Accurate and Efficient Private Release of Data Cubes & Contingency Tables

Grigory Yaroslavtsev

PENNSTATE, work done at at&t

With **Graham Cormode**,

Cecilia M. Procopiuc

Divesh Srivastava

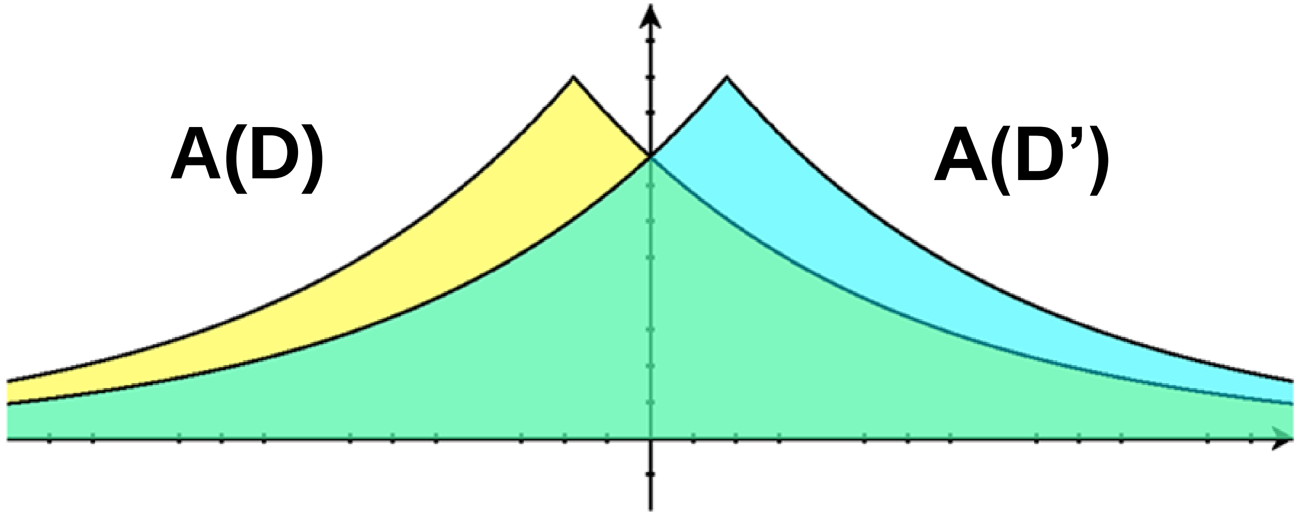
Differential privacy in databases

<u>ϵ-differential privacy</u>

For all pairs of neighbors $\boldsymbol{D}, \boldsymbol{D'}$ and all outputs **S**:

$$Pr[A(\boldsymbol{D}) = \boldsymbol{S}] \le e^{\epsilon} \Pr[A(\boldsymbol{D'}) = \boldsymbol{S}]$$

- **ϵ −privacy budget**
- Probability is over the randomness of A
- Requires the distributions to be close:



Optimizing Linear Queries

- Linear queries capture many common cases for data release
  - Data is represented as a vector x (histogram)
  - Want to release answers to linear combinations of entries of x
  - Model queries as matrix Q, want to know y=Qx
  - Examples: histograms, contingency tables in statistics

$$Q = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \qquad x = \begin{matrix} 3 \\ 5 \\ 7 \\ 0 \\ 1 \\ 4 \\ 9 \\ 2 \end{matrix}$$

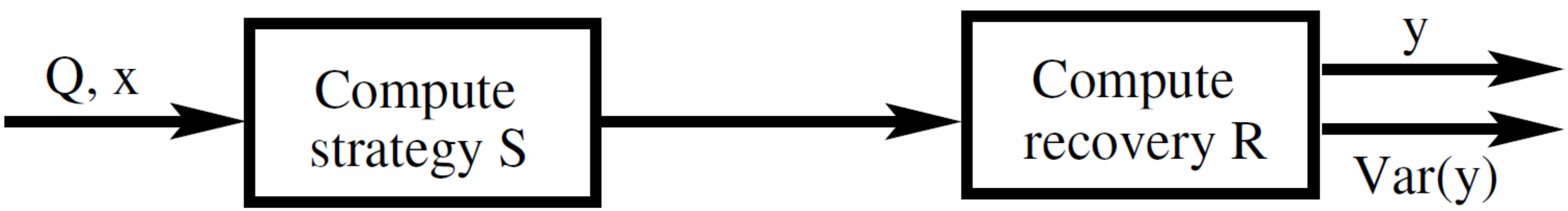
Answering Linear Queries

- Basic approach:
  - Answer each query in Q directly, partition the privacy budget **uniformly** and add **independent** noise
- Basic approach is suboptimal
  - Especially when some queries overlap and others are disjoint
- Several opportunities for optimization:
  - Can assign different privacy budgets to different queries
  - Can ask different queries S, and recombine to answer Q

$$Q=\begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

The Strategy/Recovery Approach

- Pick a strategy matrix S
  - Compute z = Sx + v → noise vector
    ↳ strategy on data
  - Find R so that Q = RS
  - Return y = Rz = Qx + Rv as the set of answers
  - Accuracy given by var(y) = var(Rv)



- Strategies used in prior work:

Q: Query Matrix

I: Identity Matrix

C: Selected Marginals

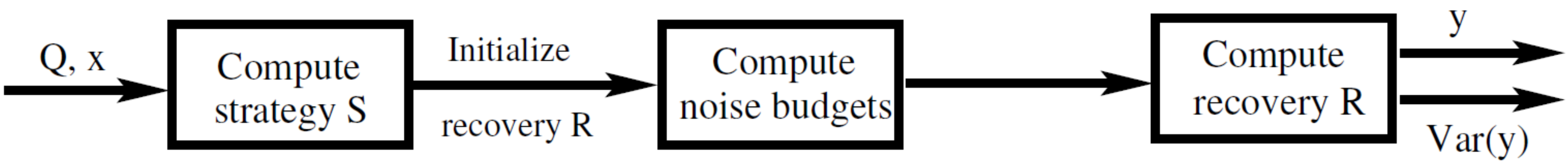
F: Fourier Transform Matrix

H: Haar Wavelets

P: Random projections

Step 2: Error Minimization

- Step 1: Fix strategy S for efficiency reasons
- Given Q, R, S, ε want to find a set of values $\{\varepsilon_i\}$
  - Noise vector v has noise in entry i with variance $1/\varepsilon_i^2$



- Yields an optimization problem of the form:

  Minimize $\sum_i b_i / \varepsilon_i^2$ (minimize variance)

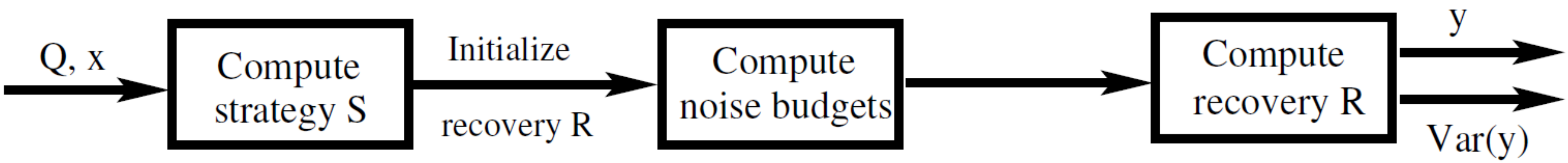
  Subject to $\sum_i |S_{i,j}| \varepsilon_i \leq \varepsilon$ $\forall$ **users j** (guarantees ε differential privacy)
- The optimization is convex, can solve via interior point methods
  - Costly when S is large
  - We seek an efficient closed form for common strategies

Grouping Approach

- We observe that many strategies S can be broken into groups that behave in a symmetrical way
  - Sets of non-zero entries of rows in the group are pairwise disjoint
  - Non-zero values in group i have same magnitude C_i
- Many common strategies meet this grouping condition
  - Identity (I), Fourier (F), Marginals (C), Projections (P), Wavelets (H)
- Simplifies the optimization:
  - A single constraint over the ε_i's
  - New constraint: $\sum_{\text{Groups } i} C_i \, \varepsilon_i = \varepsilon$
  - Closed form solution via Lagrangian

$$\begin{pmatrix}
\frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} \\
\frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} \\
\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\
\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}}
\end{pmatrix}$$

Step 3: Optimal Recovery Matrix

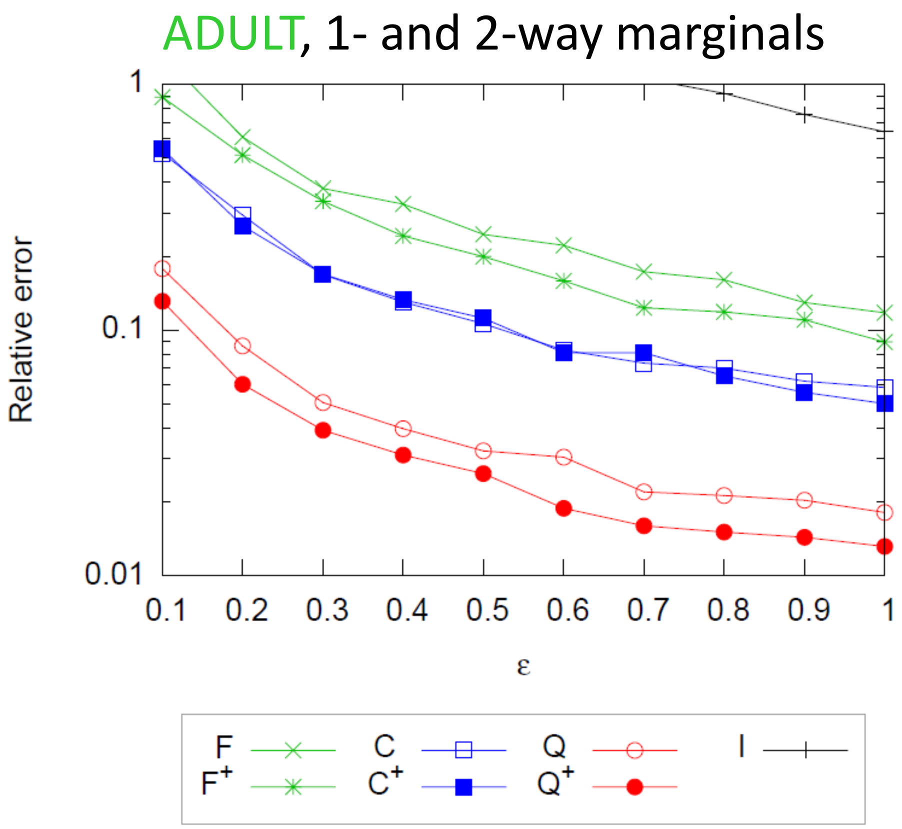


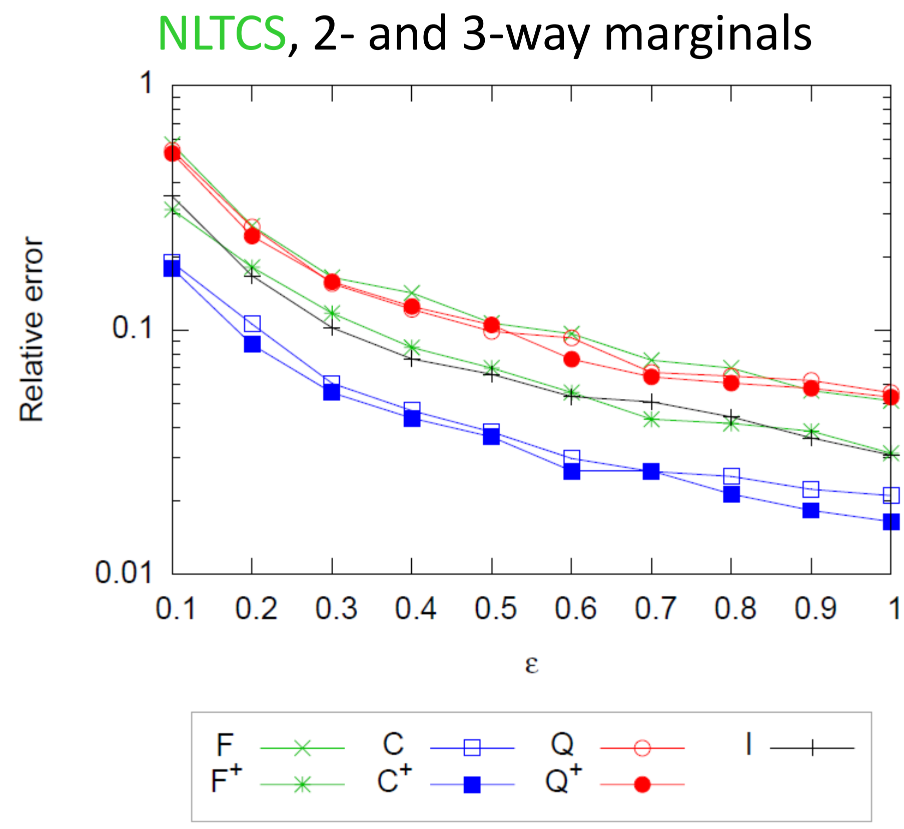
- Given Q, S, $\{\varepsilon_i\}$, find R so that $Q=RS$
  - Minimize the variance $Var(Rz) = Var(RSx + Rv) = Var(Rv)$
- Find an optimal solution by adapting Least Squares method
- This finds x' as an estimate of x given $z = Sx + v$
  - Define $\Sigma = Cov(z) = diag(2/\varepsilon_i^2)$ and $U = \Sigma^{-1/2} S$
  - OLS solution is $x' = (U^T U)^{-1} U^T \Sigma^{-1/2} z$
- Then $R = Q(S^T \Sigma^{-1} S)^{-1} S^T \Sigma^{-1}$
- Result: $y = Rz = Qx'$ is consistent—corresponds to queries on x'
  - R minimizes the variance
  - Special case: S is orthonormal basis ($S^T = S^{-1}$) then $R=QS^T$

Experimental Study

- Used two real data sets:
  - ADULT data – census data on 32K individuals (7 attributes)
  - NLTCS data– binary data on 21K individuals (16 attribues)
- Tried a variety of query workloads Q over these
  - Based on low-order k-way marginals (1-3-way)
- Compared the original and optimized strategies for:
  - Original queries, Q/Q^+
  - Fourier strategy F/F^+ [Barak et al. 07]
  - Clustered sets of marginals C/C^+ [Ding et al. 11]
  - Identity basis I

Experimental Results





- Optimized error gives constant factor improvement
- Time cost for the optimization is negligible on this data

Overall Process

- Ideal version: given query matrix Q, compute strategy S, recovery R and noise budget $\{\varepsilon_i\}$ to minimize Var(y)
  - Not practical: sets up a rank-constrained SDP [Li et al., PODS'10]
  - Follow the 3-step process instead

1. Fix S
2. Given query matrix Q, strategy S, compute optimal noise budgets $\{\varepsilon_i\}$ to minimize Var(y)
3. Given query matrix Q, strategy S and noise budgets $\{\varepsilon_i\}$, compute new recovery matrix R to minimize Var(y)

Advantages

- Best on datasets with many individuals (no dependence on how many)
- Best on large datasets (for small datasets, use [Li et al.])
- Best realtively small query workloads (for large query workloads, use multiplicative weights [Hardt, Ligett Mcsherry'12])
- Fairly fast (matrix multiplications and inversions)
  - Faster when S is e.g. Fourier, since can use FFT
  - Adds negligible computational overhead to the computation of queries themselves