CS839: Differential Privacy and Learning March 17, 2026

Lecture 17: Introduction to DP Statistics

Instructor: Gavin Brown *Scribe:* Divyam Anshumaan

Disclaimer: This document is intended as an informal supplement to in-class note-taking. It has not been given the level of scrutiny expected in polished lecture notes, let alone that reserved for peer-reviewed publications.

Summary. This lecture introduced the topic of statistical inference under differential privacy. We discussed the types of statistical goals one might pursue (for example, parameter estimation, distribution learning, hypothesis testing, prediction) and the key techniques for achieving them privately. We then went over *Subsample and Aggregate* [Nissim et al., 2007], which converts an arbitrary (non-private) algorithm into a differentially private one by partitioning the data and privately aggregating the results. Finally, we introduced the *FriendlyCore* framework of Tsfadia et al. [2022], a general-purpose approach to differentially private aggregation that filters data before aggregation.

1 Statistical Inference under Differential Privacy

As always, privacy is a worst-case requirement. However, to analyse the accuracy or utility of a private algorithm, we typically need distributional or boundedness assumptions on the data. For example, we might assume $x_i \overset{\text{iid}}{\sim} \mathcal{N}(\mu, \Sigma)$, or that $\|x_i\|_2 \leq R$ for all i. As we will see later in the class, assumptions of this sort are necessary for private statistical inference.

Types of goals. Under such assumptions, there are several statistical goals one might pursue:

1. **Parameter estimation:** e.g., find an estimate $\tilde{\mu}$ such that $\|\mu - \tilde{\mu}\|_2 \leq \alpha$.
2. **Distribution learning:** e.g., find an estimate $\tilde{P}$ of the data-generating distribution P such that $\mathrm{TV}(P, \tilde{P}) \leq \alpha$.
3. **Hypothesis testing:** e.g., given data x, determine whether $x \sim P$ or $x \sim Q$.
4. **Prediction:** e.g., as in regression, find a predictor $\tilde{\theta}$ such that $\mathbb{E}[(\langle x, \tilde{\theta}\rangle - y)^2] \leq \alpha$.

Important techniques. We highlighted the following important techniques for achieving these goals under differential privacy:

1. Bound the global sensitivity of the quantity of interest and add noise accordingly.
2. Use DP optimization (e.g., DP-SGD).
3. Subsample and aggregate.
4. Robustness-to-privacy transformations (inverse sensitivity mechanism).
5. "Stable" data-dependent noise (not a real name, just a collection of techniques)

The remainder of this lecture focused on "Subsample and Aggregate" and a related framework called the "Friendly Core".

2 Subsample and Aggregate

Suppose we have a non-private algorithm $A : \mathcal{X}^* \to \mathcal{Y}$ with no good bound on its global sensitivity. That is, changing a single data point could dramatically alter the output. *Subsample and aggregate* provides a generic way to make such algorithms private.

The idea. Partition the dataset $x = (x_1, \ldots, x_n)$ into k disjoint buckets $x^{(1)}, \ldots, x^{(k)}$, each of size $m = n/k$. Run the base algorithm independently on each bucket to obtain outputs $y_j = A(x^{(j)})$ for $j = 1, \ldots, k$. Then privately aggregate the outputs $y_1, \ldots, y_k$ to produce a final answer $\tilde{y}$.

$$\begin{array}{lll} \boxed{x^{(1)}} & \xrightarrow{A} & y_1 \\ \boxed{x^{(2)}} & \xrightarrow{A} & y_2 \\ \vdots & & \vdots \\ \boxed{x^{(k)}} & \xrightarrow{A} & y_k \end{array} \longrightarrow \boxed{\text{Aggregator}} \longrightarrow \tilde{y}$$

If the aggregation step $\mathrm{Agg}(y_1, \ldots, y_k)$ is (ε, δ)-differentially private (viewing each y_j as a "data point"), then the entire procedure is (ε, δ)-DP. This is because each individual x_i appears in exactly one bucket, so changing a single data point changes exactly one of the y_j's, and the aggregator provides privacy with respect to changes in a single input.

Why this helps. Even though the base algorithm A may have high global sensitivity, the outputs $y_1, \ldots, y_k$ may be very well-behaved. For example, if the data truly comes from some distribution P, then by a central limit theorem argument, we might expect $y_j \sim \mathcal{N}\big(y^*, \frac{1}{m} I\big)$ for large enough bucket size m. More generally, we can use information about the expected behavior of the estimator on typical data to design our aggregation step.

What we lose There is typically some loss in accuracy due to the data splitting. In many cases subsample-and-aggregate is a strong and simple baseline, but not statistically optimal.

Example: Hypothesis testing. Consider the hypothesis testing problem:

$$H_0 : x_1, \ldots, x_n \overset{\text{iid}}{\sim} P \qquad \text{vs.} \qquad H_1 : x_1, \ldots, x_n \overset{\text{iid}}{\sim} Q.$$

A non-private algorithm A can be run on each bucket to produce binary outputs $y_1, \ldots, y_k \in \{0, 1\}$ (e.g., 0 for "accept H_0" and 1 for "reject H_0").

In this class, we've seen several tools that allow us to aggregate these answers.

- **Histograms**
- **Laplace mechanism:** Compute $\#\{y_i = 0\}$ and add Laplace noise calibrated to a global sensitivity of 1.
- **Exponential mechanism:** Sample the output $o \in \{0, 1\}$ with probability proportional to $\exp\big(\frac{\varepsilon}{2} \cdot \#\{y_i = o\}\big)$.

Any of these will give a good private answer once the non-private algorithm, on inputs of size m, generates the correct answer with high constant probability.

Non-binary outputs. If the y_i's are not binary, other aggregation strategies are needed:

- If $y_i \in \mathbb{R}$, one can use a DP median algorithm, analogous to the classical "median of means" estimator.
- In some settings, a DP mean may also be appropriate.
- If $y_i \in \mathbb{R}^d$, one can compute the coordinatewise mean or median of the y_i's and add Gaussian noise. This is private and computationally efficient, but will usually be suboptimal from an accuracy standpoint.

3 The Friendly Core Framework

The Friendly Core is a "general-purpose" framework[1] for differentially private aggregation. The central idea is to filter the dataset so that any non-removed points are well-behaved ("friendly"), and then apply a relaxed notion of differential privacy that only needs to hold over friendly datasets. This encodes our heuristic of wanting to reason only about typical data.

3.1 Definitions

Given a predicate $f : \mathcal{X} \times \mathcal{X} \to \{0, 1\}$, we define the following notions.

Definition 3.1 (f-friends)**.** Two points $x, y \in \mathcal{X}$ are *f-friends* if $f(x, y) = 1$.

Definition 3.2 (f-friendly dataset)**.** A dataset $x = (x_1, \ldots, x_n) \in \mathcal{X}^n$ is *f-friendly* if for every pair x_i, x_j in x, there exists a point $z \in \mathcal{X}$ (not necessarily in x) such that $f(x_i, z) = 1$ and $f(x_j, z) = 1$.

In other words, every pair of points in a friendly dataset has a common friend.

Definition 3.3 (f-friendly (ε, δ)-DP)**.** An algorithm $A : \mathcal{X}^n \to \mathcal{Y}$ is *f-friendly (ε, δ)-differentially private* if for all neighboring datasets $x \sim x'$ (differing in at most one entry) such that $x \cup x'$ is f-friendly,

$$A(x) \approx_{\varepsilon,\delta} A(x').$$

That is, the usual DP guarantee is required to hold only when the "combined" dataset $x \cup x'$ is f-friendly.

3.2 Examples of Predicates

Example 3.4 (Bounded distance)**.** Let $f(x, y) = \mathbb{1}\{\|x - y\|_2 \leq R\}$. Then x and y are friends if and only if they are within Euclidean distance R of each other.

Example 3.5 (Clustering)**.** Suppose the base algorithm returns an m-tuple of cluster centers $(y_1, \ldots, y_m)$. Two such tuples are f-friends if they can be permuted so that all corresponding cluster centers are close. An f-friendly dataset then ensures that the clustering output is stable under small perturbations of the input.

[1] By which we mean it is stated in quite general terms and applies broadly, not that it is appropriate for every aggregation task.

3.3 The Basic Filter

How do we make a dataset friendly? The Basic Filter (Algorithm 1) achieves this by assigning each data point a weight $w_i \in [0, 1]$, which can be interpreted as the probability of retaining that point.

Algorithm 1 Basic Filter

Input: Data $x_1, \ldots, x_n \in \mathcal{X}$, predicate $f : \mathcal{X} \times \mathcal{X} \to \{0, 1\}$
Returns: Weights $\bar{w} = (w_1, \ldots, w_n) \in [0, 1]^n$

1: **for** $i = 1$ to n **do**
2: $\quad c_i \leftarrow \sum_{j=1}^{n} f(x_i, x_j)$ ▷ Count the number of f-friends of x_i
3: $\quad w_i \leftarrow \begin{cases} 1 & \text{if } c_i \geq n, \\ \dfrac{c_i - n/2}{n} & \text{if } n/2 < c_i < n, \\ 0 & \text{if } c_i \leq n/2. \end{cases}$
4: **end for**
5: **return** $\bar{w} = (w_1, \ldots, w_n)$

Intuitively, c_i counts how many data points are friends of x_i. Points with many friends (at least n) are fully retained ($w_i = 1$), points with few friends (at most $n/2$) are discarded ($w_i = 0$), and points in between are assigned a weight that interpolates linearly.

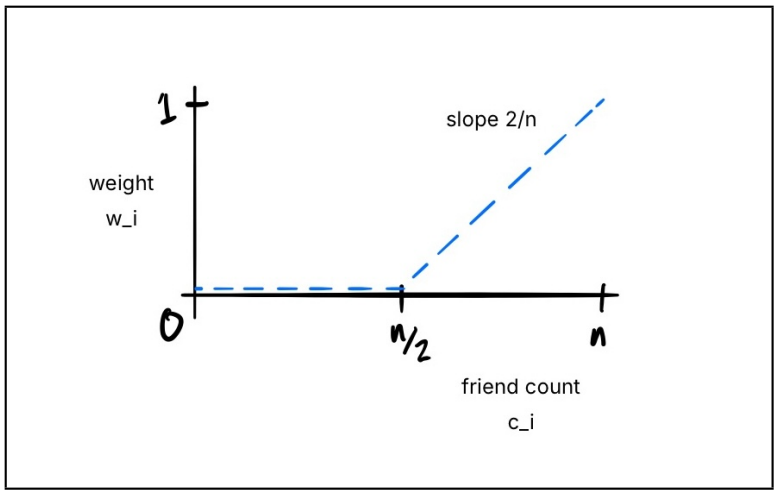

Figure 1: How FriendlyCore's BasicFilter assigns weights based on the number of friends. In the region where the function is increasing, its slope is $2/n$ (we'll use this in a proof).

Remark 3.6. Algorithm 1 differs from the presentation in Tsfadia et al. [2022] in a couple of ways. Primarily, our version outputs weights while their version returns a dataset where each point is retained independently with probability w_i. This is not a major difference; the analysis is essentially the same in either case. We like this to have deterministic algorithms when we can, both for simplicity's sake and viewing randomness as a resource to be consumed. However, some downstream tasks may not accommodate weighted datasets. In these cases, our only choice is to randomize.

3.4 Properties of the Basic Filter

We now state three key properties of the Basic Filter.

Claim 3.7 (Soundness: output is f-friendly)**.** *The output of the Basic Filter (restricted to points with $w_i > 0$) is f-friendly.*

Proof. Any point that survives the filter (i.e., has $w_i > 0$) must have strictly more than $n/2$ friends in the dataset. If two surviving points x_i and x_j each have more than $n/2$ friends, then by the pigeonhole principle, their sets of friends must overlap, so they share at least one common friend. □

Note that the output could be the all-zeros vector, corresponding to an empty data set.

Claim 3.8 (Completeness)**.** *If for all i, j we have $f(x_i, x_j) = 1$, then BasicFilter returns $w = 1$.*

This is the claim that we will usually use in our accuracy analysis: if the data is outlier-free, then nothing is removed.

Finally: adjacent datasets imply closeness in the output weights.

Claim 3.9 (Stability of weights)**.** *If $x \sim x'$ (i.e., x and x' are neighboring datasets differing in one entry), then*

$$\|w(x) - w(x')\|_1 \leq 3.$$

Proof. On input x we compute counts $c_1, \ldots, c_n$ and weights $w_1, \ldots, w_n$ and on input x' we compute $c'_1, \ldots, c'_n$ and $w'_1, \ldots, w'_n$. Suppose that x and x' differ in their first entry (by symmetry, this is without loss of generality). Then w_1 and w'_1 can differ by at most one. Also, changing one data point can change every other friend count by one. This means each other weight can change by at most $2/n$ (see Figure 1). So we have

$$\begin{aligned}
\|w - w'\|_1 &= \sum_{i=1}^{n} |w_i - w'_i| \\
&= |w_1 - w'_1| + \sum_{i=2}^{n} |w_i - w'_i| \\
&\leq 1 + (n-1) \cdot \frac{2}{n}.
\end{aligned}$$

This is at most 3, so we are done. □

Observe that this is the best we can hope for (up to a constant) since we cannot avoid the fact that changing one point may introduce a large outlier with weight zero.

In the next class we will continue using this FriendlyCore framework and apply it to estimating the mean of $\mathcal{N}(\mu, \mathbb{I})$. This means we can hope to apply it as an aggregator in subsample-and-aggregate whenever the non-private estimator has a distribution that looks (somewhat) like a multivariate Gaussian.

References

Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing*, pages 75–84. ACM, 2007. doi: 10.1145/1250790.1250803.

Eliad Tsfadia, Edith Cohen, Haim Kaplan, Yishay Mansour, and Uri Stemmer. Friendlycore: Practical differentially private aggregation. In *International Conference on Machine Learning*, pages 21828–21863. PMLR, 2022.