Chapter 5 — Properties of DP

Draft version: September 22, 2021

Contents

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 each runs a differentially private algorithm on its data set. Suppose the hospitals run separate algorithms A_1 and A_2, each of which is differentially private. 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.

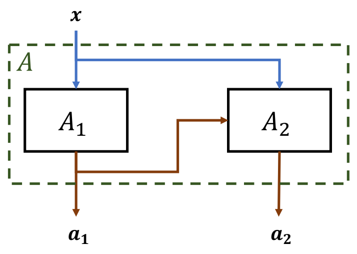

Figure 1: The adaptive composition of two algorithms (solid boxes), viewed as a single larger algorithm A (dashed box).

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 $A : \mathcal{U}^n \to \mathcal{Y}_1 \times \mathcal{Y}_2$ 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 $\mathcal{Y}_1 \times \mathcal{Y}_2$.

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

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.

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

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.

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 $A_1, A_2, \ldots, A_k$ 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 $i = 2, 3, \ldots, k$ (so A_i takes as input elements that could have been output by $A_1, \ldots, A_{i-1}$, as well as a data set in $\mathcal{U}^n$). Suppose that for each $i \in [k]$, for every $a_1 \in \mathcal{Y}_1, a_2 \in \mathcal{Y}_2, \ldots, a_{i-1} \in \mathcal{Y}_{i-1}$, we have that $A_i(a_1, \ldots, a_{i-1}, \cdot)$ 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 $B_1, B_2, \ldots, B_d$, 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 $n_1, n_2, \ldots, n_d$, 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 $B^{-1}(b)$:

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

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). Thus $B(A(\cdot))$ 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 $1 \le k \le n$. If A is ε-DP, then for all events E, we have*

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

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:

$$\boxed{\text{Useful differentially private algorithms require } \varepsilon \gg 1/n.}$$

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.

Additional Reading and Watching

- 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.

[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.