcal{X}^n$. For clarity, well use capital letters like D to refer to random variables, and lower case symbols like d to refer to specific realizations

Consider there is an attacker, we can model the adversary's background knowledge via a prior distribution $p(d) = \mathbb{P}(D = d)$. We should think of this as how likely a given data set is to occur given everything the attacker knows ahead of time. Because we dont know what other information the attacker has, we will want our analysis to work for every prior distribution p.

Given the output $a = \mathcal{A}(D)$, we can model "what the adversary learns" by the posterior distribution of the data conditioned on the algorithms output. That is,

$$p(d|a) = \mathbb{P}(D = d|\mathcal{A}(D) = a) = \frac{\mathbb{P}(\mathcal{A}(d) = a)p(d)}{\sum_{d'} \mathbb{P}(\mathcal{A}(d') = a)p(d')}. \tag{3.6}$$

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

$$p_{a,-i}(d|a) = \mathbb{P}(D = d|\mathcal{A}(D_{-i}) = a) = \frac{\mathbb{P}(\mathcal{A}(d_{-i}) = a)p(d)}{\sum_{d'} \mathbb{P}(\mathcal{A}(d'_{-i}) = a)p(d')}. \tag{3.7}$$

To formalize our claim about differential privacy, well use the following shorthand. For two distributions P and Q on the same set $\mathcal{Y}$, well write $P \approx_\epsilon Q$ as for all events $E \subset \mathcal{Y}$, $P(E) \leq e^\epsilon Q(E)$ and $Q(E) \leq e^\epsilon P(E)$.

Theorem 3.7 *Let $\mathcal{A} : \mathcal{X}^n \mapsto \mathcal{Y}$ be ϵ-DP. For every distribution on X (possibly with dependencies among the entries), for every output $a \in \mathcal{Y}$, and every index i we have*

$$p_{a,-i}(\cdot|a) \approx_{2\epsilon} p(\cdot|a). \tag{3.8}$$

Something here might seem weird: how can the attacker learn about i's data from a if d_i was not used to compute a? The answer is in the dependencies among the data records-the attacker can learn about d_{-i} , which itself reveals information about d_i. Returning to the smoking and lung disease example: Suppose the records in D 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 D is a mixture of the two. Seeing the clinical studys results basically causes the insurance companys posterior to collapse to the i.i.d. distribution in which the features are correlated. Whether the study used Alices data or not, the insurance companys posterior distribution on Alices 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 persons data.

Weve also found a useful natural language formulation of our goal when thinking about confidentially 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.

3.3 Other Definitions of Privacy

Suppose we were to require that probabilities differ by an additive error term rather than a multiplicative one. We might say that a randomized algorithm satises δ-additive privacy if for all pairs of neighboring data sets D, D' and for all sets of events E, we have

$$\mathbb{P}(A(D) \in E) \leq \mathbb{P}(A(D') \in E) + \delta.$$

How different is this from differential privacy? It certainly has things in common: for example, it is closed under composition and post-processing, and satises a similar version of group privacy. In particular, we must have $\delta > \frac{1}{n}$ to get useful information out of such an algorithm. However, it does not satisfy a reasonable analogue to Theorem 3.6, and it does allow some algorithms that are pretty obviously disclosive.

We consider the following 'name and shame mechanism'. On input $D = (d_1, \cdots, d_n)$ for each i it generates (d_i, i) with probability δ and no information otherwise. We can see it satisfies the δ-additive privacy. However, we can also see the probability that the mechanism will release at least one (d_i, i) with some $i \in [n]$ is $1-(1-\delta)^n \approx \delta n$. Since we know $\delta \gg \frac{1}{n}$. That is with high probability the mechanism will release personal data and thus it is non-private. And we can see Theorem 3.6 does not hold.

References

[1] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 202–210, 2003.

[2] Russell Impagliazzo, Rex Lei, Toniann Pitassi, and Jessica Sorrell. Reproducibility in learning. *arXiv preprint arXiv:2201.08430*, 2022.