Differential Privacy in the Streaming Model

Jalaj Upadhyay

David R. Cheriton School of Computer Science
University of Waterloo

April 24, 2015
Why We Need Privacy? Social Graph

A natural question: how many people have friends outside their circle?

The answer is the number of edges crossing the border of the set of the vertices corresponding to those people.
• A natural question: how many people have friends outside their circle?
A natural question: how many people have friends outside their circle?

The answer is the number of edges crossing the border of the set of the vertices corresponding to those people.
Suppose Facebook decides to reveal the relationship graph
There might be some people who might end up in trouble
Friendships or "What You May Call" Between People

Suppose Facebook decides to reveal the relationship graph. There might be some people who might end up in trouble.
Suppose Facebook decides to reveal the relationship graph. There might be some people who might end up in trouble.
Suppose Facebook decides to reveal the relationship graph. There might be some people who might end up in trouble.
Suppose Facebook decides to reveal the relationship graph. There might be some people who might end up in trouble.
Suppose Facebook decides to reveal the relationship graph, there might be some people who might end up in trouble.
Disclaimer

I do not support any of the above infidelity

None of this work should be used in any of the above cited or related scenarios

Mr. Clinton, Mr. Woods, or NSA did not fund this research
Why We Need Privacy? Netflix Competition

Training set: 100,480,507
Qualifying set: 2,817,131
Winners announced in September, 2009
Why We Need Privacy? Netflix Competition

Task: Deanonymize the data

Training set: 100,480,507
Qualifying set: 2,817,131
Why We Need Privacy? Netflix Competition

Task: Deanonymize the data

Training set: 100,480,507
Qualifying set: 2,817,131

Winners announced in September, 2009
**Analyst** wishes to get some task done on **Database**.
Privacy guard provides privacy of an individual in Database.
The privacy guard performs the task on the Database
The idea is that absence or presence of an individual entry should not change the output “by much"
The idea is that absence or presence of an individual entry should not change the output “by much”

**Definition.** A randomized algorithm, $\mathcal{M}$, gives ($\varepsilon, \delta$)-differential privacy if, for all “neighboring data,” $\mathcal{D}$ and $\tilde{\mathcal{D}}$, and for all $S \subseteq \text{Range}(\mathcal{M})$, $\Pr[\mathcal{M}(\mathcal{D}) \in S] \leq \exp(\varepsilon)\Pr[\mathcal{M}(\tilde{\mathcal{D}}) \in S] + \delta$
The idea is that absence or presence of an individual entry should not change the output “by much"

**Definition.** A randomized algorithm, $\mathcal{M}$, gives $(\epsilon, \delta)$-differential privacy if, for all “neighboring data," $D$ and $\tilde{D}$, and for all $S \subseteq \text{Range}(\mathcal{M})$, 

$$\Pr[\mathcal{M}(D) \in S] \leq \exp(\epsilon) \Pr[\mathcal{M}(\tilde{D}) \in S] + \delta$$

We restrict how the privacy guard can access the database
Differentially Private Streaming Model of Computation

Privacy Guard

Private Matrix

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>
Differentially Private Streaming Model of Computation

Privacy Guard

- Operates on the stream

Private Matrix

- Update the data-structure
Differentially Private Streaming Model of Computation

Privacy Guard

- Operates on the stream
- Update the data-structure

$$\begin{bmatrix}
4 & 3 \\
6 & 2 \\
1 & 8
\end{bmatrix}$$

Private Matrix

$$\begin{bmatrix}
6 & 7 \\
2 & 7 \\
1 & 7 \\
9 & 6 \\
1 &
\end{bmatrix}$$
Differentially Private Streaming Model of Computation

Privacy Guard

- Operates on the stream
- Update the data-structure

\[
\begin{pmatrix}
4 & 3 & 4 \\
6 & 2 & 8 \\
1 & 8 & 9
\end{pmatrix}
\]
An analyst comes along
Differentially Private Streaming Model of Computation

An analyst comes along

request to do a task
Differentially Private Streaming Model of Computation

An analyst comes along

request to do a task

- uses
  \begin{pmatrix}
  4 & 3 & 4 \\
  6 & 2 & 8 \\
  1 & 8 & 9 \\
  \end{pmatrix}
An analyst comes along

request to do a task

performs the task

• uses

\[
\begin{pmatrix}
4 & 3 & 4 \\
6 & 2 & 8 \\
1 & 8 & 9 \\
\end{pmatrix}
\]
cannot figure out individual information
Differentially Private Streaming Model of Computation

cannot figure out individual information

Privacy goal achieved
Following are the extra parameters

1. **number of passes** over the matrix
2. space requirement of the data-structures
3. time required to update the data-structures
Following are the extra parameters

1. number of passes over the matrix
2. space requirement of the data-structures
3. time required to update the data-structures
Following are the extra parameters

1. number of passes over the matrix
2. space requirement of the data-structures
3. time required to update the data-structures
Following are the extra parameters:

1. number of passes over the matrix
2. space requirement of the data-structures
3. time required to update the data-structures

Two common methods for constructing the data-structure:

1. Random sampling
2. Sketch based approach
Differentially Private Streaming Model of Computation

Following are the extra parameters

1. number of passes over the matrix
2. space requirement of the data-structures
3. time required to update the data-structures

Two common methods for constructing the data-structure:

1. Random sampling
2. Sketch based approach
Non-private Setting
Non-private Setting

Data-structure is a sketch generated using random matrix
Non-private Setting

Data-structure is a sketch generated using random matrix

\[\downarrow\]

Efficient one-pass streaming algorithms
Private Setting
Private Setting

Special distribution of random matrices
Private Setting

Special distribution of random matrices

Sketch generated using a random matrix picked from this distribution
Private Setting

Special distribution of random matrices

+ Sketch generated using a random matrix picked from this distribution

\[ \downarrow \]

Differentially private one-pass streaming algorithms
Private Setting

Special distribution of random matrices

+ Sketch generated using a random matrix picked from this distribution

⇓

Differentially private one-pass streaming algorithms with optimal space data-structure and comparable update time
Outline of the Talk

- Part I covers **space efficient** privacy preserving data-structure
- Part II improves the **update efficiency** of the data-structure
Part I

Space efficient Data-Structures \(^1\)

Main Result

Streaming Private Sketch Generator (PSG$_1$)

- Pick a random Gaussian matrix $\Phi$
- Multiply $\Phi$ to the streamed column

**Theorem.** If the singular values of the streamed matrix to PSG$_1$ algorithm are at least

$$\sigma_1 = \left(4\sqrt{r \log(2/\delta)} \log \left(\frac{r}{\delta}\right)\right) / \varepsilon,$$

then PSG$_1$ preserves $(\varepsilon, \delta)$-differential privacy.

Similar result was shown by [BBDS12] for non-streaming algorithms.
Main Result

Streaming Private Sketch Generator (PSG$_1$)

Pick a random Gaussian matrix $\Phi$
Multiply $\Phi$ to the streamed column

**Theorem.** If the singular values of the streamed matrix to PSG$_1$ algorithm are at least

$$\sigma_1 := \left( \frac{4 \sqrt{r \log(2/\delta) \log(r/\delta)}}{\epsilon} \right),$$

then PSG$_1$ preserves $(\epsilon, \delta)$-differential privacy
Main Result

Streaming Private Sketch Generator ($\text{PSG}_1$)

Pick a random Gaussian matrix $\Phi$
Multiply $\Phi$ to the streamed column

**Theorem.** If the singular values of the streamed matrix to $\text{PSG}_1$ algorithm are at least

$$\sigma_1 := \left(4\sqrt{r \log(2/\delta) \log(r/\delta)}\right)/\varepsilon,$$

then $\text{PSG}_1$ preserves $(\varepsilon, \delta)$-differential privacy

Similar result was shown by [BBDS12] for non-streaming algorithms
Streaming Private Sketch Generator (PSG$_2$)

Pick a random Gaussian matrix $\Phi$
Multiply $\Phi^T \Phi$ to the streamed column

**Theorem.** If the singular values of the streamed matrix to the PSG$_2$ algorithm are at least

$$\sigma_2 := \frac{(4 \log(r/\delta))}{\varepsilon},$$

then PSG$_2$ preserves $(\varepsilon, \delta)$-differential privacy.
Main Result

Streaming Private Sketch Generator (PSG₂)

Pick a random Gaussian matrix $\Phi$
Multiply $\Phi^T \Phi$ to the streamed column

**Theorem.** If the singular values of the streamed matrix to the PSG₂ algorithm are at least
\[ \sigma_2 := \frac{(4r \log(r/\delta))}{\varepsilon}, \]
then PSG₂ preserves $(\varepsilon, \delta)$-differential privacy.
A Meta Algorithm

- Get a stream in the form of column vector
A Meta Algorithm

- Get a stream in the form of column vector
- Perturb the vector to lift the singular values
A Meta Algorithm

- Get a stream in the form of column vector
- Perturb the vector to lift the singular values
- Feed it to PSG$_1$ or PSG$_2$
A Meta Algorithm

- Get a stream in the form of column vector
- Perturb the vector to lift the singular values
- Feed it to \( \text{PSG}_1 \) or \( \text{PSG}_2 \)
- Perform any post-processing
Objective: Given an input matrix $A$, find orthonormal matrix $\Psi$ such that, for $N \in \{F, S\}$

$$\|A - \Psi \Psi^T A\|_N \leq (1 + \alpha) \min_{\text{rank}(A_k) \leq k} \|A - A_k\|_N + \tau$$

with probability $1 - \beta$
Why Do We Care?

structured low-rank approximation

systems and control
- model reduction
- system identification

signal processing
- spectral estimation
- image deblurring

computational mathematics
- approx. GCD
- approx. factorization

machine learning
- recomender systems
- manifold learning
Why Do We Care? Image Processing
Non-private constructions uses the following two steps

- Compute $Y = \Phi A$, where $\Phi$ is a picked using some distribution
- Computes a rank $k$ matrix $B = \Pi_Y A$, where $\Pi_Y$ is the projection matrix corresponding to $Y$
A known private construction [HR12] does the following

- Compute \( Y = \Phi A + G \), where \( \Phi \) is a picked using some distribution

- Computes a rank \( k \) matrix \( B = \Pi_Y A + G' \), where \( \Pi_Y \) is the projection matrix corresponding to \( Y \)

It seems like two applications of \( A \) is necessary
Non-private constructions uses the following two steps:

- Compute $Y = \Phi A$, where $\Phi$ is picked using some distribution.
- Computes a rank $k$ matrix $B = \Pi_Y A$, where $\Pi_Y$ is the projection matrix corresponding to $Y$.

It seems like two applications of $A$ is necessary.

We use linear algebra to emulate the second step by using $\Phi$ and $Y$. 

---

**Standard Prototype**
Private Low-rank Approximation in Single Pass

Computation of $Y$

- Set $w$ to be above the threshold of $\text{PSG}_2$
- Pick a random Gaussian matrix $\Phi$

On receiving the column $A_i$:

- Lift the singular values: $\hat{A}_i = (we_i)$
- Compute the sketch $Y_i = \Phi \hat{A}_i$
Computation of $Y$

- Set $w$ to be above the threshold of $\text{PSG}_2$
- Pick a random Gaussian matrix $\Phi$

On receiving the column $A_i$:

- Lift the singular values: $\hat{A}_i = \begin{pmatrix} \omega e_i \\ A_i \end{pmatrix}$
- Compute the sketch $Y_i = \Phi \hat{A}_i$

$Y = \begin{pmatrix} Y_1 & \cdots & Y_d \end{pmatrix}$
Private Low-rank Approximation in Single Pass

Computation of $Y$

- Set $w$ to be above the threshold of $\text{PSG}_2$
- Pick a random Gaussian matrix $\Phi$

On receiving the column $A_i$:

- Lift the singular values: $\hat{A}_i = \begin{pmatrix} we_i \\ A_i \end{pmatrix}$
- Compute the sketch $Y_i = \Phi \hat{A}_i$

$Y = \begin{pmatrix} Y_1 & \cdots & Y_d \end{pmatrix}$

Privacy follows from $\text{PSG}_1$ as $\sigma_1 \ll w$
Post-processing: Use $Y$ and $\Phi$ to compute $B = \Pi_Y A$

- Let $\Pi_Y = \Psi \Psi^T$
- Find a solution to $B \Psi^T \Phi = \Psi^T Y$
- Compute and publish the required decomposition of $B$
Privacy follows from $\text{PSG}_1$ and $\text{PSG}_2$ as $w = \sigma^2$

Utility proof requires perturbation theory and the Johnson-Lindenstrauss property
Privacy follows from PSG$_1$ and PSG$_2$ as $w = \sigma_2$

Utility proof requires perturbation theory and the Johnson-Lindenstrauss property

Johnson-Lindenstrauss property says that using special choice of random projection matrix, we can project a set of vectors to a lower dimensional space while preserving Euclidean distances
### Comparison of Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Additive noise ($\tau$)</th>
<th># Passes</th>
<th>Norm</th>
</tr>
</thead>
<tbody>
<tr>
<td>[CSS13]</td>
<td>$O(nk/\varepsilon)$</td>
<td>$k$</td>
<td>S</td>
</tr>
<tr>
<td>[HR12]</td>
<td>$\tilde{O}\left(\frac{\sqrt{kn}}{\varepsilon} + \sqrt{\frac{\mu|A|_F}{\varepsilon}}\right)$</td>
<td>2</td>
<td>F</td>
</tr>
<tr>
<td>[HR13]</td>
<td>$\tilde{O}\left(\frac{k^2}{\varepsilon}\sqrt{r\mu + k\log n}\log n\right)$</td>
<td>$k\sqrt{\log \lambda}$</td>
<td>S</td>
</tr>
<tr>
<td>[KT13]</td>
<td>$O\left(dk^3/(\varepsilon\gamma^2)\right)$</td>
<td>$k$</td>
<td>S</td>
</tr>
<tr>
<td>[HP14]</td>
<td>$\tilde{O}\left(\frac{\lambda_1\sqrt{kn\mu \log(n/\gamma) \log \log (n/\gamma)}}{\varepsilon\gamma^{1.5} \lambda_k}\right)$</td>
<td>$k\sqrt{\log \lambda}$</td>
<td>S</td>
</tr>
<tr>
<td>[DTTZ14]</td>
<td>$\tilde{O}(k\sqrt{n/\varepsilon}) + \tilde{O}(\sqrt{k^3 n^{3/2}/\varepsilon^2})$</td>
<td>1</td>
<td>S</td>
</tr>
<tr>
<td>Our result</td>
<td>$O(\sqrt{nk \ln(k/\delta)/\varepsilon})$</td>
<td>1</td>
<td>F</td>
</tr>
<tr>
<td>Our result</td>
<td>$O(\sqrt{nk \ln(k/\delta)/\varepsilon})$</td>
<td>1</td>
<td>S</td>
</tr>
</tbody>
</table>

Comparison with previous works (F stands for Frobenius norm, S stands for Spectral norm)
### Problem

<table>
<thead>
<tr>
<th>Problem</th>
<th>Lower Bound</th>
<th>Our Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiplication</td>
<td>$\Omega \left( d\alpha^{-2} \log(nd) \right)$</td>
<td>$\tilde{O} \left( d\alpha^{-2} \log(nd) \log(1/\beta) \right)$</td>
</tr>
<tr>
<td>Regression</td>
<td>$\Omega \left( d^2 \alpha^{-1} \log(nd) \right)$</td>
<td>$\tilde{O} \left( d^2 \log(nd)^{-1} \kappa \log(1/\beta) \right)$</td>
</tr>
<tr>
<td>Low-rank (F)</td>
<td>$\Omega(\kappa \alpha^{-1} (n + d) \log(nd))$</td>
<td>$O(k \epsilon^{-1} (n + d) \log(nd))$</td>
</tr>
<tr>
<td>Low-rank (S)</td>
<td>$-$</td>
<td>$O(k \epsilon^{-1} (n + d) \log(nd))$</td>
</tr>
</tbody>
</table>
Part II

Update time and space efficient data-structure \(^2\)

Let \( n = 1 \text{ billion} = 10^9 \), \( d = 10^3 \), \( r = 100 \)
Any of the above algorithm would take \( rnd = 10^{14} \) time
Privacy and Run-time

Let \( n = 1 \text{ billion} = 10^9 \), \( d = 10^3 \), \( r = 100 \)
Any of the above algorithm would take \( r n d = 10^{14} \) time
Construction of $\Phi$

1. Pick $\{g_1, \ldots, g_n\} \sim \mathcal{N}(0, 1)^n$
2. Divide it into $r$ equal blocks of vectors $\Phi_1, \ldots, \Phi_r$.

$$P := \begin{pmatrix}
\Phi_1 & 0^{n/r} & \ldots & 0^{n/r} \\
0^{n/r} & \Phi_2 & \ldots & 0^{n/r} \\
\vdots & \vdots & \ddots & \vdots \\
0^{n/r} & \ldots & 0^{n/r} & \Phi_r
\end{pmatrix}$$

Compute $\Phi = \sqrt{\frac{1}{r}} P \Pi W$, where $W$ is a randomized Hadamard matrix and $\Pi$ is a permutation matrix
Streaming Private Sketch Generator (PSG$_1$)

- Pick $\Phi$
- Multiply $\Phi$ to the streamed column
Streaming Private Sketch Generator (PSG$$_1$$)

Pick $\Phi$
Multiply $\Phi$ to the streamed column

**Theorem.** If the singular values of the streamed matrix to PSG$$_1$$ algorithm are at least
\[
\sigma_1 := \left(4\sqrt{r \log(2/\delta) \log(r/\delta)}\right)/\epsilon,
\]
then PSG$$_1$$ preserves
\[(\epsilon, \delta + \text{negl})\text{-differential privacy}\]
Streaming Private Sketch Generator (PSG\textsubscript{2})

- Pick $\Phi$
- Multiply $\Phi^T \Phi$ to the streamed column

Theorem. If the singular values of the streamed matrix to the PSG\textsubscript{2} algorithm are at least

$$\sigma^2 := \frac{4r \log(\frac{r}{\delta})}{\epsilon},$$

then PSG\textsubscript{2} preserves $(\epsilon, \delta + \text{negl})$-differential privacy.
Streaming Private Sketch Generator (PSG$_2$)

Pick $\Phi$

Multiply $\Phi^T \Phi$ to the streamed column

**Theorem.** If the singular values of the streamed matrix to the PSG$_2$ algorithm are at least

$$\sigma_2 := \frac{(4r \log(r/\delta))}{\varepsilon},$$

then PSG$_2$ preserves

$$(\varepsilon, \delta + \text{negl})$$-differential privacy.
We prove that $\Phi$ satisfies the Johnson-Lindenstrauss Property
We prove that $\Phi$ satisfies the Johnson-Lindenstrauss Property

Hanson-Wright inequality [HW71] controls the quadratic forms in sub-gaussian random variables: for symmetric matrix $A$ and subgaussian random vector $g$, $g^TAg$ is concentrated around its mean

$$\Pr \left[ \left| g^TAg - \mathbb{E}[g^TAg] \right| \geq \eta \right] \leq \exp \left( -\min \left\{ \frac{c_1 \eta^2}{\|A\|^2_F}, \frac{c_1 \eta}{\|A\|} \right\} \right)$$
Few More Results

• Graph sparsification in composition with random projection also preserves differential privacy
• Graph sparsification in composition with random projection also preserves differential privacy
• Differentially private linear regression
Few More Results

- Graph sparsification in composition with random projection also preserves differential privacy
- Differentially private linear regression
- Application in differentially private manifold learning
Few More Results

- Graph sparsification in composition with random projection also preserves differential privacy
- Differentially private linear regression
- Application in differentially private manifold learning
- Known variants of sparse Johnson-Lindenstrauss transform does not preserve privacy
• Graph sparsification in composition with random projection also preserves differential privacy
• Differentially private linear regression
• Application in differentially private manifold learning
• Known variants of sparse Johnson-Lindenstrauss transform does not preserve privacy
• Simpler proof for many variants of Johnson-Lindenstrauss transform using Hanson-Wright inequality
Few More Results

- Graph sparsification in composition with random projection also preserves differential privacy
- Differentially private linear regression
- Application in differentially private manifold learning
- Known variants of sparse Johnson-Lindenstrauss transform does not preserve privacy
- Simpler proof for many variants of Johnson-Lindenstrauss transform using Hanson-Wright inequality
- Improved bound in compressed sensing for small number of measurements
Future Works

Three closely related open problems:

- Improve the bound for sparse random matrix $\Phi$ for large values of $r$
- Better understand low dimension embedding for any normed space other than $\ell_2$-norm
- Investigate more applications in private learning theory
Thank you for your attention
Recall

\[ P := \begin{pmatrix}
\Phi_1 & 0^{n/r} & \cdots & 0^{n/r} \\
0^{n/r} & \Phi_2 & \cdots & 0^{n/r} \\
\vdots & \vdots & \ddots & \vdots \\
0^{n/r} & \cdots & 0^{n/r} & \Phi_r
\end{pmatrix} \]

Construct \( P_1 \) and \( P_2 \) which operates as follows

- On input \( y \), \( P_2 \) feeds
  \[ \sqrt{\frac{n}{t}} \left( y_{(j-1)n/r+1}, \cdots, y_{jn/r} \right) \]
to the \( i \)-th row of \( P_1 \)

- Row-\( i \) of \( P_1 \) is \( \Phi_i \) for \( 1 \leq i \leq r \)
Recall $\Phi = \sqrt{\frac{1}{r} \Pi W}$

The proof follows proving following four claims

- $W$ preserves isometry (unitary matrix)
- $Wx$ has bounded infinity norm for all unit vector $x$
- $P_{2\Pi}$ preserves isometry if its input has bounded unit norm
- $P_1$ preserves isometry (Hanson-Wright inequality)
Differentially Private Linear Regression

---

**Objective:** Given a private matrix $A$ and a column vector $b$, output a vector $x$ such that

$$\Pr \left[ \|Ax - b\|_F \leq (1 + \alpha) \min_{y \in \mathbb{R}^{d \times 1}} \|Ay - b\|_F + \tau \right] \geq 1 - \beta$$
Linear Regression: Problem Statement

Objective: Given a private matrix $A$ and a column vector $b$, output a vector $x$ such that

$$\Pr \left[ \|Ax - b\|_F \leq (1 + \alpha) \min_{y \in \mathbb{R}^{d \times 1}} \|Ay - b\|_F + \tau \right] \geq 1 - \beta$$

Why do we care?
Objective: Given a private matrix $A$ and a column vector $b$, output a vector $x$ such that

$$\Pr \left[ \|Ax - b\|_F \leq (1 + \alpha) \min_{y \in \mathbb{R}^{d \times 1}} \|Ay - b\|_F + \tau \right] \geq 1 - \beta$$

Why do we care?

– People in finance will tell you about capital asset pricing model

– Computer scientists will tell you about Reinforcement Learning
Linear Regression

- Set $s$ above the threshold of $\text{PSG}_1$
- Pick a random Gaussian matrix $\Phi$
• Set $s$ above the threshold of $\text{PSG}_1$
• Pick a random Gaussian matrix $\Phi$

Data-structure update. On input a stream $A_c$,
• Life the singular values: $\hat{A}_c = (s e_c \ 0^{n+d} \ A_c)$
• Compute the sketch $Y_c$ of $t \hat{A}_c$.

Form the matrix $Y = (Y_1 : \cdots : Y_n)$
Linear Regression

Stored Data-structure: $Y$

Answering queries. On being queried with a vector $b$

- Set $\hat{b} = (0^d \ 0^{n+d} \ b_i)$
- Compute the sketch of $\tilde{b}$
- Compute $\arg \min_x \|Yx - \tilde{b}\|$

Privacy follows from $PSG_1$ as $s = \sigma_1$
Stored Data-structure: $Y$

Answering queries. On being queried with a vector $b$

- Set $\tilde{b} = (0^d \ 0^{n+d} \ b_i)$
- Compute the sketch of $\tilde{b}$
- Compute $\arg\min_x ||Yx - \tilde{b}||$

Utility follows using the Johnson-Lindenstrauss property
Differentially Private Manifold Learning

\[4\]

**Differentially Private Manifold Learning**

**Input:** \( Y = \{y_1, \cdots, y_m\} \) in \( \mathbb{R}^n \).

- Set \( x_i = (\sigma e_i \ y_i)^\top \) for all \( 1 \leq i \leq m \).
- Let \( X = (x_1 \ \cdots \ x_m) \).

while residual variance \( \geq \zeta \)

- Sample \( \Phi \)
- Run the Grassberger-Procaccia’s algorithm on \( \Phi X \)
- Use \( \widehat{k} \) to perform Isomap on \( \Phi X \).
- Calculate the residual variance, increment \( r \)

Invoke the Isomap algorithm with \( \widehat{k}, \{x_1, \cdots, x_m\} \), and \( \Phi \)

Privacy follows from PSG\(_1\)
Convergence requires proving two things. If

\[ r \geq O \left( \frac{k \log(nV\kappa) \log(1/\delta)}{\varepsilon^2} \right) \]

then with probability \(1 - \beta\),

- Residual variance of the embedded points are bounded by residual variance of the sampled points

\[ R_\Phi \leq R + O \left( c\varepsilon d^2 + c\varepsilon \left( \frac{\ln(4/\beta) \sqrt{16r \log(2/\beta)}}{\alpha} \right)^2 \right) \]

- Intrinsic dimension of embedded points is within multiplicative factor of that of sampled points