Tight and Robust Private Mean Estimation with Few Users

Hossein Esfandiari, Vahab Mirrokni, **Shyam Narayanan**

International Conference on Machine Learning (ICML), 2022

Outline

Outline

1) Introduction

Outline

1) Introduction
2) Results

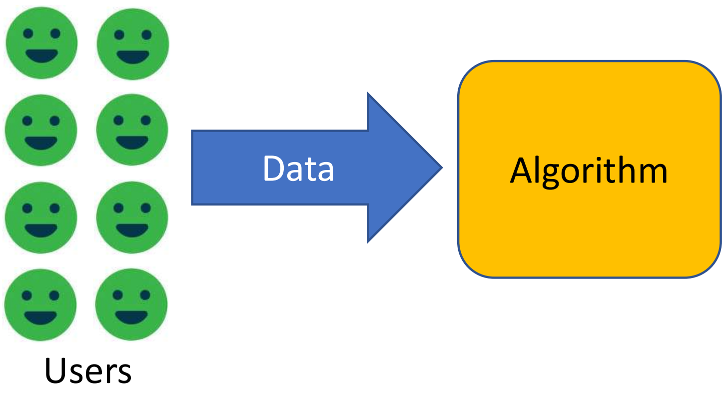
Outline

1) Introduction
2) Results
3) Overview of Algorithm

Establishing Privacy of Released Data

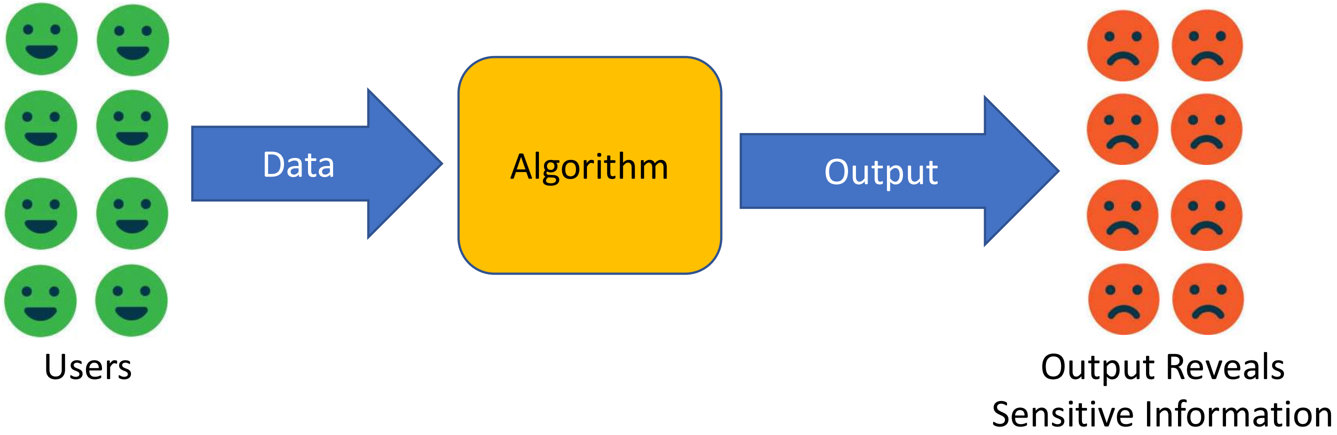
Establishing Privacy of Released Data

- Many scientific/technological endeavors involve learning from data.



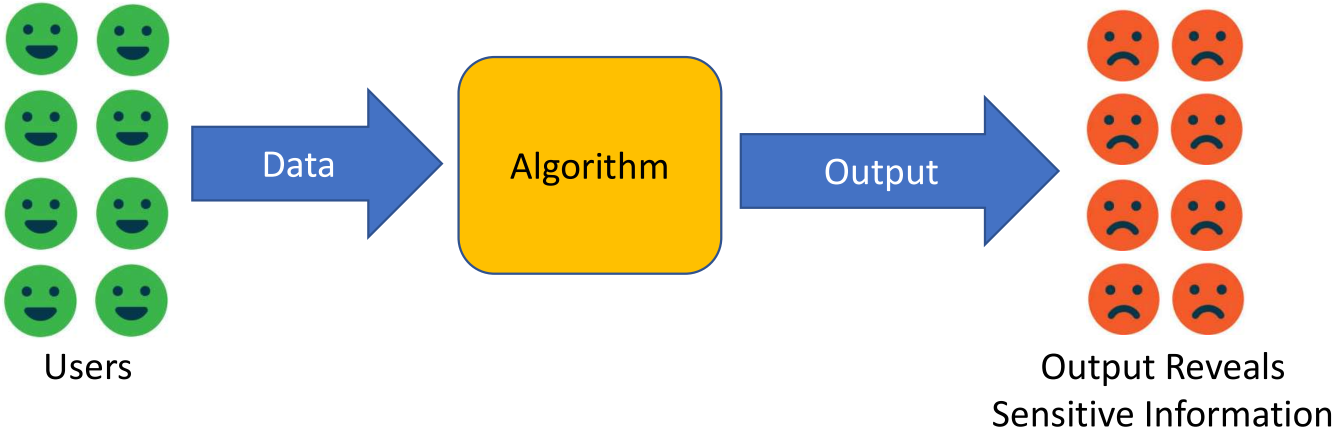
Establishing Privacy of Released Data

- Many scientific/technological endeavors involve learning from data.
- Datasets may include highly sensitive information about individuals.



Establishing Privacy of Released Data

- Many scientific/technological endeavors involve learning from data.
- Datasets may include highly sensitive information about individuals.
- Can we learn properties of data without revealing sensitive info?



Differential Privacy

Differential Privacy

- Provably ensures that no data point has its privacy compromised.

Differential Privacy

- Provably ensures that no data point has its privacy compromised.
- A randomized algorithm $\mathcal{A}$ acting on a dataset $X = \{X_1, X_2, \dots, X_n\}$ is $(\boldsymbol{\varepsilon}, \boldsymbol{\delta})$**-differentially private** ((ε, δ)-DP) if for any two **adjacent** datasets X, X' (i.e., only one data point changes) and any subset S,

Differential Privacy

- Provably ensures that no data point has its privacy compromised.
- A randomized algorithm $\mathcal{A}$ acting on a dataset $X = \{X_1, X_2, \dots, X_n\}$ is **$(\boldsymbol{\varepsilon}, \boldsymbol{\delta})$-differentially private** ((ε, δ)-DP) if for any two **adjacent** datasets X, X' (i.e., only one data point changes) and any subset S,

$$e^{-\varepsilon} \cdot \mathbb{P}(\mathcal{A}(X) \in S) - \delta \leq \mathbb{P}(\mathcal{A}(X') \in S) \leq e^{\varepsilon} \cdot \mathbb{P}(\mathcal{A}(X) \in S) + \delta.$$

Differential Privacy

- Provably ensures that no data point has its privacy compromised.
- A randomized algorithm $\mathcal{A}$ acting on a dataset $X = \{X_1, X_2, \dots, X_n\}$ is **$(\boldsymbol{\varepsilon}, \boldsymbol{\delta})$-differentially private** ((ε, δ)-DP) if for any two **adjacent** datasets X, X' (i.e., only one data point changes) and any subset S,

$$e^{-\varepsilon} \cdot \mathbb{P}(\mathcal{A}(X) \in S) - \delta \leq \mathbb{P}(\mathcal{A}(X') \in S) \leq e^{\varepsilon} \cdot \mathbb{P}(\mathcal{A}(X) \in S) + \delta.$$

- This means that conditioned on seeing any output, we won't know if any individual data point X_i was in fact some other data point X_i'.

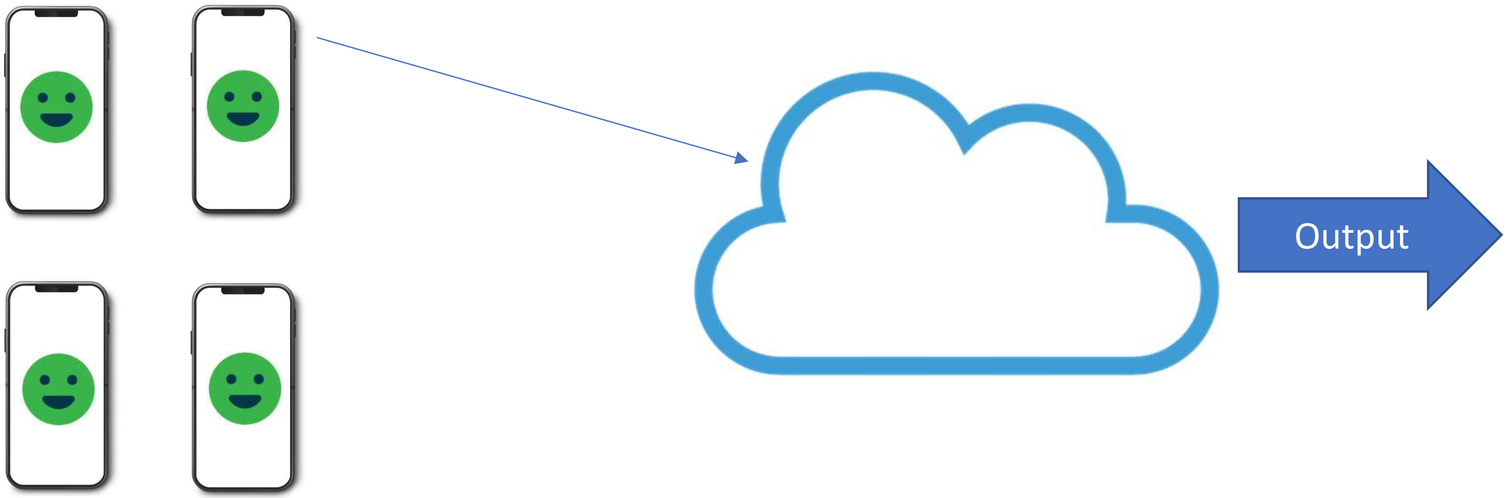
User-Level Privacy

User-Level Privacy

- Our goal is not to protect the privacy of a data point, but rather to protect the privacy of each **user** who contributes data points.

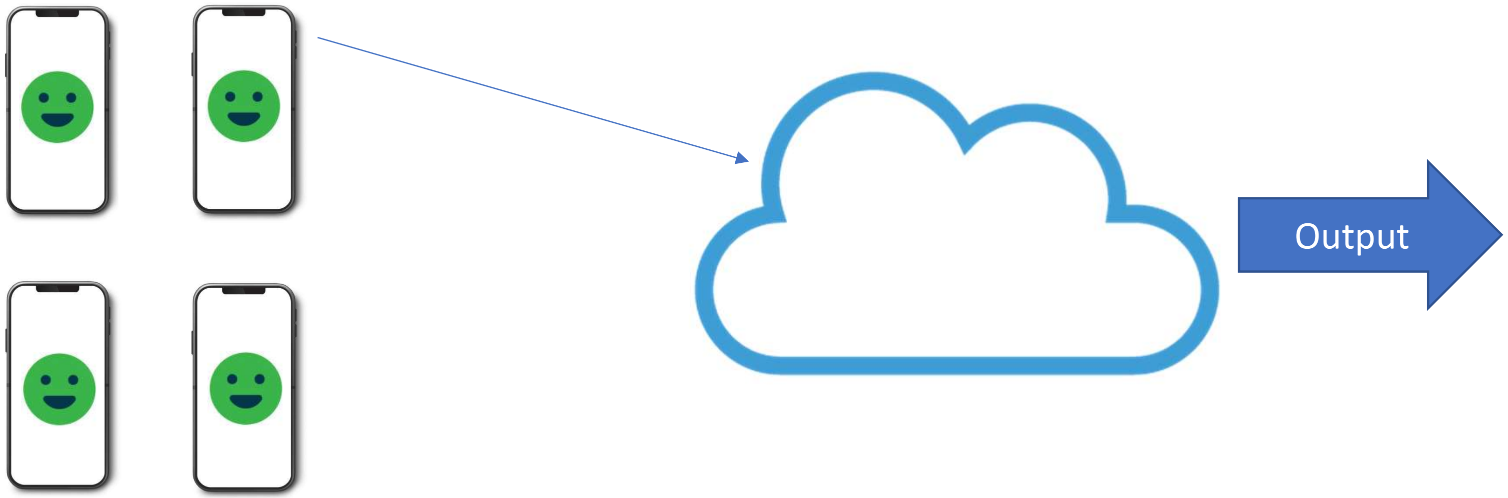
User-Level Privacy

- Our goal is not to protect the privacy of a data point, but rather to protect the privacy of each **user** who contributes data points.



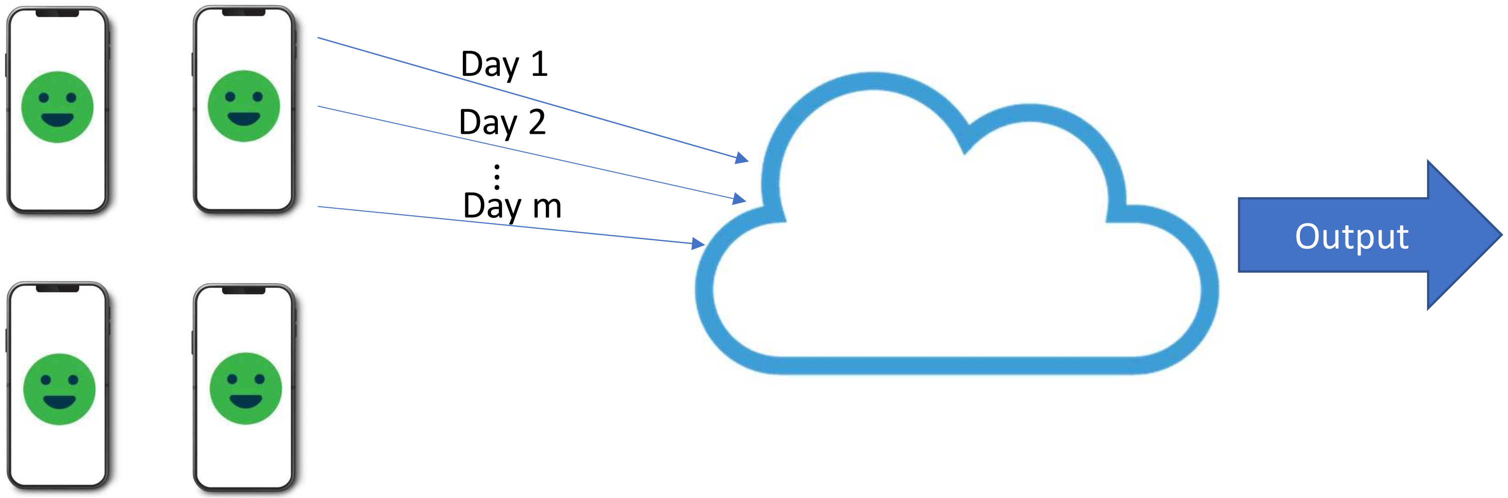
User-Level Privacy

- Our goal is not to protect the privacy of a data point, but rather to protect the privacy of each **user** who contributes data points.
- Users may contribute more than one data point!



User-Level Privacy

- Our goal is not to protect the privacy of a data point, but rather to protect the privacy of each **user** who contributes data points.
- Users may contribute more than one data point!



User-Level Differential Privacy (DP)

User-Level Differential Privacy (DP)

- n users each with m samples $\{X_{i,j}\}_{j=1}^{m}$. **Adjacent:** data of at most one user changes (but **all** the samples of that user may change).

User-Level Differential Privacy (DP)

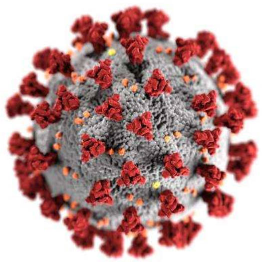
- n users each with m samples $\{X_{i,j}\}_{j=1}^{m}$. **Adjacent:** data of at most one user changes (but **all** the samples of that user may change).
- **Main question:** How much more difficult is ensuring **user-level** privacy as opposed to **item-level** privacy?

User-Level Differential Privacy (DP)

- n users each with m samples $\{X_{i,j}\}_{j=1}^{m}$. **Adjacent:** data of at most one user changes (but **all** the samples of that user may change).
- **Main question:** How much more difficult is ensuring **user-level** privacy as opposed to **item-level** privacy?
- Setting with very few users (but perhaps many samples per user):

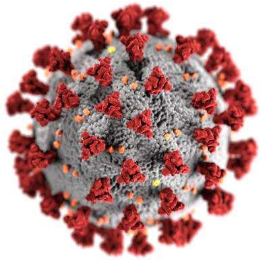
User-Level Differential Privacy (DP)

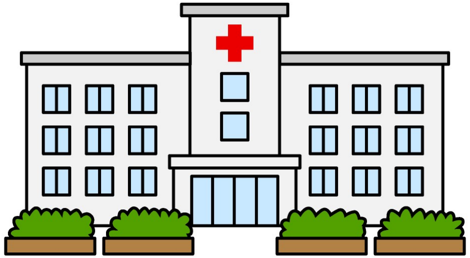
- n users each with m samples $\{X_{i,j}\}_{j=1}^{m}$. **Adjacent:** data of at most one user changes (but **all** the samples of that user may change).
- **Main question:** How much more difficult is ensuring **user-level** privacy as opposed to **item-level** privacy?
- Setting with very few users (but perhaps many samples per user):
  - Analyzing rare situation (such as understanding Covid-19 in early days).



User-Level Differential Privacy (DP)

- n users each with m samples $\{X_{i,j}\}_{j=1}^{m}$. **Adjacent:** data of at most one user changes (but **all** the samples of that user may change).
- **Main question:** How much more difficult is ensuring **user-level** privacy as opposed to **item-level** privacy?
- Setting with very few users (but perhaps many samples per user):
  - Analyzing rare situation (such as understanding Covid-19 in early days).
  - Analyzing local information (such as information of each hospital separately).





Mean Estimation

Mean Estimation

- One of the simplest learning problems.

Mean Estimation

- One of the simplest learning problems.
- Given a distribution $\mathcal{D}$ over $\mathbb{R}^d$, compute the mean $\mu = \mathbb{E}[\mathcal{D}]$.

Mean Estimation

- One of the simplest learning problems.
- Given a distribution $\mathcal{D}$ over $\mathbb{R}^d$, compute the mean $\mu = \mathbb{E}[\mathcal{D}]$.
- Fundamental subroutine in extremely wide array of learning algos.

Mean Estimation

- One of the simplest learning problems.
- Given a distribution $\mathcal{D}$ over $\mathbb{R}^d$, compute the mean $\mu = \mathbb{E}[\mathcal{D}]$.
- Fundamental subroutine in extremely wide array of learning algos.
- In our setting, we suppose each of n users outputs m i.i.d. samples from $\mathcal{D}$. Goal is to estimate μ with user-level DP.

Mean Estimation

- One of the simplest learning problems.
- Given a distribution $\mathcal{D}$ over $\mathbb{R}^d$, compute the mean $\mu = \mathbb{E}[\mathcal{D}]$.
- Fundamental subroutine in extremely wide array of learning algos.
- In our setting, we suppose each of n users outputs m i.i.d. samples from $\mathcal{D}$. Goal is to estimate μ with user-level DP.
- **Our result:** a private and low-error algorithm for mean estimation even with very few users (though each user may have many samples).

Our Results

Our Results

- **Theorem:** Let $\mathcal{D}$ be a distribution over $\mathbb{R}^d$, concentrated in a ball of radius r (around an unknown location) and mean μ. Given $n = O\left(\frac{1}{\varepsilon} \cdot \log \frac{1}{\delta}\right)$ users and m samples per user, there is an (ε, δ)-user level DP algorithm that, if each sample were i.i.d. from $\mathcal{D}$, estimates μ up to error $r\sqrt{d/m}$.

Our Results

- **Theorem:** Let $\mathcal{D}$ be a distribution over $\mathbb{R}^d$, concentrated in a ball of radius r (around an unknown location) and mean μ. Given $n = O\left(\frac{1}{\varepsilon} \cdot \log \frac{1}{\delta}\right)$ users and m samples per user, there is an (ε, δ)-user level DP algorithm that, if each sample were i.i.d. from $\mathcal{D}$, estimates μ up to error $r\sqrt{d/m}$.
- Our algorithm runs in almost linear time in n and d. Our algorithm also works in the robust setting if even 49% of all users have all their samples corrupted (but the rest of the users have all samples intact).

Our Results

- **Theorem:** Let $\mathcal{D}$ be a distribution over $\mathbb{R}^d$, concentrated in a ball of radius r (around an unknown location) and mean μ. Given $n = O\left(\frac{1}{\varepsilon} \cdot \log \frac{1}{\delta}\right)$ users and m samples per user, there is an (ε, δ)-user level DP algorithm that, if each sample were i.i.d. from $\mathcal{D}$, estimates μ up to error $r\sqrt{d/m}$.
- Our algorithm runs in almost linear time in n and d. Our algorithm also works in the robust setting if even 49% of all users have all their samples corrupted (but the rest of the users have all samples intact).
- Algorithm can be applied to various learning problems (learning discrete distributions, stochastic convex optimization, etc.).

Our Results (pt 2)

Our Results (pt 2)

- We also show a tight trade-off between number of users n, number of samples per user m, and the overall error in estimating μ.

Our Results (pt 2)

- We also show a tight trade-off between number of users n, number of samples per user m, and the overall error in estimating μ.
- Answers a conjecture of Amin et al. (ICML 2019) asking about this user-sample tradeoff.

Our Results (pt 2)

- We also show a tight trade-off between number of users n, number of samples per user m, and the overall error in estimating μ.
- Answers a conjecture of Amin et al. (ICML 2019) asking about this user-sample tradeoff.
- Also improves over previous work of Liu et al. (NeurIPS 2020) and Levy et al. (NeurIPS 2021) which required $n \gg \sqrt{d \log \frac{1}{\delta}} / \varepsilon$.

Main Focus

Main Focus

- Will focus on (non-robust) mean estimation.

Main Focus

- Will focus on (non-robust) mean estimation.
- **Recall goal:** We have $n = O\left(\frac{1}{\varepsilon} \cdot \log \frac{1}{\delta}\right)$ users, each with m samples in $\mathbb{R}^d$ from $\mathcal{D}$, bounded in unknown ball of radius r.

Main Focus

- Will focus on (non-robust) mean estimation.
- **Recall goal:** We have $n = O\left(\frac{1}{\varepsilon} \cdot \log \frac{1}{\delta}\right)$ users, each with m samples in $\mathbb{R}^d$ from $\mathcal{D}$, bounded in unknown ball of radius r.
- Want to estimate $\mu = \mathbb{E}[\mathcal{D}]$ up to error $r\sqrt{d/m}$.

Main Focus

- Will focus on (non-robust) mean estimation.
- **Recall goal:** We have $n = O\left(\frac{1}{\varepsilon} \cdot \log \frac{1}{\delta}\right)$ users, each with m samples in $\mathbb{R}^d$ from $\mathcal{D}$, bounded in unknown ball of radius r.
- Want to estimate $\mu = \mathbb{E}[\mathcal{D}]$ up to error $r\sqrt{d/m}$.
- Can show sample mean of each user is $O\left(\frac{r}{\sqrt{m}}\right)$ away from μ, so suffices to solve the item-level privacy problem by scaling.

Main Focus

- Will focus on (non-robust) mean estimation.
- **Recall goal:** We have $n = O\left(\frac{1}{\varepsilon} \cdot \log \frac{1}{\delta}\right)$ users, each with m samples in $\mathbb{R}^d$ from $\mathcal{D}$, bounded in unknown ball of radius r.
- Want to estimate $\mu = \mathbb{E}[\mathcal{D}]$ up to error $r\sqrt{d/m}$.
- Can show sample mean of each user is $O\left(\frac{r}{\sqrt{m}}\right)$ away from μ, so suffices to solve the item-level privacy problem by scaling.
- Given $n = O\left(\frac{1}{\varepsilon} \cdot \log \frac{1}{\delta}\right)$ points $X_1, \dots, X_n$ in unknown ball of radius 1, approximate the ball up to error $\sqrt{d}$.

Algorithm (Attempt 1)

Algorithm (Attempt 1)

- Based on **exponential mechanism**: assign a score $s(p)$ for any point p, and sample p with density proportional to $e^{\varepsilon \cdot s(p)}$.

Algorithm (Attempt 1)

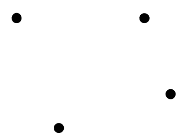
- Based on **exponential mechanism**: assign a score $s(p)$ for any point p, and sample p with density proportional to $e^{\varepsilon \cdot s(p)}$.
- $s(p)$: the number of points among $X_1, \ldots, X_n$ within $\sqrt{d}$ of p.

Algorithm (Attempt 1)

- Based on **exponential mechanism**: assign a score $s(p)$ for any point p, and sample p with density proportional to $e^{\varepsilon \cdot s(p)}$.
- $s(p)$: the number of points among $X_1, \dots, X_n$ within $\sqrt{d}$ of p.
- Private because changing single data point changes $s(p)$ by at most 1.

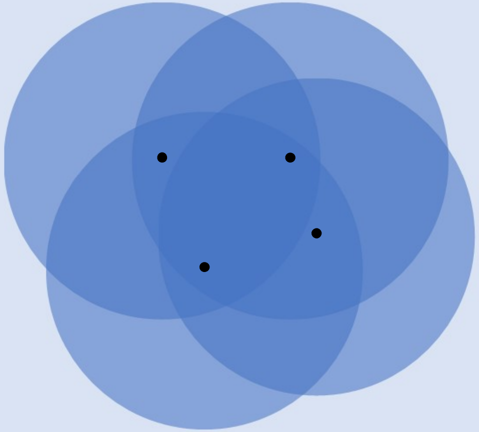
Algorithm (Attempt 1)

- Based on **exponential mechanism**: assign a score $s(p)$ for any point p, and sample p with density proportional to $e^{\varepsilon \cdot s(p)}$.
- $s(p)$: the number of points among $X_1, \ldots, X_n$ within $\sqrt{d}$ of p.
- Private because changing single data point changes $s(p)$ by at most 1.

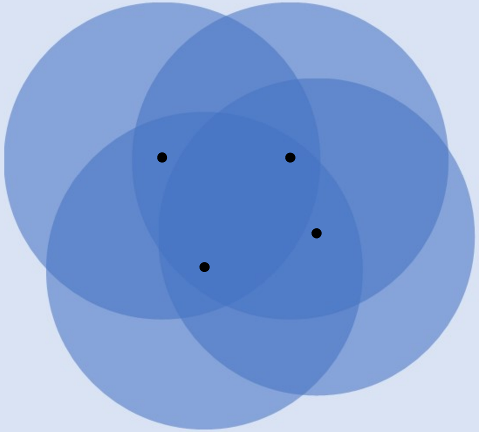


Algorithm (Attempt 1)

- Based on **exponential mechanism**: assign a score $s(p)$ for any point p, and sample p with density proportional to $e^{\varepsilon \cdot s(p)}$.
- $s(p)$: the number of points among $X_1, \ldots, X_n$ within $\sqrt{d}$ of p.
- Private because changing single data point changes $s(p)$ by at most 1.

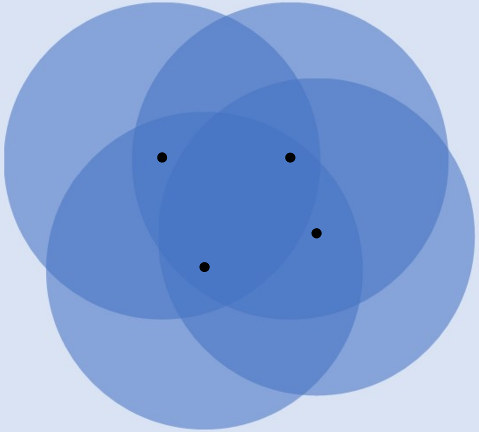


Attempt 2: Fixing Accuracy



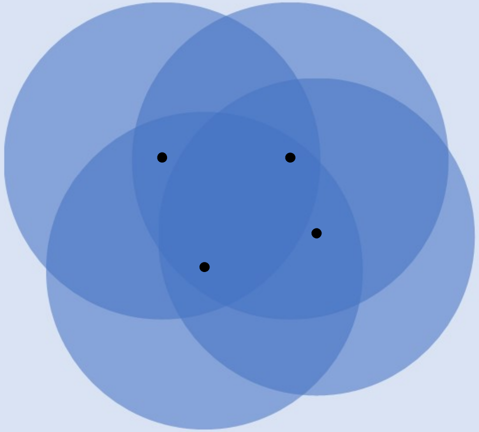
Attempt 2: Fixing Accuracy

- Problem: infinite space gets sampled also: won’t sample point near center.



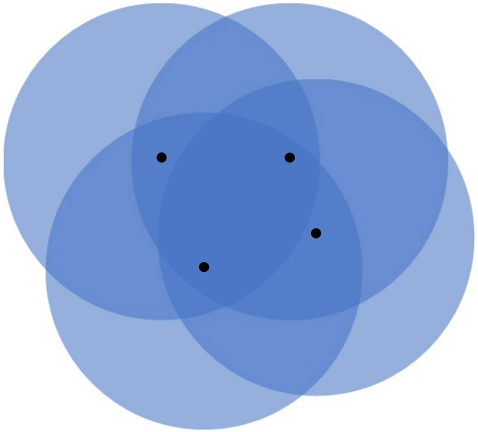
Attempt 2: Fixing Accuracy

- Problem: infinite space gets sampled also: won't sample point near center.
- Attempt to fix: sample proportional to $e^{\varepsilon \cdot s(p)}$ only when $s(p) \geq 1$.



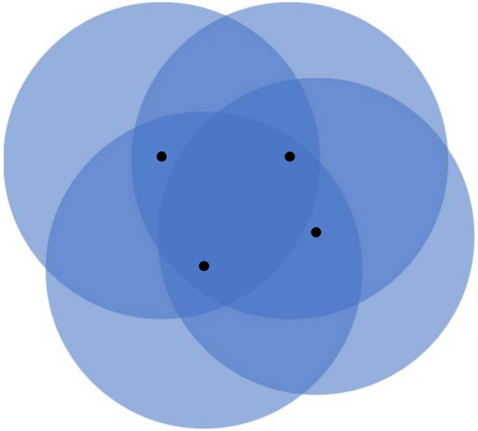
Attempt 2: Fixing Accuracy

- Problem: infinite space gets sampled also: won't sample point near center.
- Attempt to fix: sample proportional to $e^{\varepsilon \cdot s(p)}$ only when $s(p) \geq 1$.

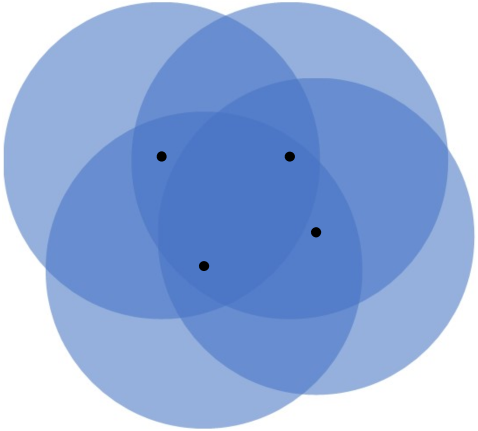


Attempt 2: Fixing Accuracy

- Problem: infinite space gets sampled also: won't sample point near center.
- Attempt to fix: sample proportional to $e^{\varepsilon \cdot s(p)}$ only when $s(p) \geq 1$.
- Now we lose privacy because sampling probability can drastically change from $s(p) = 1$ to $s(p) = 0$.

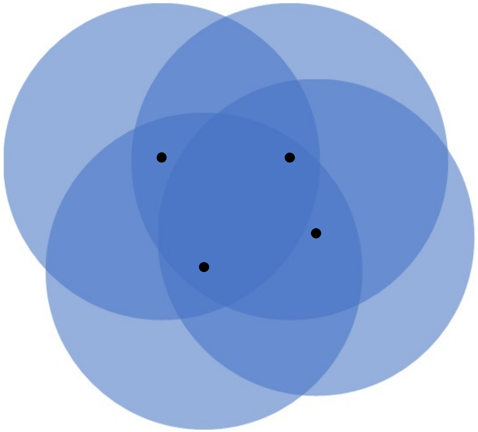


Attempt 3: Fixing Privacy



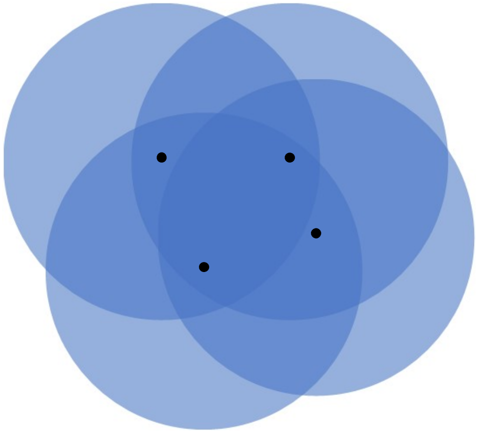
Attempt 3: Fixing Privacy

- If we change one point, volume of points that change from $s(p) = 1$ to $s(p) = 0$ is at most V, the volume of a ball of radius $\sqrt{d}$ in $\mathbb{R}^d$.



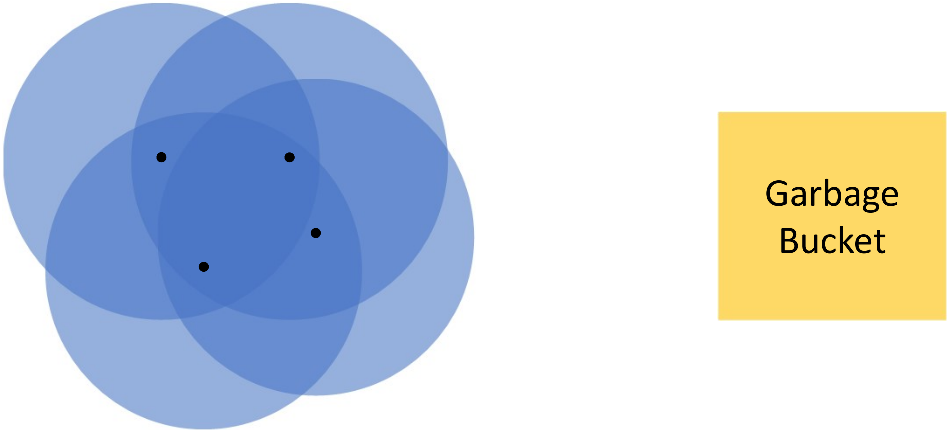
Attempt 3: Fixing Privacy

- If we change one point, volume of points that change from $s(p) = 1$ to $s(p) = 0$ is at most V, the volume of a ball of radius $\sqrt{d}$ in $\mathbb{R}^d$.
- We add "garbage bucket" of size proportional to V/δ to drown out the volume of points that lose privacy.



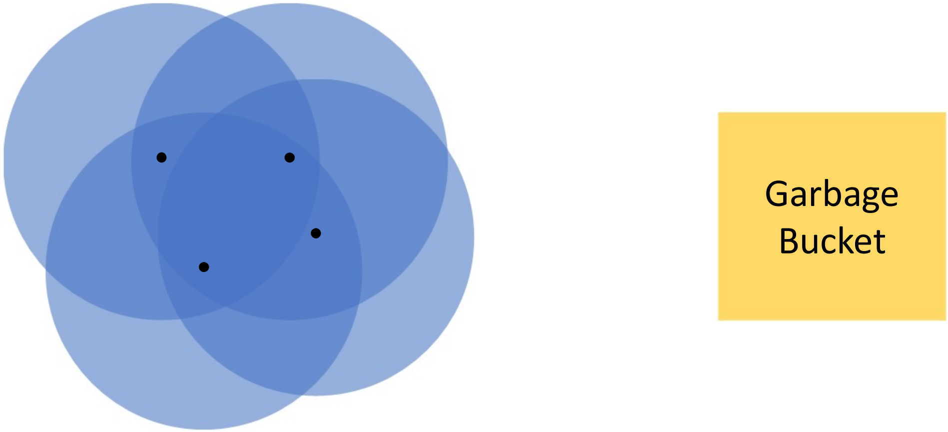
Attempt 3: Fixing Privacy

- If we change one point, volume of points that change from $s(p) = 1$ to $s(p) = 0$ is at most V, the volume of a ball of radius $\sqrt{d}$ in $\mathbb{R}^d$.
- We add "garbage bucket" of size proportional to V/δ to drown out the volume of points that lose privacy.



Attempt 3: Fixing Privacy

- If we change one point, volume of points that change from $s(p) = 1$ to $s(p) = 0$ is at most V, the volume of a ball of radius $\sqrt{d}$ in $\mathbb{R}^d$.
- We add "garbage bucket" of size proportional to V/δ to drown out the volume of points that lose privacy.
- Sample garbage bucket proportional to V/δ to ensure (ε, δ)-DP.



Why is this accurate?

Why is this accurate?

- Need to ensure that we do not sample garbage bucket w.h.p. if points $X_1, \ldots, X_n$ are all in ball of radius 1.

Why is this accurate?

- Need to ensure that we do not sample garbage bucket w.h.p. if points $X_1, \dots, X_n$ are all in ball of radius 1.
- **Key Lemma:** if $X_1, \dots, X_n$ are in ball of radius 1, then the intersection of the balls of radius $\sqrt{d}$ around each X_i has volume at least $e^{-\sqrt{\log n}} \cdot V$, where V is the volume of a single ball of radius $\sqrt{d}$.

Why is this accurate?

- Need to ensure that we do not sample garbage bucket w.h.p. if points $X_1, \ldots, X_n$ are all in ball of radius 1.
- **Key Lemma:** if $X_1, \ldots, X_n$ are in ball of radius 1, then the intersection of the balls of radius $\sqrt{d}$ around each X_i has volume at least $e^{-\sqrt{\log n}} \cdot V$, where V is the volume of a single ball of radius $\sqrt{d}$.
- These points in intersection have density $e^{\varepsilon \cdot n}$ since $s(p) = n$.

Why is this accurate?

- Need to ensure that we do not sample garbage bucket w.h.p. if points $X_1, \dots, X_n$ are all in ball of radius 1.
- **Key Lemma:** if $X_1, \dots, X_n$ are in ball of radius 1, then the intersection of the balls of radius $\sqrt{d}$ around each X_i has volume at least $e^{-\sqrt{\log n}} \cdot V$, where V is the volume of a single ball of radius $\sqrt{d}$.
- These points in intersection have density $e^{\varepsilon \cdot n}$ since $s(p) = n$.
- Need $$\underbrace{\frac{V}{\delta}}_{\text{volume of garbage bucket}} \ll \underbrace{e^{-\sqrt{\log n}} \cdot V}_{\text{volume of intersection}} \cdot \underbrace{e^{\varepsilon \cdot n}}_{\text{density}}.$$

Why is this accurate?

- Need to ensure that we do not sample garbage bucket w.h.p. if points $X_1, \dots, X_n$ are all in ball of radius 1.
- **Key Lemma:** if $X_1, \dots, X_n$ are in ball of radius 1, then the intersection of the balls of radius $\sqrt{d}$ around each X_i has volume at least $e^{-\sqrt{\log n}} \cdot V$, where V is the volume of a single ball of radius $\sqrt{d}$.
- These points in intersection have density $e^{\varepsilon \cdot n}$ since $s(p) = n$.
- Need $$\underbrace{\frac{V}{\delta}}_{\text{volume of garbage bucket}} \ll \underbrace{e^{-\sqrt{\log n}} \cdot V}_{\text{volume of intersection}} \cdot \underbrace{e^{\varepsilon \cdot n}}_{\text{density}}.$$
- Holds if $n \geq O(\varepsilon^{-1} \log \delta^{-1})$.

Getting a Fast Algorithm

Getting a Fast Algorithm

- While the previous algorithm is optimal from an information theoretic perspective, it is not clear how to implement this efficiently.

Getting a Fast Algorithm

- While the previous algorithm is optimal from an information theoretic perspective, it is not clear how to implement this efficiently.
- Difficult to run the sampling procedure, can take exponential time.

Getting a Fast Algorithm

- While the previous algorithm is optimal from an information theoretic perspective, it is not clear how to implement this efficiently.
- Difficult to run the sampling procedure, can take exponential time.
- Fast algorithm via method of **Rejection Sampling**.

Getting a Fast Algorithm

- While the previous algorithm is optimal from an information theoretic perspective, it is not clear how to implement this efficiently.
- Difficult to run the sampling procedure, can take exponential time.
- Fast algorithm via method of **Rejection Sampling**.
- Will sample each point X_i and a random point p in the ball of radius $\sqrt{d}$ around X_i.

Getting a Fast Algorithm

- While the previous algorithm is optimal from an information theoretic perspective, it is not clear how to implement this efficiently.
- Difficult to run the sampling procedure, can take exponential time.
- Fast algorithm via method of **Rejection Sampling**.
- Will sample each point X_i and a random point p in the ball of radius $\sqrt{d}$ around X_i.
- We "accept" the point with probability based on $s(p)$, and reject otherwise, to keep the distribution proportional to $e^{\varepsilon \cdot s(p)}$.

Getting a Fast Algorithm

- While the previous algorithm is optimal from an information theoretic perspective, it is not clear how to implement this efficiently.
- Difficult to run the sampling procedure, can take exponential time.
- Fast algorithm via method of **Rejection Sampling**.
- Will sample each point X_i and a random point p in the ball of radius $\sqrt{d}$ around X_i.
- We "accept" the point with probability based on $s(p)$, and reject otherwise, to keep the distribution proportional to $e^{\varepsilon \cdot s(p)}$.
- If we reject, we keep trying until we accept a point p.

A few caveats

A few caveats

- Rejection sampling may still be slow in certain cases, namely when not all points are close together.

A few caveats

- Rejection sampling may still be slow in certain cases, namely when not all points are close together.
- In this setting, however: not necessary to output an accurate answer.

A few caveats

- Rejection sampling may still be slow in certain cases, namely when not all points are close together.
- In this setting, however: not necessary to output an accurate answer.
- **Fix (attempt 1):** stop rejection sampling after some N rounds. Unfortunately, this method may no longer be private.

A few caveats

- Rejection sampling may still be slow in certain cases, namely when not all points are close together.
- In this setting, however: not necessary to output an accurate answer.
- **Fix (attempt 1):** stop rejection sampling after some N rounds. Unfortunately, this method may no longer be private.
- **Fix:** stop after $Expo(N)$ rounds, for $N \approx e^{\sqrt{\log n}} = n^{o(1)}$. Allows for algorithm to be both fast and maintains privacy!

Conclusion

Conclusion

- We obtain an optimal user-level private algorithm for d-dimensional mean estimation. It only requires $\Omega\left(\frac{1}{\varepsilon}\log\frac{1}{\delta}\right)$ users.

Conclusion

- We obtain an optimal user-level private algorithm for d-dimensional mean estimation. It only requires $\Omega\left(\frac{1}{\varepsilon}\log\frac{1}{\delta}\right)$ users.
- Our algorithm improves over previous methods, which required the number of users to depend at least on $\sqrt{d}$.

Conclusion

- We obtain an optimal user-level private algorithm for d-dimensional mean estimation. It only requires $\Omega\left(\frac{1}{\varepsilon}\log\frac{1}{\delta}\right)$ users.
- Our algorithm improves over previous methods, which required the number of users to depend at least on $\sqrt{d}$.
- Algorithm based on modifying exponential mechanism with garbage bucket, and rejection sampling techniques.

Thanks for attending!

Questions?