# Releasing Private Data for Numerical Queries

Yuan Qiu, Wei Dong, Ke Yi, Bin Wu, Feifei Li

HKUST

Alibaba Group

# Differential Privacy

- $D \in \mathcal{X}^n$: A dataset containing $n$ tuples from universe $\mathcal{X}$
- A mechanism $\mathcal{M}$ is $(\varepsilon, \delta)$-DP if for all neighboring datasets $D \sim D'$ and subset of outputs $O$, we have

$$\Pr[\mathcal{M}(D) \in O] \leq e^{\varepsilon} \cdot \Pr[\mathcal{M}(D') \in O] + \delta$$

- Adding noise calibrated to the global sensitivity of a query protects DP
  - Given query $f: \mathcal{X}^n \to \mathbb{R}$, the mechanism $\mathcal{M}(D) = f(D) + \mathrm{Lap}\left(\frac{\Delta_f}{\varepsilon}\right)$ is $(\varepsilon, 0)$-DP.
  - $\Delta_f = \max_{D,D':D\sim D'} |f(D) - f(D')|$ is the Global Sensitivity of $f$

# Counting/Linear Queries vs Numerical Queries

- A linear query is given by $\ell: \mathcal{X} \to [0,1]$, and $\ell(D) = \sum_{t \in D} \ell(t)$
- A numerical query is given by $w: \mathcal{X} \to \mathbb{R}$, and $w(D) = \sum_{t \in D} w(t)$
- Example
  - The number of people with income between $a$ and $b$

$$w(t) = \mathbf{1}[a \leq t[\text{income}] \leq b]$$

  - The total income of people whose income is between $a$ and $b$

$$w(t) = \mathbf{1}[a \leq t[\text{income}] \leq b] \cdot t[\text{income}]$$

  - The variance of income of people whose age is between $a$ and $b$

$$w(t) = \mathbf{1}[a \leq t[\text{age}] \leq b] \cdot t[\text{income}]^2$$

  - The total weighted income

$$w(t) = \text{UDF}(t[\text{age}], t[\text{income}]) \cdot t[\text{income}]$$

| Age | Income |
|---|---|
| 35 | 2560 |
| 20 | 1500 |
| 45 | 8170 |
| 35 | 5200 |
| 45 | 3000 |
| ... | ... |

# Private Multiplicative Weights [Hardt et al. '12]

- Given a dataset $D \in \mathcal{X}^n$ and a set of <u>linear</u> queries $\mathcal{L} = \{\ell_1, \ell_2, \dots, \ell_{|\mathcal{L}|}\}$
- The private multiplicative weights mechanism has the following guarantees
  - It runs in $T$ iterations, with each round being $(\varepsilon_0, 0)$-DP and taking $\tilde{O}(|\mathcal{X}| \cdot |\mathcal{L}|)$ time
  - With probability $1-\beta$, all queries $\ell \in \mathcal{L}$ can be answered on $\widetilde{D} = \mathcal{M}(D)$ within error

$$\alpha = O\left(\frac{n\sqrt{\log|\mathcal{X}|}}{\sqrt{T}} + \frac{\log(|\mathcal{L}|/\beta)}{\varepsilon_0}\right)$$

- Setting $T = \widetilde{\Theta}(\varepsilon n)$ and $\varepsilon_0 = \Theta\left(\frac{\varepsilon}{\sqrt{T\log(1/\delta)}}\right)$ achieves $(\varepsilon, \delta)$-DP with error

$$\alpha = O\left(\frac{\sqrt{n\log(|\mathcal{L}|/\beta)\sqrt{\log|\mathcal{X}|\log(1/\delta)}}}{\sqrt{\varepsilon}}\right) = \tilde{O}(\sqrt{n})$$

# DP Numerical Queries: Normalization

- For simplicity, we consider numerical queries $w: \mathcal{X} \rightarrow \{0,1,2,\ldots,\Delta\}$
  - We also assume $\Delta$ is a power of 2, e.g. $2^{64}$
- The target is to answer a set of numerical queries $Q = \{w_1, w_2, \ldots, w_{|Q|}\}$ privately
- Normalization
  - Given a numerical query $w$, define $\Delta_w \coloneqq \max_{t \in \mathcal{X}} w(t)$
  - It is clear that $\ell_w(t) \coloneqq w(t)/\Delta_w \in [0,1]$ is a linear query
  - Every normalized query $\ell_w$ for $w \in Q$ can be answered by $\widetilde{D}$ with error $\tilde{O}(\sqrt{n})$
  - Rescaling the results, query $w$ can be answered with error $\tilde{O}(\sqrt{n} \cdot \Delta_w)$
- Problem
  - $\Delta_w$ is data-independent, and can be arbitrarily large, e.g. $2^{64}$

# DP Numerical Queries: Truncation [Huang et al., '21]

- When $Q = \{w\}$ contains a single numerical query, recent work has error $\tilde{O}\left(\Delta_w(D)\right)$
  - $\Delta_w(D) \coloneqq \max_{t \in D} w(t)$ is an instance-specific bound
- Truncation
  - Find a privatized truncation threshold $\tau$ such that
    - Only $\tilde{O}(1)$ tuples in $D$ have $w(t) > \tau$
    - $\tau \leq \Delta_w(D)$
  - Define a truncated query $\overline{w}(t) = \min\{w(t), \tau\}$
  - Answer the truncated query with $O(\tau) = O\left(\Delta_w(D)\right)$ noise
  - The truncation error $|w(D) - \overline{w}(D)|$ is also $\tilde{O}\left(\Delta_w(D)\right)$
- Problem
  - It is nontrivial to extend it to multiple queries

# Comparison of Error Bounds

- Normalization
  - Normalize each query by $\Delta_w$, and apply PMW to answer the linear queries
- Composition
  - Run truncation in [Huang et al., ’21] for each $w \in Q$ with tighter privacy budgets
- Global Truncation:
  - Spend a constant fraction of budget to find threshold $\Delta(D) \coloneqq \max_{w \in Q} \max_{t \in D} \Delta(D)$

| Mechanism | Error bound for $w \in Q$ | Many Queries? | Query-Specific? | Instance-Specific? |
|---|---|---|---|---|
| Normalization | $\tilde{O}(\sqrt{n} \cdot \Delta_w)$ | ✓ | ✓ | |
| Composition | $\tilde{O}\left(\sqrt{\lvert Q \rvert} \cdot \Delta_w(D)\right)$ | | ✓ | ✓ |
| Global truncation | $\tilde{O}\left(\sqrt{n} \cdot \Delta(D)\right)$ | ✓ | | ✓ |
| New method | $\tilde{O}\left(\sqrt{n} \cdot \Delta_w(D)\right)$ | ✓ | ✓ | ✓ |

# Comparison of Error Bounds: Example

- Assume the dataset consists of integers, $\mathcal{X} = [0, 2^{32}]$
- Consider a set of range-aggregate queries with all different ranges $[a, b]$

$$w(t) = \mathbf{1}[a \le t \le b] \cdot t$$

- As there are many queries $|Q| = \Theta(|\mathcal{X}|^2) \gg n$, composition has a large error
- Normalization
  - $\Delta_w = \max_{t \in \mathcal{X}} w(t) = b$
- Global Truncation
  - $\Delta(D) = \max_{w \in Q} \max_{t \in D} w(t) = \max\{t \in D\}$
- New method
  - $\Delta_w(D) = \max_{t \in D} w(t) = \max\{t \in D : t \le b\}$

| Mechanism | Error bound |
|---|---|
| Composition | $\tilde{O}\left(\sqrt{\lvert Q \rvert} \cdot \Delta_w(D)\right)$ |
| Normalization | $\tilde{O}(\sqrt{n} \cdot \Delta_w)$ |
| Global truncation | $\tilde{O}\left(\sqrt{n} \cdot \Delta(D)\right)$ |
| New method | $\tilde{O}\left(\sqrt{n} \cdot \Delta_w(D)\right)$ |

# Query- and Instance-Specific Truncation

- The sketch of our algorithm is as follows
  1. Given numerical queries $Q$, generate a set of counting queries $\mathcal{C}(Q)$
  2. Run the PMW mechanism to privately answer all the queries in $\mathcal{C}(Q)$
  3. From these query answers, extract the truncation threshold $\overline{\Delta}_w(D)$ for every $w \in Q$
  4. Truncate and normalize each query $w$ by $\overline{\Delta}_w(D)$ to obtain a set of linear queries $\mathcal{L}(Q)$
  5. Run the PMW mechanism to privately answer all the queries in $\mathcal{L}(Q)$
  6. Scale the results back by $\overline{\Delta}_w(D)$ to get a privatized $w(D)$

# Truncation Thresholds

- We want to find $\overline{\Delta}_w(D)$ for query $w$ with the following guarantees
  1. $|\{t \in D: w(t) > \overline{\Delta}_w(D)\}| \leq 2\alpha$
     - $\alpha = \tilde{O}(\sqrt{n})$ is the error in answering linear queries
     - Only $O(\alpha)$ values are truncated, each brings error $w(t) \leq \max_{t \in D} w(t) = \Delta_w(D)$
  2. $\overline{\Delta}_w(D) \leq 2\Delta_w(D)$
     - After normalizing by $\overline{\Delta}_w(D)$, we answer the linear queries with error $\alpha$
     - When scaling the linear query back, the error is scaled by $\overline{\Delta}_w(D) = O\big(\Delta_w(D)\big)$
- If we can (privately) find $\overline{\Delta}_w(D)$ with these guarantees, it follows that any $w \in Q$ is answered with error $O\big(\alpha \cdot \Delta_w(D)\big) = \tilde{O}\left(\sqrt{n} \cdot \Delta_w(D)\right)$

# Finding Truncation Thresholds

- We can perform a doubling search to find the truncation thresholds
- Candidates: $\tau \in \{0,1,2,4,8, \ldots, \Delta\}$
- For each candidate $\tau$, we ask the query
  - $c_{w,\tau}(t) = \mathbf{1}[w(t) > \tau]$
  - i.e., How many $t \in D$ have $w(t) > \tau$?
- The query can be answered with error $\alpha$, so if the count is $c_{w,\tau}(D) \leq \alpha$, we can return $\overline{\Delta}_w(D) = \tau$ so that it satisfies condition 1
  - $|\{t \in D: w(t) > \overline{\Delta}_w(D)\}| \leq 2\alpha$
- It is can also be shown that condition 2 is satisfied
  - $\overline{\Delta}_w(D) \leq 2\Delta_w(D)$

# Combining the Two PMW Instances

- The two PMW instances are run on the same $D$ with different queries $\mathcal{C}(Q), \mathcal{L}(Q)$
- We can combine them by feeding the union of all queries
- The counting queries $\mathcal{C}(Q) = \left\{c_{w,\tau} | w \in Q, \tau \in \left\{0,1,2,4,8, \dots, \frac{\Delta}{2}\right\}\right\}$
  - Where $c_{w,\tau}(t) = \mathbf{1}[w(t) > \tau]$
- The linear queries $\mathcal{L}(Q) = \left\{\ell_{w,\tau} | w \in Q, \tau \in \{1,2,4,8, \dots, \Delta\}\right\}$
  - Where $\ell_{w,\tau}(t) = \frac{\min\{w(t),\tau\}}{\tau} = \min\left\{\frac{w(t)}{\tau}, 1\right\}$
- There are only $O(|Q| \log \Delta)$ queries to be answered by PMW

$$\alpha = O\left(\frac{\sqrt{n \log((|Q| \log \Delta)/\beta) \sqrt{\log|\mathcal{X}| \log(1/\delta)}}}{\sqrt{\varepsilon}}\right) = \tilde{O}(\sqrt{n})$$

# Decomposable Queries

- Recall that each iteration of PMW takes $\tilde{O}(|\mathcal{X}| \cdot |Q|)$ time
- For numerical queries, $|\mathcal{X}|$ is usually large
  - e.g., age $\in [1,128]$ and income $\in [1,2^{32}]$, then $|\mathcal{X}| = 2^{40}$
- Decomposable queries
  - We say a set of queries $Q$ is decomposable if
    - There exists an equivalence relation $R$ over $\mathcal{X}$
    - There exists a function $g: \mathcal{X} \to \{0,1,2,\dots,\Delta\}$
    - Every $w \in Q$ can be written as $w(t) = f_w([t]_R) \cdot g(t)$ for some $f_w: \mathcal{X}/R \to [0,1]$
  - $[t]_R$ is the equivalence class induced by $R$ containing $t$
  - $g$ is common to the entire $Q$, while $f_w$ is different for each $w$

| Age | Income |
|---|---|
| 35 | 2560 |
| 20 | 1500 |
| 45 | 8170 |
| 35 | 5200 |
| 45 | 3000 |
| ... | ... |

# Decomposable Queries: Example

- There is a trivial decomposition for any set of queries $Q$
  - $R = \{(t, t): t \in \mathcal{X}\}$
  - $\mathcal{X}/R = \mathcal{X}$
  - $g(t) \equiv \Delta$
  - $f_w(t) = w(t)/\Delta$
- We are interested in decompositions where $|\mathcal{X}/R|$ is small
  - If $Q$ consists of queries of form

$$w(t) = \mathbf{1}[a \leq t[\text{age}] \leq b] \cdot t[\text{income}]$$

  - $R$ puts all tuples of the same age into an equivalence class
  - $\mathcal{X}/R = \text{dom}(\text{age})$
  - $g(t) = t[\text{income}]$
  - $f_w(t) = \mathbf{1}[a \leq t[\text{age}] \leq b]$

| Age | Income |
|---|---|
| 35 | 2560 |
| 20 | 1500 |
| 45 | 8170 |
| 35 | 5200 |
| 45 | 3000 |
| ... | ... |

# Reducing Universe Size for Decomposable Queries

- Decomposable query: $w(t) = f_w([t]_R) \cdot g(t)$
- We consider a new universe
  - $\hat{\mathcal{X}} = \mathcal{X}/R \times \{1,2,4,8, \dots, \Delta\}$
  - Decompose $g(t)$ for every $t$ using binary decomposition
  - Note that $g(t)$ is common to $Q$
- e.g. Decomposing tuple (age=35, income=2560)
  - We generate 2 tuples (35, 2048) and (35, 512) over $\hat{\mathcal{X}}$
  - For any $w$, we have
  - $w((35{,}2560)) = f_w(35) \cdot 2560 = f_w(35) \cdot 2048 + f_w(35) \cdot 512$
  - We just need to run the query on the new $\hat{D}$ over $\hat{\mathcal{X}}$
- A separate privacy analysis is needed

| Age | Income |
|---|---|
| 35 | 2560 |
| 20 | 1500 |
| 45 | 8170 |
| 35 | 5200 |
| 45 | 3000 |
| … | … |

# Improving for Queries with Structural Properties

- For special counting queries, e.g. range/half-space counting, the accuracy is better
- This also applies to our mechanism
  - $\{f_w\}$ can have structural properties
  - e.g. If $Q$ consists of queries of form

$$w(t) = \mathbf{1}[a \leq t[\text{age}] \leq b] \cdot t[\text{income}]$$

    then $f_w$ are all range queries
  - As range counting has error $\tilde{O}(1)$ under DP, we can achieve error $\tilde{O}\left(\Delta_w(D)\right)$

# Conclusion

- We initiate the study of private data release for numerical queries
- Our mechanism achieves instance- and query-specific error $\tilde{O}\left(\sqrt{n} \cdot \Delta_w(D)\right)$
- The error bound also leads to excellent practical performance
- For decomposable queries, the running time and accuracy can be further improved

## References


Moritz Hardt, Katrina Ligett, and Frank McSherry. 2012. A Simple and Practical Algorithm for Differentially Private Data Release. In Conference on Neural Information Processing Systems (NeurIPS).

Ziyue Huang, Yuting Liang, and Ke Yi. 2021. Instance-optimal Mean Estimation Under Differential Privacy. In Conference on Neural Information Processing Systems (NeurIPS).