Smooth Sensitivity and Sampling in Private Data Analysis

Sofya Raskhodnikova ,Kobbi Nissim, Adam Smith
Presented by: Lidor Avigad

Weizmann Institute

March 17, 2008

Differential Privacy

- A client would like to calculate some function on database.
- The function should not reveal any specific information about any user.
- We mask the real output by noise function.
- But the result should be reasonable accurate.

Differential Privacy

- A client would like to calculate some function on database.
- The function should not reveal any specific information about any user.
- We mask the real output by noise function.
- But the result should be reasonable accurate.

Differential Privacy

- A client would like to calculate some function on database.
- The function should not reveal any specific information about any user.
- We mask the real output by noise function.
- But the result should be reasonable accurate.

Differential Privacy

- A client would like to calculate some function on database.
- The function should not reveal any specific information about any user.
- We mask the real output by noise function.
- But the result should be reasonable accurate.

Privacy Definition

Definition:

Definition (Indistinguishability)

A randomized algorithm $\mathcal{A}$, is (ϵ, δ)-indistinguishable if for all $x, y \in D^n$ satisfying $d(x, y) = 1$, and for all sets S of possible outputs:

$$Pr[\mathcal{A}(x) \in S] \leq e^{\epsilon} \cdot Pr[\mathcal{A}(y) \in S] + \delta$$

where δ is negligible function of n.

Privacy Definition-Remarks

- No individual has a pronounced effect on the statistics published by the server.

- Can be considered as a client-server interaction. Each step calculation some function f on database:
  - Composes smoothly - t rounds each individually ϵ-indistinguishable is $t\epsilon$-indistinguishable.
  - will consider only 1-round protocols.

Privacy Definition-Remarks

- No individual has a pronounced effect on the statistics published by the server.
- Can be considered as a client-server interaction. Each step calculation some function f on database:
  - Composes smoothly - t rounds each individually ϵ-indistinguishable is $t\epsilon$-indistinguishable.
  - will consider only 1-round protocols.

Privacy Definition-Remarks

- No individual has a pronounced effect on the statistics published by the server.
- Can be considered as a client-server interaction. Each step calculation some function f on database:
  - Composes smoothly - t rounds each individually ϵ-indistinguishable is $t\epsilon$-indistinguishable.
  - will consider only 1-round protocols.

Privacy Definition-Remarks

- No individual has a pronounced effect on the statistics published by the server.
- Can be considered as a client-server interaction. Each step calculation some function *f* on database:
  - Composes smoothly - t rounds each individually ϵ-indistinguishable is $t\epsilon$-indistinguishable.
  - will consider only 1-round protocols.

Calibrating Noise to Sensitivity

- The use of output *perturbation*. Adding random noise to mask the private information.
- Outputting : $f(x) + Y$ where Y is the random noise added.

Definition (Global Sensitivity)

For $f : D^n \rightarrow \mathbb{R}^d$, the global sensitivity of f is:

$$GS_f = \max_{x,y:d(x,y)=1} \| f(x) - f(y) \|$$

where $\| \cdot \| = \| \cdot \|_1$.

Calibrating Noise to Sensitivity

- The use of output *perturbation*. Adding random noise to mask the private information.
- Outputting : $f(x) + Y$ where Y is the random noise added.

Definition (Global Sensitivity)

For $f : D^n \to \mathbb{R}^d$, the global sensitivity of f is:

$$GS_f = \max_{x,y:d(x,y)=1} \| f(x) - f(y) \|$$

where $\| \cdot \| = \| \cdot \|_1$.

Calibrating Noise to Sensitivity

- The use of output *perturbation*. Adding random noise to mask the private information.
- Outputting : $f(x) + Y$ where Y is the random noise added.

Definition (Global Sensitivity)

For $f : D^n \rightarrow \mathbb{R}^d$, the global sensitivity of f is:

$$GS_f = \max_{x,y:d(x,y)=1} \| f(x) - f(y) \|$$

where $\| \cdot \| = \| \cdot \|_1$.

- Theorem: for $f : D^n \rightarrow \mathbb{R}^d$, $\mathcal{A}(x) = f(x) + (Y_1, ..., Y_d)$ where $Y_i \sim Lap(GS_f/\epsilon)$ is ϵ-indistinguishable.
- Yields two generic approaches to construction $\mathcal{A}(x)$:
  - Show that GS_f is low so can be added directly on $f(x)$.
  - Express $f(x)$ in term of functions $g_1, g_2, ...$ with low global sensitivity. Then analyze how noisy answers $g_1, g_2, ...$ interfere with computation of $f(x)$
- Approaches are productive to many functions: Principle component analysis, the Perceptron algorithm, k-means, learning ID3 decision trees, statistical learning and many more.

- Theorem: for $f : D^n \rightarrow \mathbb{R}^d$, $\mathcal{A}(x) = f(x) + (Y_1, ..., Y_d)$ where $Y_i \sim Lap(GS_f/\epsilon)$ is ϵ-indistinguishable.
- Yields two generic approaches to construction $\mathcal{A}(x)$:
  - Show that GS_f is low so can be added directly on $f(x)$.
  - Express $f(x)$ in term of functions $g_1, g_2, ...$ with low global sensitivity. Then analyze how noisy answers $g_1, g_2, ...$ interfere with computation of $f(x)$
- Approaches are productive to many functions: Principle component analysis, the Perceptron algorithm, k-means, learning ID3 decision trees, statistical learning and many more.

- Theorem: for $f : D^n \rightarrow \mathbb{R}^d$, $\mathcal{A}(x) = f(x) + (Y_1, ..., Y_d)$ where $Y_i \sim Lap(GS_f/\epsilon)$ is ϵ-indistinguishable.
- Yields two generic approaches to construction $\mathcal{A}(x)$:
  - Show that GS_f is low so can be added directly on $f(x)$.
  - Express $f(x)$ in term of functions $g_1, g_2, ...$ with low global sensitivity. Then analyze how noisy answers $g_1, g_2, ...$ interfere with computation of $f(x)$
- Approaches are productive to many functions: Principle component analysis, the Perceptron algorithm, k-means, learning ID3 decision trees, statistical learning and many more.

- Theorem: for $f : D^n \to \mathbb{R}^d$, $\mathcal{A}(x) = f(x) + (Y_1, ..., Y_d)$ where $Y_i \sim Lap(GS_f/\epsilon)$ is ϵ-indistinguishable.
- Yields two generic approaches to construction $\mathcal{A}(x)$:
  - Show that GS_f is low so can be added directly on $f(x)$.
  - Express $f(x)$ in term of functions $g_1, g_2, ...$ with low global sensitivity. Then analyze how noisy answers $g_1, g_2, ...$ interfere with computation of $f(x)$
- Approaches are productive to many functions: Principle component analysis, the Perceptron algorithm, k-means, learning ID3 decision trees, statistical learning and many more.

- Theorem: for $f : D^n \rightarrow \mathbb{R}^d$, $\mathcal{A}(x) = f(x) + (Y_1, ..., Y_d)$ where $Y_i \sim Lap(GS_f/\epsilon)$ is ϵ-indistinguishable.
- Yields two generic approaches to construction $\mathcal{A}(x)$:
  - Show that GS_f is low so can be added directly on $f(x)$.
  - Express $f(x)$ in term of functions $g_1, g_2, ...$ with low global sensitivity. Then analyze how noisy answers $g_1, g_2, ...$ interfere with computation of $f(x)$
- Approaches are productive to many functions: Principle component analysis, the Perceptron algorithm, k-means, learning ID3 decision trees, statistical learning and many more.

Global Sensitivity - Drawbacks

- The global sensitivity does not consider the instance of the database.
- Yields high noise that might destroy the output. Examples to follow...
- Worst case scenario.

Global Sensitivity - Drawbacks

- The global sensitivity does not consider the instance of the database.
- Yields high noise that might destroy the output. Examples to follow...
- Worst case scenario.

Global Sensitivity - Drawbacks

- The global sensitivity does not consider the instance of the database.
- Yields high noise that might destroy the output. Examples to follow...
- Worst case scenario.

Local Sensitivity

- Would like to add noise according to the database instance.
- We add less noise. i.e. tailored noise.
- "Average" case scenario.

Definition (Local Sensitivity)

For $f : D^n \rightarrow \mathbb{R}^d$, the local sensitivity of f at x is:

$$LS_f(x) = \max_{y:d(x,y)=1} \| f(x) - f(y) \|$$

Local Sensitivity

- Would like to add noise according to the database instance.
- We add less noise. i.e. tailored noise.
- "Average" case scenario.

Definition (Local Sensitivity)

For $f : D^n \to \mathbb{R}^d$, the local sensitivity of f at x is:

$$LS_f(x) = \max_{y:d(x,y)=1} \| f(x) - f(y) \|$$

Local Sensitivity

- Would like to add noise according to the database instance.
- We add less noise. i.e. tailored noise.
- "Average" case scenario.

Definition (Local Sensitivity)

For $f : D^n \to \mathbb{R}^d$, the local sensitivity of f at x is:

$$LS_f(x) = \max_{y:d(x,y)=1} \| f(x) - f(y) \|$$

Local Sensitivity

- Would like to add noise according to the database instance.
- We add less noise. i.e. tailored noise.
- "Average" case scenario.

Definition (Local Sensitivity)

For $f : D^n \to \mathbb{R}^d$, the local sensitivity of f at x is:

$$LS_f(x) = \max_{y:d(x,y)=1} \| f(x) - f(y) \|$$

Local Sensitivity - Remarks

- Note that $GS_f = \max_x LS_f(x)$.
- Would like noise magnitude proportional to $LS_f(x)$. Cannot be added directly-too naive.
- Sometimes hard to calculate.

Local Sensitivity - Remarks

- Note that $GS_f = \max_x LS_f(x)$.
- Would like noise magnitude proportional to $LS_f(x)$. Cannot be added directly-too naive.
- Sometimes hard to calculate.

Local Sensitivity - Remarks

- Note that $GS_f = \max_x LS_f(x)$.
- Would like noise magnitude proportional to $LS_f(x)$. Cannot be added directly-too naive.
- Sometimes hard to calculate.

Local Sensitivity Example

Example

$f_{med}(x) = median(x_1, ..., x_n)$ on bounded interval $D = [0, \Lambda]$.

- Worst case: $GS_{f_{med}} = \Lambda$.
- On 'typical' inputs f_{med} is not very sensitive: $LS_{f_{med}} = max\{x_m - x_{m-1}, x_{m+1} - x_m\}$.
- Would like noise magnitude to be proportional $LS_{f(x)}$. However the noise level can reveal information:
  - Consider case: $f_{med}(x) = 0$ and $f_{med}(y) = \Lambda$ s.t. $d(x, y) = 1$.
  - In the first case the probability to get non-zero median is exactly 0.
  - In the second case the probability to get non-zero median is > 0.
  - No differential privacy: $Pr[y \in S] > Pr[x \in S]$ where S is the event "getting non zero median".

Local Sensitivity Example

Example

$f_{med}(x) = median(x_1, ..., x_n)$ on bounded interval $D = [0, \Lambda]$.

- Worst case: $GS_{f_{med}} = \Lambda$.
- On 'typical' inputs f_{med} is not very sensitive: $LS_{f_{med}} = max\{x_m - x_{m-1}, x_{m+1} - x_m\}$.
- Would like noise magnitude to be proportional $LS_{f(x)}$. However the noise level can reveal information:
  - Consider case: $f_{med}(x) = 0$ and $f_{med}(y) = \Lambda$ s.t. $d(x, y) = 1$.
  - In the first case the probability to get non-zero median is exactly 0.
  - In the second case the probability to get non-zero median is > 0.
  - No differential privacy: $Pr[y \in S] > Pr[x \in S]$ where S is the event "getting non zero median".

Local Sensitivity Example

Example

$f_{med}(x) = median(x_1, ..., x_n)$ on bounded interval $D = [0, \Lambda]$.

- Worst case: $GS_{f_{med}} = \Lambda$.
- On 'typical' inputs f_{med} is not very sensitive: $LS_{f_{med}} = max\{x_m - x_{m-1}, x_{m+1} - x_m\}$.
- Would like noise magnitude to be proportional $LS_{f(x)}$. However the noise level can reveal information:
  - Consider case: $f_{med}(x) = 0$ and $f_{med}(y) = \Lambda$ s.t. $d(x, y) = 1$.
  - In the first case the probability to get non-zero median is exactly 0.
  - In the second case the probability to get non-zero median is > 0.
  - No differential privacy: $Pr[y \in S] > Pr[x \in S]$ where S is the event "getting non zero median".

Local Sensitivity Example

Example

$f_{med}(x) = median(x_1, ..., x_n)$ on bounded interval $D = [0, \Lambda]$.

- Worst case: $GS_{f_{med}} = \Lambda$.
- On 'typical' inputs f_{med} is not very sensitive: $LS_{f_{med}} = max\{x_m - x_{m-1}, x_{m+1} - x_m\}$.
- Would like noise magnitude to be proportional $LS_{f(x)}$. However the noise level can reveal information:
  - Consider case: $f_{med}(x) = 0$ and $f_{med}(y) = \Lambda$ s.t. $d(x, y) = 1$.
  - In the first case the probability to get non-zero median is exactly 0.
  - In the second case the probability to get non-zero median is > 0.
  - No differential privacy: $Pr[y \in S] > Pr[x \in S]$ where S is the event "getting non zero median".

Local Sensitivity Example

Title
Introduction
Instance Based Additive Noise
Computing Smooth Sensitivity
Sample Aggregate Framework
Conclusions

Example

$f_{med}(x) = median(x_1, ..., x_n)$ on bounded interval $D = [0, \Lambda]$.

- Worst case: $GS_{f_{med}} = \Lambda$.
- On 'typical' inputs f_{med} is not very sensitive: $LS_{f_{med}} = max\{x_m - x_{m-1}, x_{m+1} - x_m\}$.
- Would like noise magnitude to be proportional $LS_{f(x)}$. However the noise level can reveal information:
  - Consider case: $f_{med}(x) = 0$ and $f_{med}(y) = \Lambda$ s.t. $d(x, y) = 1$.
  - In the first case the probability to get non-zero median is exactly 0.
  - In the second case the probability to get non-zero median is > 0.
  - No differential privacy: $Pr[y \in S] > Pr[x \in S]$ where S is the event "getting non zero median".

Local Sensitivity Example

Example

$f_{med}(x) = median(x_1, ..., x_n)$ on bounded interval $D = [0, \Lambda]$.

- Worst case: $GS_{f_{med}} = \Lambda$.
- On 'typical' inputs f_{med} is not very sensitive: $LS_{f_{med}} = max\{x_m - x_{m-1}, x_{m+1} - x_m\}$.
- Would like noise magnitude to be proportional $LS_{f(x)}$. However the noise level can reveal information:
  - Consider case: $f_{med}(x) = 0$ and $f_{med}(y) = \Lambda$ s.t. $d(x, y) = 1$.
  - In the first case the probability to get non-zero median is exactly 0.
  - In the second case the probability to get non-zero median is > 0.
  - No differential privacy: $Pr[y \in S] > Pr[x \in S]$ where S is the event "getting non zero median".

Local Sensitivity Example

Example

$f_{med}(x) = median(x_1, ..., x_n)$ on bounded interval $D = [0, \Lambda]$.

- Worst case: $GS_{f_{med}} = \Lambda$.
- On 'typical' inputs f_{med} is not very sensitive: $LS_{f_{med}} = max\{x_m - x_{m-1}, x_{m+1} - x_m\}$.
- Would like noise magnitude to be proportional $LS_{f(x)}$. However the noise level can reveal information:
  - Consider case: $f_{med}(x) = 0$ and $f_{med}(y) = \Lambda$ s.t. $d(x, y) = 1$.
  - In the first case the probability to get non-zero median is exactly 0.
  - In the second case the probability to get non-zero median is > 0.
  - No differential privacy: $Pr[y \in S] > Pr[x \in S]$ where S is the event "getting non zero median".

Local Sensitivity Example

Example

$f_{med}(x) = median(x_1, ..., x_n)$ on bounded interval $D = [0, \Lambda]$.

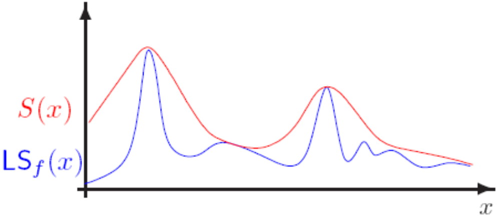
- Worst case: $GS_{f_{med}} = \Lambda$.
- On 'typical' inputs f_{med} is not very sensitive: $LS_{f_{med}} = max\{x_m - x_{m-1}, x_{m+1} - x_m\}$.
- Would like noise magnitude to be proportional $LS_{f(x)}$. However the noise level can reveal information:
  - Consider case: $f_{med}(x) = 0$ and $f_{med}(y) = \Lambda$ s.t. $d(x, y) = 1$.
  - In the first case the probability to get non-zero median is exactly 0.
  - In the second case the probability to get non-zero median is > 0.
  - No differential privacy: $Pr[y \in S] > Pr[x \in S]$ where S is the event "getting non zero median".

- The problem: the noise magnitude is sensitive.
- The solution : The noise magnitude should be insensitive too!

- The problem: the noise magnitude is sensitive.
- The solution : The noise magnitude should be insensitive too!

Smooth Bound

Consider the following function:



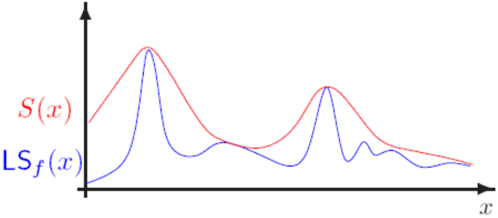
Definition (Smooth Bound)

For $\beta > 0$, a function $S : D^n \to \mathbb{R}^+$ is a *β-smooth upper bound* on the local sensitivity of f if it satisfies the following requirements:

$$\forall x \in D^n: \quad S(x) \geq LS_f(x)$$
$$\forall x, y \in D^n, d(x,y) = 1: \quad S(x) \leq e^{\beta} S(y)$$

Smooth Bound

Consider the following function:



Definition (Smooth Bound)

For $\beta > 0$, a function $S : D^n \to \mathbb{R}^+$ is a *β-smooth upper bound* on the local sensitivity of f if it satisfies the following requirements:

$$\forall x \in D^n: \quad S(x) \geq LS_f(x)$$
$$\forall x, y \in D^n, d(x,y) = 1: \quad S(x) \leq e^{\beta} S(y)$$

Calibrating Noise to Smooth Bounds

Adding noise proportional to $S_f(x)/\alpha$, where α is a noise parameter and and S_f is a β smooth upper bound on local sensitivity of f yields a secure output.

Calibrating Noise to Smooth Bounds

Definition (Admissible Noise Distribution)

A probability distribution h on $\mathbb{R}^d$ is (α, β)-admissible if, for $\alpha = \alpha(\epsilon, \delta)$, $\beta = \beta(\epsilon, \delta)$, the following two conditions hold for all $\| \Delta \| \leq \alpha$ and $|\lambda| \leq \beta$ and for all subsets $S \subseteq \mathbb{R}^d$:

- *Sliding Property*:
  $\Pr_{Z \sim h}[Z \in S] \leq e^{\frac{\epsilon}{2}} \Pr_{Z \sim h}[Z \in S + \Delta] + \frac{\delta}{2}$
- *Dilation Property*:
  $\Pr_{Z \sim h}[Z \in S] \leq e^{\frac{\epsilon}{2}} \Pr_{Z \sim h}[Z \in e^{\lambda} \cdot S] + \frac{\delta}{2}$

Example

Let $h(z) \propto \frac{1}{1+|z|^\gamma}$ for $\gamma > 1$. These $h(x)$ are $(\frac{\epsilon}{4\gamma}, \frac{\epsilon}{\gamma})$-admissible, and yields $\delta = 0$.

Example (Laplace Distribution)

Let $h(z) \propto \frac{1}{2} \cdot e^{-|z|}$ is $(\frac{\epsilon}{2}, \frac{\epsilon}{2 \ln 1/\delta})$-admissible.

Example

Let $h(z) \propto \frac{1}{1+|z|^\gamma}$ for $\gamma > 1$. These $h(x)$ are $(\frac{\epsilon}{4\gamma}, \frac{\epsilon}{\gamma})$-admissible, and yields $\delta = 0$.

Example (Laplace Distribution)

Let $h(z) \propto \frac{1}{2} \cdot e^{-|z|}$ is $(\frac{\epsilon}{2}, \frac{\epsilon}{2 \ln 1/\delta})$-admissible.

Noise Distribution:

Theorem

- *Let Z be a fresh random variable sampled according to (α, β)-admissible noise probability distribution.*
- *For a functionf $: D^n \to \mathbb{R}^d$ let $S : D^n \to \mathbb{R}$ be a β-smooth upper bound on the local sensitivity of f*

Then the database access mechanism:

$$\mathcal{A}(x) = f(x) + \frac{S(x)}{\alpha} \cdot Z$$

is (ϵ, δ)-indistinguishable.

Noise Distribution:

Theorem

- *Let Z be a fresh random variable sampled according to (α, β)-admissible noise probability distribution.*
- *For a function$f : D^n \to \mathbb{R}^d$ let $S : D^n \to \mathbb{R}$ be a β-smooth upper bound on the local sensitivity of f*

Then the database access mechanism:

$$\mathcal{A}(x) = f(x) + \frac{S(x)}{\alpha} \cdot Z$$

is (ϵ, δ)-indistinguishable.

Noise Distribution:

Theorem

- *Let Z be a fresh random variable sampled according to* (α, β)*-admissible noise probability distribution.*
- *For a functionf* $: D^n \rightarrow \mathbb{R}^d$ *let S* $: D^n \rightarrow \mathbb{R}$ *be a* β*-smooth upper bound on the local sensitivity of f*

Then the database access mechanism:

$$\mathcal{A}(x) = f(x) + \frac{S(x)}{\alpha} \cdot Z$$

is (ϵ, δ)*-indistinguishable.*

Proof.

On two neighbor databases x and y, the output distribution $\mathcal{A}(y)$ is a shifted and scaled version of $\mathcal{A}(x)$. The sliding and dilation properties ensure that $\Pr[\mathcal{A}(y) \in S]$ and $\Pr[\mathcal{A}(x) \in S]$ are close for all sets S of outputs. $\square$

How to calculate it ?

Smooth Sensitivity

Define **smooth sensitivity of f**:

Definition (Smooth Sensitivity)

For $\beta > 0$, the β-smooth sensitivity of f is

$$S^*_{f,\beta}(x) = \max_{y \in D^n}(LS_f(y) \cdot e^{-\beta d(x,y)})$$

This function is is an **optimal β-smooth upper bound.**

- Some generic observations:
  - Define sensitivity of f as k entries of x are modified:

Definition

The sensitivity of f at distance k is

$$A^{(k)}(x) = \max_{y \in D^n : d(x,y) \le k} (LS_f(y))$$

  - Smooth sensitivity in term of $A^{(k)}(x)$:

$$S^*_{f,\epsilon}(x) = \max_{k=0,1,2,\ldots,n} e^{-k\epsilon} \left(\max_{y \in D^n : d(x,y) = k} LS_f(y) \right) \Rightarrow$$

$$S^*_{f,\epsilon}(x) = \max_{k=0,1,2,\ldots,n} e^{-k\epsilon} A^{(k)}(x)$$

  - Focus our attention to $A^{(k)}(x)$.

- Some generic observations:
  - Define sensitivity of f as k entries of x are modified:

Definition

The sensitivity of f at distance k is

$$A^{(k)}(x) = \max_{y \in D^n : d(x,y) \le k} (LS_f(y))$$

- Smooth sensitivity in term of $A^{(k)}(x)$:

$$S^*_{f,\epsilon}(x) = \max_{k=0,1,2,\ldots,n} e^{-k\epsilon} \left(\max_{y \in D^n : d(x,y)=k} LS_f(y) \right) \Rightarrow$$

$$S^*_{f,\epsilon}(x) = \max_{k=0,1,2,\ldots,n} e^{-k\epsilon} A^{(k)}(x)$$

- Focus our attention to $A^{(k)}(x)$.

- Some generic observations:
  - Define sensitivity of f as k entries of x are modified:

Definition

The sensitivity of f at distance k is

$$A^{(k)}(x) = \max_{y \in D^n : d(x,y) \leq k} (LS_f(y))$$

  - Smooth sensitivity in term of $A^{(k)}(x)$:

$$S^*_{f,\epsilon}(x) = \max_{k=0,1,2,\ldots,n} e^{-k\epsilon} \left(\max_{y \in D^n : d(x,y)=k} LS_f(y) \right) \Rightarrow$$

$$S^*_{f,\epsilon}(x) = \max_{k=0,1,2,\ldots,n} e^{-k\epsilon} A^{(k)}(x)$$

  - Focus our attention to $A^{(k)}(x)$.

- Some generic observations:
  - Define sensitivity of f as k entries of x are modified:

Definition

The sensitivity of f at distance k is

$$A^{(k)}(x) = \max_{y \in D^n : d(x,y) \leq k} (LS_f(y))$$

  - Smooth sensitivity in term of $A^{(k)}(x)$:

$$S^*_{f,\epsilon}(x) = \max_{k=0,1,2,\ldots,n} e^{-k\epsilon} \left(\max_{y \in D^n : d(x,y) = k} LS_f(y) \right) \Rightarrow$$

$$S^*_{f,\epsilon}(x) = \max_{k=0,1,2,\ldots,n} e^{-k\epsilon} A^{(k)}(x)$$

  - Focus our attention to $A^{(k)}(x)$.

- Some generic observations:
  - Define sensitivity of f as k entries of x are modified:

Definition

The sensitivity of f at distance k is

$$A^{(k)}(x) = \max_{y \in D^n : d(x,y) \leq k} (LS_f(y))$$

  - Smooth sensitivity in term of $A^{(k)}(x)$:

$$S^*_{f,\epsilon}(x) = \max_{k=0,1,2,\ldots,n} e^{-k\epsilon} \left(\max_{y \in D^n : d(x,y)=k} LS_f(y) \right) \Rightarrow$$

$$S^*_{f,\epsilon}(x) = \max_{k=0,1,2,\ldots,n} e^{-k\epsilon} A^{(k)}(x)$$

  - Focus our attention to $A^{(k)}(x)$.

Example

Median.

A Motivating Example: Clustering

- The goal: privately releasing k-means cluster centers.
- Considering $k - squared - error - distribution$ (k-SED) clustring:
  - Input: set of points $x_1, x_2, ..., x_n \in \mathbb{R}^l$.
  - Output: k centers $c_1, c_2, ..., c_k$ with minimum cost.
  - The cost: $cost_x(c_1, c_2, ..., c_k) = \frac{1}{n}\sum_{i=1}^{n} \min_j \parallel x_i - c_j \parallel_2^2$
- Need to compute distance for sensitivity framework.
- L_2 norm is not good. two permutations of the centers might be far apart.

Wesserstein Distance (earthmover metric):

$$d_W(\{c_1, ..., c_k\}, \{\hat{c}_1, ..., \hat{c}_k\}) = \left(\min_{\pi \in S_k} \sum_{j=1}^{k} \parallel c_j - \hat{c}_{\pi(j)} \parallel_2^2\right)^{\frac{1}{2}}$$

A Motivating Example: Clustering

- The goal: privately releasing k-means cluster centers.
- Considering $k - squared - error - distribution$ (k-SED) clustring:
  - Input: set of points $x_1, x_2, ..., x_n \in \mathbb{R}^l$.
  - Output: k centers $c_1, c_2, ..., c_k$ with minimum cost.
  - The cost: $cost_x(c_1, c_2, ..., c_k) = \frac{1}{n}\sum_{i=1}^{n} \min_j \| x_i - c_j \|_2^2$
- Need to compute distance for sensitivity framework.
- L_2 norm is not good. two permutations of the centers might be far apart.

Wesserstein Distance (earthmover metric):

$$d_W(\{c_1, ..., c_k\}, \{\hat{c}_1, ..., \hat{c}_k\}) = \left(\min_{\pi \in S_k} \sum_{j=1}^{k} \| c_j - \hat{c}_{\pi(j)} \|_2^2 \right)^{\frac{1}{2}}$$

A Motivating Example: Clustering

- The goal: privately releasing k-means cluster centers.
- Considering $k - squared - error - distribution$ (k-SED) clustring:
  - Input: set of points $x_1, x_2, ..., x_n \in \mathbb{R}^l$.
  - Output: k centers $c_1, c_2, ..., c_k$ with minimum cost.
  - The cost: $cost_x(c_1, c_2, ..., c_k) = \frac{1}{n}\sum_{i=1}^{n} \min_j \| x_i - c_j \|_2^2$
- Need to compute distance for sensitivity framework.
- L_2 norm is not good. two permutations of the centers might be far apart.

Wesserstein Distance (earthmover metric):

$$d_W(\{c_1, ..., c_k\}, \{\widehat{c}_1, ..., \widehat{c}_k\}) = \left(\min_{\pi \in S_k} \sum_{j=1}^{k} \| c_j - \widehat{c}_{\pi(j)} \|_2^2 \right)^{\frac{1}{2}}$$

A Motivating Example: Clustering

- The goal: privately releasing k-means cluster centers.
- Considering $k - squared - error - distribution$ (k-SED) clustring:
  - Input: set of points $x_1, x_2, ..., x_n \in \mathbb{R}^l$.
  - Output: k centers $c_1, c_2, ..., c_k$ with minimum cost.
  - The cost: $cost_x(c_1, c_2, ..., c_k) = \frac{1}{n}\sum_{i=1}^{n} \min_j \parallel x_i - c_j \parallel_2^2$
- Need to compute distance for sensitivity framework.
- L_2 norm is not good. two permutations of the centers might be far apart.

Wesserstein Distance (earthmover metric):

$$d_W(\{c_1, ..., c_k\}, \{\widehat{c}_1, ..., \widehat{c}_k\}) = \left(\min_{\pi \in S_k} \sum_{j=1}^{k} \parallel c_j - \widehat{c}_{\pi(j)} \parallel_2^2\right)^{\frac{1}{2}}$$

A Motivating Example: Clustering

- The goal: privately releasing k-means cluster centers.
- Considering $k - squared - error - distribution$ (k-SED) clustring:
  - Input: set of points $x_1, x_2, ..., x_n \in \mathbb{R}^l$.
  - Output: k centers $c_1, c_2, ..., c_k$ with minimum cost.
  - The cost: $cost_x(c_1, c_2, ..., c_k) = \frac{1}{n} \sum_{i=1}^{n} \min_j \| x_i - c_j \|_2^2$
- Need to compute distance for sensitivity framework.
- L_2 norm is not good. two permutations of the centers might be far apart.

Wesserstein Distance (earthmover metric):

$$d_W(\{c_1, ..., c_k\}, \{\widehat{c}_1, ..., \widehat{c}_k\}) = \left(\min_{\pi \in S_k} \sum_{j=1}^{k} \| c_j - \widehat{c}_{\pi(j)} \|_2^2 \right)^{\frac{1}{2}}$$

A Motivating Example: Clustering

- The goal: privately releasing k-means cluster centers.
- Considering $k - \textit{squared} - \textit{error} - \textit{distribution}$ (k-SED) clustring:
  - Input: set of points $x_1, x_2, ..., x_n \in \mathbb{R}^l$.
  - Output: k centers $c_1, c_2, ..., c_k$ with minimum cost.
  - The cost: $cost_x(c_1, c_2, ..., c_k) = \frac{1}{n}\sum_{i=1}^{n} \min_j \parallel x_i - c_j \parallel_2^2$
- Need to compute distance for sensitivity framework.
- L_2 norm is not good. two permutations of the centers might be far apart.

Wesserstein Distance (earthmover metric):

$$d_W(\{c_1, ..., c_k\}, \{\widehat{c}_1, ..., \widehat{c}_k\}) = \left(\min_{\pi \in S_k} \sum_{j=1}^{k} \parallel c_j - \widehat{c}_{\pi(j)} \parallel_2^2\right)^{\frac{1}{2}}$$

A Motivating Example: Clustering

- The goal: privately releasing k-means cluster centers.
- Considering $k - squared - error - distribution$ (k-SED) clustring:
  - Input: set of points $x_1, x_2, ..., x_n \in \mathbb{R}^l$.
  - Output: k centers $c_1, c_2, ..., c_k$ with minimum cost.
  - The cost: $cost_x(c_1, c_2, ..., c_k) = \frac{1}{n} \sum_{i=1}^{n} \min_j \| x_i - c_j \|_2^2$
- Need to compute distance for sensitivity framework.
- L_2 norm is not good. two permutations of the centers might be far apart.

Wesserstein Distance (earthmover metric):

$$d_W(\{c_1, ..., c_k\}, \{\widehat{c}_1, ..., \widehat{c}_k\}) = \left(\min_{\pi \in S_k} \sum_{j=1}^{k} \| c_j - \widehat{c}_{\pi(j)} \|_2^2 \right)^{\frac{1}{2}}$$

A Motivating Example: Clustering

- The goal: privately releasing k-means cluster centers.
- Considering $k - squared - error - distribution$ (k-SED) clustring:
  - Input: set of points $x_1, x_2, ..., x_n \in \mathbb{R}^l$.
  - Output: k centers $c_1, c_2, ..., c_k$ with minimum cost.
  - The cost: $cost_x(c_1, c_2, ..., c_k) = \frac{1}{n} \sum_{i=1}^{n} \min_j \| x_i - c_j \|_2^2$
- Need to compute distance for sensitivity framework.
- L_2 norm is not good. two permutations of the centers might be far apart.

Wesserstein Distance (earthmover metric):

$$d_W(\{c_1, ..., c_k\}, \{\widehat{c}_1, ..., \widehat{c}_k\}) = \left(\min_{\pi \in S_k} \sum_{j=1}^{k} \| c_j - \widehat{c}_{\pi(j)} \|_2^2 \right)^{\frac{1}{2}}$$

- The output space of algorithm $\mathcal{M}$ is $(\mathbb{R}^l)^k$.
- Computing the Wesserstein distance is efficient: maximum matching in a bipartite graph.
- Add noise with respect to L_2^{lk} norm. L_2 distance is an upper bound on the Wesserstein distance.
- Compute sensitivity w.r.t. Wesserstein distance, but add noise w.r.t. L_2.

- The output space of algorithm $\mathcal{M}$ is $(\mathbb{R}^l)^k$.
- Computing the Wesserstein distance is efficient: maximum matching in a bipartite graph.
- Add noise with respect to L_2^{lk} norm. L_2 distance is an upper bound on the Wesserstein distance.
- Compute sensitivity w.r.t. Wesserstein distance, but add noise w.r.t. L_2.

- The output space of algorithm $\mathcal{M}$ is $(\mathbb{R}^l)^k$.
- Computing the Wesserstein distance is efficient: maximum matching in a bipartite graph.
- Add noise with respect to L_2^{lk} norm. L_2 distance is an upper bound on the Wesserstein distance.
- Compute sensitivity w.r.t. Wesserstein distance, but add noise w.r.t. L_2.

- The output space of algorithm $\mathcal{M}$ is $(\mathbb{R}^l)^k$.
- Computing the Wesserstein distance is efficient: maximum matching in a bipartite graph.
- Add noise with respect to L_2^{lk} norm. L_2 distance is an upper bound on the Wesserstein distance.
- Compute sensitivity w.r.t. Wesserstein distance, but add noise w.r.t. L_2.

Sensitivity of Clustering

- Denote by $f_{cc}(x)$ the k-SED cluster centers. Assume $Diam(x) = \Lambda$.
- The cost function has global sensitivity of at most $\frac{\Lambda}{n}$.
- The global sensitivity of $f_{cc}(x)$ is much higher: $\Omega(\Lambda)$. See figure below.
- Adding noise to $f_{cc}(x)$ according to global sensitivity erases centers completely.
- Intuition: Local sensitivity should be low since moving a few data points should not change the centers significantly.
- Do not know how to smooth bound $LS_{f_{cc}}$.

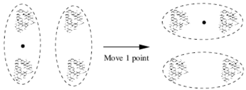


Figure 1: A sensitive 2-SED instance

Basic Framework

The Settings:

- $\mathcal{M}$ a metric space with distance function $d_{\mathcal{M}}(\cdot, \cdot)$ with diameter Λ.
- f is defined on databases with variable size.
- For a particular input $x \in D^n$ the function value $f(x)$ can be **approximated well** by evaluating f on $o(n)$ random sample.

Basic Framework

The Settings:

- $\mathcal{M}$ a metric space with distance function $d_{\mathcal{M}}(\cdot, \cdot)$ with diameter Λ.
- f is defined on databases with variable size.
- For a particular input $x \in D^n$ the function value $f(x)$ can be **approximated well** by evaluating f on $o(n)$ random sample.

Basic Framework

The Settings:

- $\mathcal{M}$ a metric space with distance function $d_{\mathcal{M}}(\cdot,\cdot)$ with diameter Λ.
- f is defined on databases with variable size.
- For a particular input $x \in D^n$ the function value $f(x)$ can be **approximated well** by evaluating f on $o(n)$ random sample.

Basic Framework

The framework:

- Randomly partition the database into m small databases equal sized.
- Let $U_1, U_2, \dots, U_m$ be random subsets of size $\frac{n}{m}$ selected from 1,...,n with no replacement.
- Denote by $x|_U$ the subset of x with indices in U.
- Evaluate $f(x|_{U_1}), \dots, f(x|_{U_n})$ denote result as $z_1, \dots, z_n$.
- output $\bar{f}(x) = g(z_1, \dots, z_n)$. Where g is the aggregation function called **center-of-attention**.

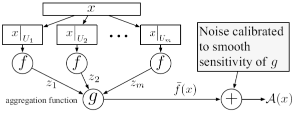


Figure 2: The Sample-Aggregate Framework

The Center of Attention:

- Aggregation the results of approximation function on the database.
- Properties:
  - Smooth upper bounded.
  - Add little noise.
  - Give solution that close to the real function calculated.
- Focus on small balls centered at point in the input set.
- Even if we take out s points the majority of the points will be inside of the ball.
- Hence the solution will not change much after changing some points.

The Center of Attention:

- Aggregation the results of approximation function on the database.
- Properties:
  - Smooth upper bounded.
  - Add little noise.
  - Give solution that close to the real function calculated.
- Focus on small balls centered at point in the input set.
- Even if we take out s points the majority of the points will be inside of the ball.
- Hence the solution will not change much after changing some points.

The Center of Attention

The Center of Attention:

- Aggregation the results of approximation function on the database.
- Properties:
  - Smooth upper bounded.
  - Add little noise.
  - Give solution that close to the real function calculated.
- Focus on small balls centered at point in the input set.
- Even if we take out s points the majority of the points will be inside of the ball.
- Hence the solution will not change much after changing some points.

The Center of Attention

The Center of Attention:

- Aggregation the results of approximation function on the database.
- Properties:
  - Smooth upper bounded.
  - Add little noise.
  - Give solution that close to the real function calculated.
- Focus on small balls centered at point in the input set.
- Even if we take out s points the majority of the points will be inside of the ball.
- Hence the solution will not change much after changing some points.

The Center of Attention

The Center of Attention:

- Aggregation the results of approximation function on the database.
- Properties:
  - Smooth upper bounded.
  - Add little noise.
  - Give solution that close to the real function calculated.
- Focus on small balls centered at point in the input set.
- Even if we take out s points the majority of the points will be inside of the ball.
- Hence the solution will not change much after changing some points.

The Center of Attention

The Center of Attention:

- Aggregation the results of approximation function on the database.
- Properties:
  - Smooth upper bounded.
  - Add little noise.
  - Give solution that close to the real function calculated.
- Focus on small balls centered at point in the input set.
- Even if we take out s points the majority of the points will be inside of the ball.
- Hence the solution will not change much after changing some points.

The Center of Attention

The Center of Attention:

- Aggregation the results of approximation function on the database.
- Properties:
  - Smooth upper bounded.
  - Add little noise.
  - Give solution that close to the real function calculated.
- Focus on small balls centered at point in the input set.
- Even if we take out s points the majority of the points will be inside of the ball.
- Hence the solution will not change much after changing some points.

The Center of Attention

The Center of Attention:

- Aggregation the results of approximation function on the database.
- Properties:
  - Smooth upper bounded.
  - Add little noise.
  - Give solution that close to the real function calculated.
- Focus on small balls centered at point in the input set.
- Even if we take out s points the majority of the points will be inside of the ball.
- Hence the solution will not change much after changing some points.

The Center of Attention

The Center of Attention:

- Aggregation the results of approximation function on the database.
- Properties:
  - Smooth upper bounded.
  - Add little noise.
  - Give solution that close to the real function calculated.
- Focus on small balls centered at point in the input set.
- Even if we take out s points the majority of the points will be inside of the ball.
- Hence the solution will not change much after changing some points.

- The main idea: Changing one point in the database will change very few small databases.
- $\bar{f}(x)$ should have the following properties:
  - if most of the z_i's are close to some point, then $\bar{f}(x)$ should be close to that point.
  - We can efficiently compute a smooth upper bound on the local sensitivity of $\bar{f}(x)$.
- But what is **approximated well** ?

Definition

A function $f : D^* \rightarrow \mathcal{M}$ is approximated to within accuracy r on the input x using samples of size n' if

$$\Pr_{U \subset [n], |U| = n'} [d_{\mathcal{M}}(f(x|_U), f(x)) \leq r] \geq \frac{3}{4}$$

Theorem (Main)

Let $f : D^* \rightarrow \mathbb{R}^d$ be an efficient computable function with range of diameter Λ and L_1 metric on the output space. Set $\epsilon > \frac{2d}{\sqrt{m}}$ and $m = w(\log^2 n)$. The sample-aggregate mechanism $\mathcal{A}$ is an ϵ-indistinguishable efficient mechanism. Moreover. if f is approximated within accuracy r on the database $x = (x_1, ..., x_n)$ using sample size $\frac{n}{m}$, then each coordinate of the random variable $\mathcal{A}(x) - f(x)$ has expected magnitude of $O(\frac{r}{\epsilon}) + \frac{\Lambda}{\epsilon} e^{-\Omega(\frac{\epsilon\sqrt{m}}{d})}$.

Corollary

Suppose that ϵ is constant. If f is approximated within accuracy r on input x using sample of size $o(\frac{n}{d^2}\log^2 n)$, then $\mathcal{A}$ releases $f(x)$ with expected error $O(r) + \Lambda \cdot neg(\frac{n}{d})$ in each coordinate.

Aggregation for General Metric Spaces

Good Aggregation

- No point has a probability of at least $1 - 2^{-\sqrt{m}+\log n}$ probability to effect more then $\sqrt{m}$ small databases.
- Therefore we will focus on generalization of the local sensitivity:

Definition

For $g : D^m \to \mathcal{M}$ and $z \in D^m$, the local sensitivity of g at x with step s is:

$$LS_g^{(s)}(z) = \max_{z' : d(z,z') \leq s} d_{\mathcal{M}}(g(z), g(z'))$$

Aggregation for General Metric Spaces

- The definition of β-smooth upper bound has to be changed too:

Definition

For $\beta > 0$ a function $S : D^m \to \mathbb{R}^+$ is a β-smooth upper bound on the sensitivity of g with step size s if

$$\forall z \in D^m: \quad S(z) \geq LS_g^{(s)}(z)$$

$$\forall z, z' \in D^m, d(z, z') \leq s: \quad S(z) \leq e^{\beta} S(z')$$

Good Aggregation

Definition (Good Aggregation)

In a metric space $\mathcal{M}$ with diameter Λ, an (m, β, s)-aggregation is a pair of functions, an aggregation function $g : \mathcal{M}^m \rightarrow \mathcal{M}$ and a sensitivity function function $S : \mathcal{M}^m \rightarrow \mathbb{R}^+$, such that

1. S is a β-smooth upper bound on $LS_g^{(s)}$.
2. If at least $\frac{2m}{3}$ entries in z are in some ball $\mathcal{B}(c, r)$ then
   (a) $g(z) \in \mathcal{B}(c, O(r))$
   (b) $S(z) = O(r) + \Lambda \cdot e^{-\Omega(\beta m/s)}$

Good Aggregation

Definition (Good Aggregation)

Let $g_0(z) \in \mathcal{M}$ be a point with minimum t_0-radius, where $t_0 = (\frac{m+s}{2} + 1)$, and let $S_0(z) = 2\max_{k\geq 0}(r^z(t_0 + (k+1)s)e^{-\beta k})$. Then the pair (g_0, S_0) is a good aggregation.

What Did We Cover ?

- The server adds noise $f(x) + N(x)Z$, where $N(x)$ scale-up factor (noise magnitude), $Z \sim NoiseDist(D^n)$ with $\sigma(Z) = 1$.
- Noise magnitude is proportional to global sensitivity. Independent of x.
- Drawbacks:
  - Noise magnitude can be too large, effecting accuracy.
  - Does not use the properties of x.
- Use Local Sensitivity. But can be sensitive too!

What Did We Cover ?

- The server adds noise $f(x) + N(x)Z$, where $N(x)$ scale-up factor (noise magnitude), $Z \sim NoiseDist(D^n)$ with $\sigma(Z) = 1$.
- Noise magnitude is proportional to global sensitivity. Independent of x.
- Drawbacks:
  - Noise magnitude can be too large, effecting accuracy.
  - Does not use the properties of x.
- Use Local Sensitivity. But can be sensitive too!

What Did We Cover ?

- The server adds noise $f(x) + N(x)Z$, where $N(x)$ scale-up factor (noise magnitude), $Z \sim NoiseDist(D^n)$ with $\sigma(Z) = 1$.
- Noise magnitude is proportional to global sensitivity. Independent of x.
- Drawbacks:
  - Noise magnitude can be too large, effecting accuracy.
  - Does not use the properties of x.
- Use Local Sensitivity. But can be sensitive too!

What Did We Cover ?

- The server adds noise $f(x) + N(x)Z$, where $N(x)$ scale-up factor (noise magnitude), $Z \sim NoiseDist(D^n)$ with $\sigma(Z) = 1$.
- Noise magnitude is proportional to global sensitivity. Independent of x.
- Drawbacks:
  - Noise magnitude can be too large, effecting accuracy.
  - Does not use the properties of x.
- Use Local Sensitivity. But can be sensitive too!

What Did We Cover ?

- The server adds noise $f(x) + N(x)Z$, where $N(x)$ scale-up factor (noise magnitude), $Z \sim NoiseDist(D^n)$ with $\sigma(Z) = 1$.
- Noise magnitude is proportional to global sensitivity. Independent of x.
- Drawbacks:
  - Noise magnitude can be too large, effecting accuracy.
  - Does not use the properties of x.
- Use Local Sensitivity. But can be sensitive too!

What Did We Cover ?

- The server adds noise $f(x) + N(x)Z$, where $N(x)$ scale-up factor (noise magnitude), $Z \sim NoiseDist(D^n)$ with $\sigma(Z) = 1$.
- Noise magnitude is proportional to global sensitivity. Independent of x.
- Drawbacks:
  - Noise magnitude can be too large, effecting accuracy.
  - Does not use the properties of x.
- Use Local Sensitivity. But can be sensitive too!

Smooth Upper Bound

- The class of *smooth* upper bounds S_f to LS_f s.t. adding noise proportional to S_f is safe.
- Define special class S_f^* that is optimal in the sense that $S_f^*(x) \leq S_f(x)$ for every other smooth S_f.
- Will show how to calculate the smooth sensitivity for:
  - Median
  - Minimal spanning tree cost

What Did We Cover ?

Smooth Upper Bound

- The class of *smooth* upper bounds S_f to LS_f s.t. adding noise proportional to S_f is safe.
- Define special class S^*_f that is optimal in the sense that $S^*_f(x) \leq S_f(x)$ for every other smooth S_f.
- Will show how to calculate the smooth sensitivity for:
  - Median
  - Minimal spanning tree cost

What Did We Cover ?

Smooth Upper Bound

- The class of *smooth* upper bounds S_f to LS_f s.t. adding noise proportional to S_f is safe.
- Define special class S_f^* that is optimal in the sense that $S_f^*(x) \leq S_f(x)$ for every other smooth S_f.
- Will show how to calculate the smooth sensitivity for:
  - Median
  - Minimal spanning tree cost

What Did We Cover ?

Smooth Upper Bound

- The class of *smooth* upper bounds S_f to LS_f s.t. adding noise proportional to S_f is safe.
- Define special class S_f^* that is optimal in the sense that $S_f^*(x) \leq S_f(x)$ for every other smooth S_f.
- Will show how to calculate the smooth sensitivity for:
  - Median
  - Minimal spanning tree cost

What Did We Cover ?

Smooth Upper Bound

- The class of *smooth* upper bounds S_f to LS_f s.t. adding noise proportional to S_f is safe.
- Define special class S_f^* that is optimal in the sense that $S_f^*(x) \leq S_f(x)$ for every other smooth S_f.
- Will show how to calculate the smooth sensitivity for:
  - Median
  - Minimal spanning tree cost

What Did We Cover ?

Smooth Upper Bound

- The class of *smooth* upper bounds S_f to LS_f s.t. adding noise proportional to S_f is safe.
- Define special class S_f^* that is optimal in the sense that $S_f^*(x) \leq S_f(x)$ for every other smooth S_f.
- Will show how to calculate the smooth sensitivity for:
  - Median
  - Minimal spanning tree cost

What Did We Cover ?

The Sample and Aggregate Framework:

- Replacing f with $\bar{f}$ for which low sensitivity is low and efficiently computable. $\bar{f}$ as smoothed version of f.
- f is evaluated on a sublinear number of random samples from database x.
- Evaluations done several times.
- Results combined with a novel aggregation function called *center of attention*.
- The output denoted as $\bar{f}$ released with the smooth sensitivity framework.
- If $f(x)$ approximated well by evaluation on random samples $\Rightarrow \bar{f}(x)$ is close to $f(x)$.

What Did We Cover ?

The Sample and Aggregate Framework:

- Replacing f with $\bar{f}$ for which low sensitivity is low and efficiently computable. $\bar{f}$ as smoothed version of f.
- f is evaluated on a sublinear number of random samples from database x.
- Evaluations done several times.
- Results combined with a novel aggregation function called *center of attention*.
- The output denoted as $\bar{f}$ released with the smooth sensitivity framework.
- If $f(x)$ approximated well by evaluation on random samples $\Rightarrow \bar{f}(x)$ is close to $f(x)$.

What Did We Cover ?

The Sample and Aggregate Framework:

- Replacing f with $\bar{f}$ for which low sensitivity is low and efficiently computable. $\bar{f}$ as smoothed version of f.
- f is evaluated on a sublinear number of random samples from database x.
- Evaluations done several times.
- Results combined with a novel aggregation function called *center of attention*.
- The output denoted as $\bar{f}$ released with the smooth sensitivity framework.
- If $f(x)$ approximated well by evaluation on random samples $\Rightarrow \bar{f}(x)$ is close to $f(x)$.

What Did We Cover ?

The Sample and Aggregate Framework:

- Replacing f with $\bar{f}$ for which low sensitivity is low and efficiently computable. $\bar{f}$ as smoothed version of f.
- f is evaluated on a sublinear number of random samples from database x.
- Evaluations done several times.
- Results combined with a novel aggregation function called *center of attention*.
- The output denoted as $\bar{f}$ released with the smooth sensitivity framework.
- If $f(x)$ approximated well by evaluation on random samples $\Rightarrow \bar{f}(x)$ is close to $f(x)$.

What Did We Cover ?

The Sample and Aggregate Framework:

- Replacing f with $\bar{f}$ for which low sensitivity is low and efficiently computable. $\bar{f}$ as smoothed version of f.
- f is evaluated on a sublinear number of random samples from database x.
- Evaluations done several times.
- Results combined with a novel aggregation function called *center of attention*.
- The output denoted as $\bar{f}$ released with the smooth sensitivity framework.
- If $f(x)$ approximated well by evaluation on random samples $\Rightarrow \bar{f}(x)$ is close to $f(x)$.

What Did We Cover ?

The Sample and Aggregate Framework:

- Replacing f with $\bar{f}$ for which low sensitivity is low and efficiently computable. $\bar{f}$ as smoothed version of f.
- f is evaluated on a sublinear number of random samples from database x.
- Evaluations done several times.
- Results combined with a novel aggregation function called *center of attention*.
- The output denoted as $\bar{f}$ released with the smooth sensitivity framework.
- If $f(x)$ approximated well by evaluation on random samples $\Rightarrow \bar{f}(x)$ is close to $f(x)$.

What Did We Cover ?

The Sample and Aggregate Framework:

- Replacing f with $\bar{f}$ for which low sensitivity is low and efficiently computable. $\bar{f}$ as smoothed version of f.
- f is evaluated on a sublinear number of random samples from database x.
- Evaluations done several times.
- Results combined with a novel aggregation function called *center of attention*.
- The output denoted as $\bar{f}$ released with the smooth sensitivity framework.
- If $f(x)$ approximated well by evaluation on random samples $\Rightarrow \bar{f}(x)$ is close to $f(x)$.

Questions ?

Thank You !

Questions ?

Thank You !