Faster Algorithms for Privately Releasing Marginals

Justin Thaler (Harvard University)
Jon Ullman (Harvard University)
Salil Vadhan (Harvard University)
**K-way Marginal Queries**

\[ D \in \left( \{0, 1\}^d \right)^n \]

<table>
<thead>
<tr>
<th>Exercise?</th>
<th>Healthy?</th>
<th>Ice Cream?</th>
<th>Criminal?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

Query on a row: \( q(x) = \text{Ice Cream?} \land \text{Criminal?} \)

Query on database: \( \frac{1}{n} \sum_i q(x_i) \)

- **k-way marginal queries**: \( q \) has at most \( k \) literals.
- Number of k-way marginal queries \( \sim d^k \).
Goal: Private One-Shot Release Mechanism

• Want to release a summary of D such that for all k-way marginals q:

  \[ |\text{Summary}(q) - q(D)| \leq 0.01 \]

• Two parameters to optimize: running time of the sanitizer and minimal database size required.
Prior Work on Marginal

Running Time

Minimum DB Size
Prior Work on Marginals

\[ d^k = |Q| \]

- Laplace Mechanism
  [DN, BDMN, DMNS]

Minimum DB Size

\[ d^{k/2} \]
Prior Work on Marginals

$2^d$ • [BLR,...]  
$d^k = |Q|$  • Laplace Mechanism  
$[DN,BDMN,DMNS]$  
$k\sqrt{d}$  
$dk^{k/2}$  
Minimum DB Size
Prior Work on Marginals

- Minimum DB Size: $d^k = |Q|$
- Running Time: $2^d$
- $d^{k/2}$
- $d^{c\sqrt{k}}$
- $k\sqrt{d}$

- Laplace Mechanism [DN, BDMN, DMNS]

- [HRS]

- [BLR, ...]
Prior Work on Marginals

- $2^d$
- $d^k = |Q|$
- $poly(d,k)$
- $k\sqrt{d}$
- $d^{c\sqrt{k}}$
- $d^{k/2}$

Minimum DB Size

- [BLR,...]
- [HRS]
- Laplace Mechanism
  [DN,BDMN,DMNS]
- Holy grail
Our Result

- $2^d$
- $d^k = |Q|$
- $d^{c\sqrt{k}}$
- $\text{poly}(d,k)$
- $k\sqrt{d}$
- $d^{c\sqrt{k}}$
- $d^{k/2}$

- Holy grail
- [BLR, ...]
- [HRS]
- This Work
- Laplace Mechanism [DN, BDMN, DMNS]
Our Results

• Faster algorithm for privately releasing marginals with small worst-case error (accuracy $\pm 0.01$).
  • Time: $d^{C\sqrt{k}}$, minimum database size: $n \geq d^{C\sqrt{k}}$.
  • First sanitizer for $k$-way marginals with running time and minimal DB size sublinear in total number of $k$-way marginals $\sim d^k$.
• Can handle more general settings as well (e.g. where rows of the DB represent decision lists).
Our Algorithm

\[ D \subseteq (\{0, 1\}^d)^n \]

<table>
<thead>
<tr>
<th>x_1</th>
<th>x_2</th>
<th>x_3</th>
<th>x_4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

• View each row x as a function \( f_x \) from queries to \( \{0, 1\} \):
  \[ f_x(q) = 1 \text{ iff row } x \text{ satisfies marginal } q. \]
Our Algorithm

\[ D \subseteq (\{0, 1\}^d)^n \]

<table>
<thead>
<tr>
<th>x₁</th>
<th>x₂</th>
<th>x₃</th>
<th>x₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

• View each row \( x \) as a function \( f_x \) from queries to \( \{0, 1\} \):
  \[ f_x(q) = 1 \text{ iff row } x \text{ satisfies marginal } q. \]

• For every \( x \), there exists a \( d \)-variate polynomial \( p_x \) such that:
  • \[ |p_x(q) - f_x(q)| \leq .01 \] for all \( q \) corresponding to \( k \)-way marginals.
  • \( \text{Degree}(p) \leq C\sqrt{k} \) for some constant \( C \).
  • All coefficients of \( p \) are in \([\pm d^{C\sqrt{k}}]\).
Our Algorithm

\[ D \subseteq (\{0, 1\}^d)^n \]

<table>
<thead>
<tr>
<th></th>
<th>x_1</th>
<th>x_2</th>
<th>x_3</th>
<th>x_4</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_{x_1}(y) = \ldots</td>
<td>3y_1y_2 + 7y_2y_4 + \ldots</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>p_{x_2}(y) = \ldots</td>
<td>4y_1y_2 - 3y_2y_4 + \ldots</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>p_{x_3}(y) = \ldots</td>
<td>-3y_1y_2 + 2y_2y_4 + \ldots</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>p_{x_4}(y) = \ldots</td>
<td>8y_1y_2 + y_2y_4 + \ldots</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Our Algorithm

\[ D \subseteq (\{0, 1\}^d)^n \]

<table>
<thead>
<tr>
<th></th>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_3 )</th>
<th>( x_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p_{x_1}(y) )</td>
<td>3y_1y_2 + 7y_2y_4 + ...</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( p_{x_2}(y) )</td>
<td>4y_1y_2 - 3y_2y_4 + ...</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( p_{x_3}(y) )</td>
<td>-3y_1y_2 + 2y_2y_4 + ...</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( p_{x_4}(y) )</td>
<td>8y_1y_2 + y_2y_4 + ...</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- Let \( p_D(y) = \frac{1}{n} \sum_i p_{x_i}(y) \) be the average of the polynomials approximating each row.
- We output a noisy version of \( p_D \).
Our Algorithm

\[ D \in (\{0, 1\}^d)^n \]

<table>
<thead>
<tr>
<th>x₁</th>
<th>x₂</th>
<th>x₃</th>
<th>x₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\( p_{x1}(y) = 3y_1y_2 + 7y_2y_4 + \ldots \)
\( p_{x2}(y) = 4y_1y_2 - 3y_2y_4 + \ldots \)
\( p_{x3}(y) = -3y_1y_2 + 2y_2y_4 + \ldots \)
\( p_{x4}(y) = 8y_1y_2 + y_2y_4 + \ldots \)

• Let \( p_D(y) = (1/n) \sum_i p_{xi}(y) \) be the average of the polynomials approximating each row.
• We output a noisy version of \( p_D \).
  • Degree(\( p_D \)) = \( C\sqrt{k} \). So about \( d^{C\sqrt{k}} \) coefficients.
  • \( p_D \) has coefficients in \( [\pm d^{C\sqrt{k}}] \), each coeff has sensitivity ~ \( d^{C\sqrt{k}} / n \)
  • Add independent Laplace noise to each coeff of magnitude ~ \( d^{C\sqrt{k}} / n \).
 Conclusion

• Previous sanitizers [HRS, etc.] gave a learning algorithm restricted access to the DB.
• We cut out the learning algorithm, and give our sanitizer direct access to the database.
  • We use the same structural results underlying many learning algorithms.
• Does relying on learning algorithms for differential privacy unnecessarily tie our hands?