Learning from private data

Example: Learn new words and emoji’s

Why is privacy a concern?

• Adversary can observe the output, or even worse, compromise the server

• Companies use similar technology for analyzing e.g., sensitive data
  • Google uses it for collecting malicious URL visit
  • Apple uses it for collecting new words people type, health related information
  • LinkedIn uses it for collecting salary information

Prior work

[Bassily and Smith 2015]

• Algorithm based on dimensionality reduction via the Johnson Lindenstrauss transformation
  • Optimal error guarantee, non-optimal in terms of computation, storage and communication

[Thakurta et al., 2016]

• Optimal frequency estimator via count-median-sketch
  • Heavy hitters algorithm is a heuristic, with no formal guarantees

[Erlingsson et al., 2013, Fanti et al., 2016]

• Empirical results only, with no formal guarantees

[Hsie et al., 2014]

• Algorithm based on group testing

Our contributions

Two new algorithms with optimal error, with optimal time, space and communication complexity

• TreeHist: Based on classic count sketch [CCFC02]
  • Nearly optimal error guarantee
  • Easily implementable in practice

• Bistogram: Based on error correcting codes
  • Optimal error guarantee

Implementation of our TreeHist protocol

• Scales to 10 million data samples, with domain \( \approx 10^8 \)
• Performs better than state-of-the-art open source tools

Our results at a glance

<table>
<thead>
<tr>
<th>Performance metric</th>
<th>TreeHist (this work)</th>
<th>Bistogram (this work)</th>
<th>Bassily and Smith/15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server time</td>
<td>( O(n) )</td>
<td>( O(n \log^2 d) )</td>
<td>( O(n \log d \log^2 \log d) )</td>
</tr>
<tr>
<td>User time</td>
<td>( O(1) )</td>
<td>( O(1) )</td>
<td>( O(1) )</td>
</tr>
<tr>
<td>Server memory</td>
<td>( O(\sqrt{n \log d}) )</td>
<td>( O(\sqrt{n \log d}) )</td>
<td>( O(\sqrt{n \log d}) )</td>
</tr>
<tr>
<td>User memory</td>
<td>( O(1) )</td>
<td>( O(1) )</td>
<td>( O(1) )</td>
</tr>
<tr>
<td>Communication</td>
<td>( O(1) )</td>
<td>( O(1) )</td>
<td>( O(1) )</td>
</tr>
<tr>
<td>Error</td>
<td>( \sqrt{n \log d} )</td>
<td>( \sqrt{n \log d} )</td>
<td>( \sqrt{n \log d} )</td>
</tr>
</tbody>
</table>

\( n \): Number of data samples
\( d \): Domain size
\( \epsilon \): can find all heavy hitters with frequency \( \geq 2^{\epsilon} \)
and none with frequency \( \leq n^{1+\epsilon} \)


Algorithm overview

Data

Frequency estimator

Heavy hitter identifier

Locally private frequency estimator

Problem: Design locally differentially private estimator with

• Accuracy: \( O(\sqrt{\log d}) \)
• Query time / estimate: \( O(\log d) \)

Communication: \( 1 \) bit / user
Client computation: Constant
Server computation: \( O(\sqrt{\log d}) \)

Non-private count-sketch (CCFC02)

Client side: Sketch matrix: \( M \in \{-1,0,1\}^{3\times n} \)

\( \epsilon_0 \) \( \vdots \) \( \epsilon_5 \)

Server side: Hash functions: Pairwise independent \( s_i \in [k] \) with \( d_i \in \{|0,1\} \)

Frequent independent \( s_i \in [k] \) with \( d_i \in \{|0,1\} \)

The server computes the frequent heavy hitters via the prefix tree

Pseudocode for TreeHist communication protocol

1. Sample a row \( M_i \) from the sketch matrix \( M \)
2. Replace each zero of \( M_i \) with a sampled \( x \in [n] \)
3. Flip each bit of \( M_i \) with probability \( \frac{1}{2^{\epsilon_0}} \) and send \((i,M_i)\)

Server side: For client report \( (i,v) \), add \( \epsilon_0 \) to the \( i \)-th row of the server side sketch \( S \)

Optimal error guarantee, non-optimal in terms of computation, storage and communication

TreeHist algorithm is \( \epsilon \) locally differentially private, and satisfies all the desired utility guarantees, except communication

Reducing communication via Hadamard transform

Theorem: Hadamard transform and sampling does not affect privacy or utility, but reduces communication to \( O(1) \) bit

Empirical evaluation with TreeHist

Comparison to RAPTOR project:

• \( n = 2 \), \( m = 10 \) mi, \( d \) of unique words = \( 20001 \), wordlength = \( 6 \)

Thresholds: 15/16

Empirical evaluation with TreeHist

Complexity of RAPTOR project:

• \( n = 1.19 \), \( m = 1 \) mi, domain = \( 100 \)

Heavy hitters for NLTK data set:

\( \epsilon = 1.09 \), \( n = 20 \) mi, \( d \) of unique words = \( 20001 \), wordlength = \( 6 \)

Thresholds: 15/16

Practical Locally Private Heavy Hitters

Raef Bassily1 Kobbi Nissim2 Uri Stemmer3 Abhradeep Thakurta4

1Ohio State University 2Georgetown University 3Harvard University 4University of California Santa Cruz