CS208: Applied Privacy for Data Science
The Local Model: Implementations

James Honaker & Salil Vadhan
School of Engineering & Applied Sciences
Harvard University

April 5, 2019
**Example: Randomized response**

- Each person has data $x_i \in X$
  - Analyst wants to know average of $f: X \to \{-1,1\}$ over $x$
  - E.g. “what is the fraction of diabetics”?
- Randomization operator takes $y \in \{-1,1\}$:
  
  $$Q(y) = \begin{cases} 
  +y & \text{w. p. } \frac{e^\epsilon}{e^\epsilon + 1} \\
  -y & \text{w. p. } \frac{1}{e^\epsilon + 1}
  \end{cases}$$
  
  ratio is $e^\epsilon$

- Observe:
  - If $c_\epsilon = \frac{e^\epsilon + 1}{e^\epsilon - 1}$, then $E(c_\epsilon \cdot Q(y)) = y$
- How can we estimate a proportion?
  - $A(x_1, \ldots, x_n) = \frac{1}{n} \sum_i c_\epsilon \cdot Q(f(x_i))$
- **Proposition:** $E \left| A(x) - \frac{1}{n} \sum_i f(x_i) \right| \leq \frac{c_\epsilon}{2\sqrt{n}} \approx \frac{1}{\epsilon \sqrt{n}}$.

*Slide from Adam Smith*
Probability Truth Revealed in Local Model

\[ \text{probability} \]

\[ \text{epsilon} \]
Local DP

**Require:** for all $i, x_i, x_i'$ differing on one row, all strategies $A$

\[
\Pr[A \text{ outputs YES after interacting w/ } Q_i(x_i)] \\
\leq e^\varepsilon \cdot \Pr[A \text{ outputs YES after interacting w/ } Q_i(x_i')] + \delta
\]
Recap: DP Histograms

- Local randomizer \( Q(x_i) \) for \( x_i \in \{1, \ldots, D\} \)
  1. Construct “1-hot” vector \( e_{x_i} = (0,0, \ldots, 0,1,0, \ldots, 0) \in \{0,1\}^D \).
  2. Apply \((\varepsilon/2)\)-DP RR to each coordinate to get \( y_i \in \{0,1\}^D\):

\[
y_i[j] = \begin{cases} 
  e_{x_i}[j] & \text{w.p. } e^{\varepsilon/2}/(1 + e^{\varepsilon/2}) \\
  1 - e_{x_i}[j] & \text{o.w.}
\end{cases}
\]

  3. Send \( y_i \) to server.

- Server uses \((y_1, \ldots, y_n)\) to estimate histogram \( f = \sum_i e_{x_i} \).
- Error per bin \( O(\sqrt{n}/\varepsilon) \).
Discovering Unknown Values (Apple)

- Method described so far requires server to decide on a small set values $\nu_1, \ldots, \nu_s$ to estimate frequencies of.

- **Goal:** find unanticipated frequent values $\nu$

- **Idea:** reconstruct $\nu$ one symbol at a time
  - do RR on $h(x_i)||x_i[j]$ for each bitposition $j = 1, \ldots, \log D$.
  - $\hat{f}[w||\sigma_j]$ is large $\Rightarrow$ there probably is a frequently occurring value $\nu \in \{1, \ldots, D\}$ such that $h(\nu) = w$ and $\nu[j] = \sigma_j$.
  - $\hat{f}[w||\sigma_1], \ldots, \hat{f}[w||\sigma_{\log D}]$ large $\Rightarrow \nu = \sigma_1 \cdots \sigma_{\log D}$
**Algorithm 2** Client-Side $\mathcal{A}_{\text{client-CMS}}$

**Require:** Data element: $d \in D$; $\epsilon$, $\mathcal{H}$.
1. Sample $j$ uniformly at random from $[k]$.
2. Initialize a vector $\mathbf{v} \leftarrow -1 \in \mathbb{R}^m$.
3. Set $v_{h_j(d)} \leftarrow 1$.
4. Sample $\mathbf{b} \in \{-1, +1\}^m$, where each $b_\ell$ is i.i.d. where $\Pr[b_\ell = +1] = \frac{e^{\epsilon/2}}{e^{\epsilon/2}+1}$.
5. $\bar{\mathbf{v}} \leftarrow (v_1b_1, \ldots, v_mb_m)$.
6. **return** $\bar{\mathbf{v}}$, index $j$.

**Algorithm 9** Client-Side $\mathcal{A}_{\text{client-SFP}}$

**Require:** String: $s \in D$; privacy parameters: $(\epsilon, \epsilon')$, hash functions $(\mathcal{H}, \mathcal{H}')$, and $h$ with output size 256.
1. Sample $\ell$ uniformly at random from $\{1, 3, 5, 7, 9\}$.
2. Set $\mathbf{r} \leftarrow h(s)||s[\ell : \ell + 1]$.
3. **return** $(\mathcal{A}_{\text{client-CMS}}(r, \epsilon', \mathcal{H}'), \mathcal{A}_{\text{client-CMS}}(s, \epsilon, \mathcal{H}), \ell)$. 
Discovering Unknown Values (Apple)

<table>
<thead>
<tr>
<th>Actual Names:</th>
<th>setosa</th>
<th>versicolor</th>
<th>virginica</th>
</tr>
</thead>
<tbody>
<tr>
<td>temp.l</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>temp.b</td>
<td>1 2 3 4 5 6 7 8 9 10</td>
<td>0 0 0 0 0 0 0 264 236 220 316</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>0 0 0 0 0 0 247 0 0 89 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>0 0 0 0 0 79 0 98 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>0 354 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>g</td>
<td>0 0 0 99 0 0 0 0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>i</td>
<td>0 72 0 0 188 0 80 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>l</td>
<td>0 0 0 0 0 0 0 0 105 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n</td>
<td>0 0 0 0 0 82 0 0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>o</td>
<td>0 0 0 219 0 0 0 101 0 87 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>0 0 190 0 0 0 0 0 98 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s</td>
<td>244 0 0 81 250 0 0 0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>0 0 243 0 0 0 0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>v</td>
<td>185 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[1] "s" "e" "t" "o" "s" "a" "" "" "" "" ""

<table>
<thead>
<tr>
<th>Actual Names:</th>
<th>setosa</th>
<th>versicolor</th>
<th>virginica</th>
</tr>
</thead>
<tbody>
<tr>
<td>temp.l</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>temp.b</td>
<td>1 2 3 4 5 6 7 8 9 10</td>
<td>0 0 0 0 0 0 0 104 94 89 324</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>0 0 0 0 0 0 125 0 0 228 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>0 0 0 0 0 251 0 254 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>0 335 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>g</td>
<td>0 0 0 238 0 0 0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>i</td>
<td>0 243 0 0 493 0 231 0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>l</td>
<td>0 0 0 0 0 0 0 0 260 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n</td>
<td>0 0 0 0 0 234 0 0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>o</td>
<td>0 0 0 86 0 0 258 0 260 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>0 0 488 0 0 0 0 0 234</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s</td>
<td>100 0 0 222 92 0 0 0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>0 0 81 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>v</td>
<td>519 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[1] "v" "e" "r" "g" "i" "c" "o" "l" "o" ""
RAPPOR Setup

1. Change directory to where you want to install the RAPPOR repository, then:
   
   ```sh
   $ git clone https://github.com/google/rappor.git
   ```

2. Install R packages:
   
   ```r
   library.list <- c("glmnet", "limSolve", "ggplot2", "optparse", "shiny")
   install.packages(library.list, repos="https://cran.cnr.berkeley.edu/")
   ```

3. Change directory to
   
   ```sh
   $ cd rappor/apps/rappor-sim
   $ ./run_app.sh
   ```

4. Point browser to: `http://localhost:6788/`

5. Also worth looking at if you are exploring this codebase:
   
   ```sh
   cd rappor/apps/rappor-analysis
   $ ./run_app.sh
   ```
RAPPOR Simulation

Probability of reporting noise (\(q\)): 0.01

Probability of reporting signal (\(q\)): 0.75

Probability of lies (\(f\)): 0.0

Effective \(p\): 0.5625
Effective \(q\): 0.6875
\(\exp(e_{1})\): 2.9279
\(e_{1}\): 1.0743
\(\exp(e_{\inf})\): 81.000
\(e_{\inf}\): 4.3944
Detection frequency: 0.0065

* In addition to \(p\), \(q\) and \(f\), the number of hash functions (set in the RAPPOR tab) also affects privacy guarantees.