Final Submission - Sprint 1

Team: SynCity

November 13, 2020

1 Main Idea

We produce final counts by answering queries that count the number of users who fit some criteria in the private dataset. These queries have sensitivity 1. We then aggregate the results of these queries to a produce our final output.

2 Pre-processing

We construct our queries that count users by creating dictionary structure **Q** that maps every combination of **N**, **Y**, **M**, **D**, and **T** (defined below) to the number of users that satisfy these attributes.

- N - neighborhood
- Y - year
- M - month
- I - incident type
- T - number of times a person made a call of incident type **I** from neighborhood **N** during a time period (year **Y** - month **M**).

[main.py: lines 81-185] We construct our final output histogram **C**, which is every combination of **N**, **Y**, **M**, and **D**. We can construct both **Q** and **C** using the schema provided in parameters.json, which has sensitvity cost 0 since creating these structures does not require interacting with the private dataset.

[main.py: lines 120-128] Next, we want to reduce the number of queries we run, i.e. reduce the number of keys in **Q**. To do so, we use will use information outside the private dataset to find the relevant queries we need. In our final solution, we simply use the development dataset (which has sensitivity cost 0 since it is public). Specifically, we reduce the queries in **Q** to the combinations of **N**, **Y**, **M**, **D**, and **T** that exist/have positive count in our public dataset.

[main.py: lines 147-184] Finally, we expand our set of queries in $\mathbf{Q}$ in the following ways:

1. For each query, we duplicate the query and change month ($\mathbf{M}$) to the preceding and following **3** months. Our motivation is that if we observe in the public dataset that people make some type of call during a certain month, we should also check if people make the same call in the preceding/following months.
2. For each query, we duplicate the query and change the $\mathbf{T}$. Our motivation is that if there exists a person who make a certain type of call $\mathbf{T} = t$ times, we should also check if there are people who make the same calls at other values of $\mathbf{T} = \hat{t}$. In our submission, $\hat{t} \in \{1, 2\}$.

Note that this process only interacts with $\mathbf{Q}$ and has sensitivity cost 0.

3 Privatization

Citing Dwork et al. (2014), we first define the following:

Definition 3.1 (Differential Privacy (DP))**.** A randomized algorithm $\mathcal{M} : \mathcal{X}^* \to \mathcal{R}$ satisfies (ε, δ)-differential privacy (DP) if or all databases x, x' differing at most one entry, and every measurable subset $\subseteq \mathcal{R}$, we have that

$$\Pr[\mathcal{M}(x) \in S] \le e^{\varepsilon} \Pr\left[\mathcal{M}\left(x'\right) \in S\right] + \delta$$

If $\delta = 0$, we say that $\mathcal{M}$ satisfies ϵ-differential privacy.

Definition 3.2 (l_1-sensitivity)**.** The l_1-sensitivity of a function $f : \mathrm{N}^{|\mathcal{X}|} \to \mathbb{R}^k$ is:

$$\Delta f = \max_{\substack{x, y \in \mathbb{N}|X| \\ \|x - y\|_1 = 1}} \|f(x) - f(y)\|_1$$

Definition 3.3 (The Laplace Mechanism)**.** Given any function $f : \mathrm{N}^{|\mathcal{X}|} \to \mathbb{R}^k$, the Laplace Mechanism is defined as:

$$\mathcal{M}_L(x, f(\cdot), \varepsilon) = f(x) + (Y_1, \ldots, Y_k)$$

where Y_i are i.i.d. random variables drawn from $\mathrm{Lap}(\Delta f / \varepsilon)$.

Each query counts the number of users that satisfy some set of attributes defined by the keys in $\mathbf{Q}$. Therefore for each query, one person can contribute to at most 1 to the answer. Therefore the l_1-sensitivity of queries in $\mathbf{Q}$ is 1.

[main.py: lines 200-245] To achieve differential privacy, we add Laplacian noise with scale = $\frac{1}{\epsilon}$. We then group the results of these querys by **N**, **Y**, **M**, and **D** to fill in the values in our final histogram **C**.

We therefore achieve ϵ-**DP** using the Laplace Mechanism.

4 Post-processing

The submission rules require finalized counts to be positive integers. To address this, we set all negative values to 0 and round all outputs to integer values.

References

Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4): 211–407, 2014.