Lecture 16: Differential Privacy (Local Model)

MIT - 6.893

Fall 2020

Henry Corrigan-Gibbs

Plan

* Discussion from last time.
* Recap
  - ↳ Diff Privacy
  - ↳ Laplace Mechanism
* Difficulties in practice
* Local Model
* Applications

Logistics

* HW5 due Friday 11/13 at 5pm via Gradescope
* No class on 11/11 ! Veterans' Day Holiday
* Guest lecture on Monday 11/16 (Emily Stark, Google)

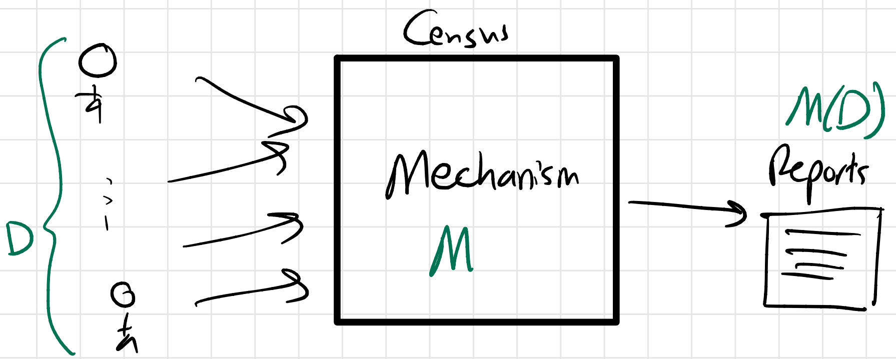
Recap: Differential Privacy

Two Parts

1. Definition of privacy-preserving systems.
   - \+ Strong, precise, robust
   - \- Difficult to satisfy.

2. Mechanisms – Protocols & systems that satisfy this defn.

So far...



Defn: Mechanism $\mathcal{M}$ satisfies ε-DP if $\forall$ neighboring DBs D, D' and every set of values S in $\mathrm{Im}(\mathcal{M})$

$$\Pr_{\mathcal{M}}\left[\mathcal{M}(D) \in S\right] \leq e^{\varepsilon} \cdot \Pr_{\mathcal{M}}\left[\mathcal{M}(D') \in S\right].$$

* Typically ε is small constant e.g. $\varepsilon = 0.1, 1, 5$

* $\varepsilon = 0 \rightarrow$ Perfect Privacy, $\varepsilon = \infty \rightarrow$ no privacy

* If there's an output you don't like (S), if $\mathcal{M}$ satisfies ε-DP, prob of output occuring increases by only a little (e^{ε}) if your data included

* $\Rightarrow$ Mechanism $\mathcal{M}$ must be randomized (if non-trivial)

Why we like it: post-processing, composition, group privacy...

Recap: Laplace Mechanism

A simple way to achieve ε-DP in certain cases.

E.g. You take a survey of students, asking whether they voted for candidate ___ or not.

$$x_i = \begin{cases} 1 & \text{if student } i \text{ voted for } \_\_\_ \\ 0 & \text{o.w} \end{cases}$$

Want to publish $\mathbf{S} = \sum_{i=1}^{n} x_i$ w/ ε-DP

Idea of Laplace Mechanism: Publish $\mathbf{S}$ + noise

$\text{Lap}(1/\varepsilon)$

Smaller ε (more privacy) $\longrightarrow$ more noise
Bigger ε (less privacy) $\longrightarrow$ less noise

Noise comes from Laplace distribution

Laplace
mean: 0
Var: $2/\varepsilon^2$
PDF: $\frac{\varepsilon}{2} \cdot e^{-\varepsilon \cdot |x|}$

Very simple! So easy to implement!

Are we done? ....

Difficulties using DP in practice

→ As you release more statistics, effective $\varepsilon \to$ BIG (sums up). Very quickly, the privacy guarantee becomes vacuous.

↳ No good way to "reset" privacy budget

→ Non-sensical outputs. E.g. in census, cities w/ negative population.

→ Data consistency: Need marginals to add up, etc.

→ Analyzing complex mechanisms (e.g. ML training) is very difficult.

→ What is the right value of ε?

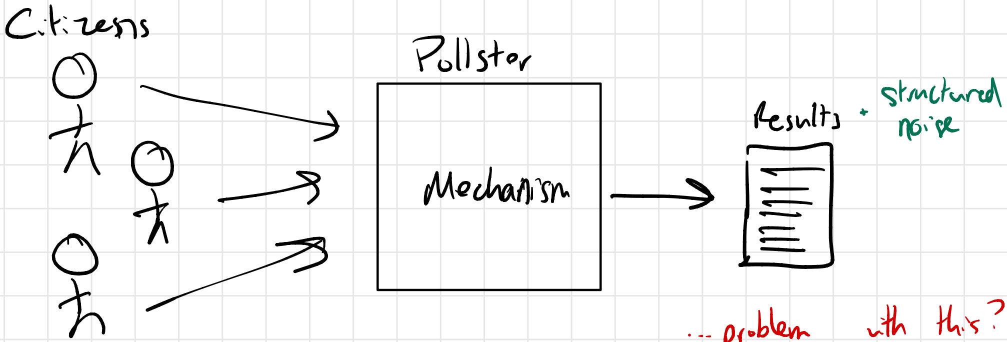
Take away:

DP is one powerful and important defn of privacy.
It doesn't solve all of our problems.
It doesn't always perfectly capture true privacy leakage.
But it is the best we have so far.

→ Central party still has all of your data?!
(Breach, surveillance, ...)

Central Model (so far)

"Will you vote for ________ on Tuesday?"



... problem with this?

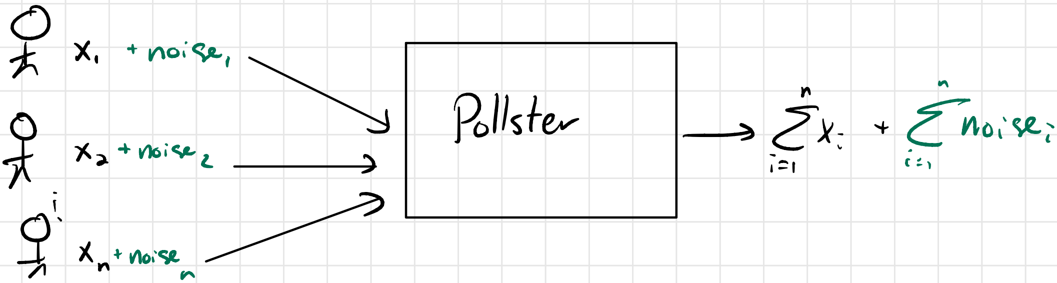
Other examples
* Census data
* Google training ML model

Pro: + Easy to implement (?)
+ Fewer changes to existing processes

Con: − Pollster sees all of your data
↳ No privacy w.r.t. pollster / Census / Google

Local Model ("Randomized response")

Idea: Push mechanism to the edge



e.g. send $\begin{cases} x_i & \text{w.p. } p_\varepsilon \\ \neg x_i & \text{w.p. } 1-p_\varepsilon \end{cases}$ ← Classic randomized response

Pros

+ No central point of privacy failure
+ DP guarantees mean that arbitrary postprocessing ok

Cons

- More noise $\approx \sqrt{n}$ times more (additive error is $\approx \pm\sqrt{n}/\varepsilon$)
- Cannot set ε too small or else noise blows away signal
- Privacy guarantee for user is weaker than under an MPC implementing central model
  - ↳ Pollster still "learns something" about x_i ... can guess x_i w/ non-ngl advantage.

Examples of using Local Model

Apple uses it for collecting telemetry data on iOS and MacOS.

Safari: 2 submissions/user/day
$\varepsilon = 8$ for each submission

$\Rightarrow$ $\varepsilon = 16$ per day... after 1 week, not clear that system is buying much in terms of privacy

Microsoft uses LDP for collecting # of mins that Windows 10 users use each app

$\varepsilon \approx 0.7$ every 6 hours

Chrome used to use LDP for collecting telemetry data ("Rappor")

$\varepsilon \stackrel{?}{\approx} 1$ ... As far as I know, on its way out.

A couple of non-obvious issues...

* Detecting rare events (crash affecting 1% users)

  - Noise is $\pm\sqrt{n}/\varepsilon$.
  - Set $\varepsilon = 0.1$, $\sqrt{n} = 2^{10}$ $(n = 2^{20})$
  - Error is $\approx \pm 2^{13} \approx 1\%$.

  ↳ Hard to distinguish zero from non-zero (small)

  ↳ Partial fix: increase # of users...

  % noise: $\frac{\sqrt{n}\,\varepsilon}{n} = \frac{\varepsilon}{\sqrt{n}} \to 0$ as $n \to$ big.

* Collecting statistics other than sums.
  Common one: Heavy hitters

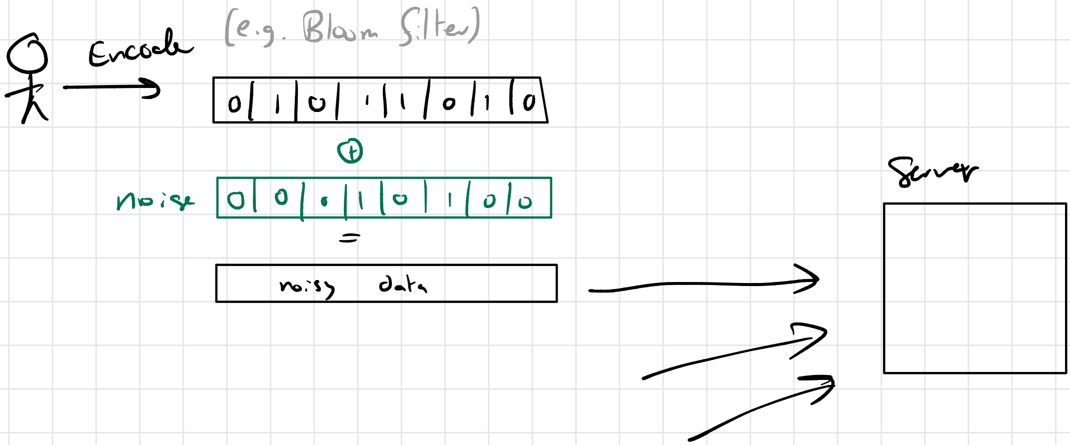
  - Each client i holds a string $x_i \in \{0,1\}^\ell$
  - Servers want all strings that more than 1% of clients hold
    (e.g. homepage, URL crashed browser, ...)

 $\longrightarrow$ URL + noise? ✗

↳ Lots of really clever approaches to solve this problem.

(a la RAPPOR)

General Strategy for Computing Heavy Hitters & Other Statistics w/ LDP



Server avgs noisy vectors, decodes to get output.

e.g. Bloom Filter