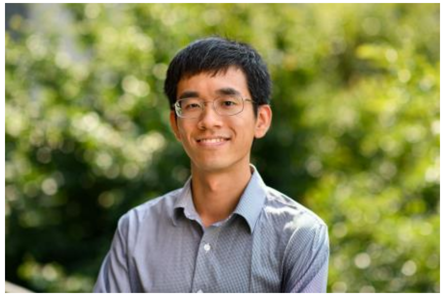
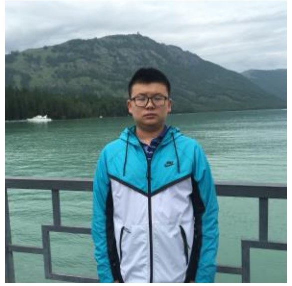
# A Strong Separation for Adversarially Robust $\ell_0$ Estimation for Linear Sketches



Elena Gribelyuk
Honghao Lin
David P. Woodruff
Huacheng Yu
Samson Zhou



# Streaming Model

- Input: Elements of an underlying data set $S$, which arrives sequentially
- Output: Evaluation (or approximation) of a given function
- Goal: Use space *sublinear* in the size $m$ of the input $S$

# Lots of problems…

- Graph problems: Matchings, MST, MAX-CUT
- Geometric problems: Clustering, facility location
- Statistical problems: Heavy-hitters, norm/moment estimation, quantile estimation
- Algebraic problems: Subspace embeddings, regression, low-rank approximation
- String problems: pattern matching, periodicity
- Others: CSPs, coding theory, submodular optimization, etc

# Distinct Elements

- Given a set $S$ of $m$ elements from $[n]$, let $f_i$ be the frequency of element $i$. (How often it appears)
- Let $F_0$ be the frequency moment of the vector:

$$F_0 = |\{i : f_i \neq 0\}|$$

- Goal: Given a set $S$ of $m$ elements from $[n]$ and an accuracy parameter $\varepsilon$, output a $(1+\varepsilon)$-approximation to $F_0$
- Motivation: Traffic monitoring

# Insertion-Only Streams

- Each update of the stream can only increase a coordinate of the frequency vector $x \in \mathbb{R}^n$

$$1\ 4\ 2\ 1\ 3\ 4\ 4\ 1 \rightarrow [3, 1, 1, 3, 0] := x$$

4 Distinct Elements

# Streaming Algorithms for $\ell_0$ Estimation

$(1+\varepsilon)$-multiplicative approximation streaming algorithms for distinct elements estimation using space:

- $O(\log n)$, assuming constant $\varepsilon$ and random oracle [FlajoletMartin85]
- $O(\log n)$, assuming constant $\varepsilon$ [AlonMatiasSzegedy99]
- $O\left(\frac{1}{\varepsilon^2}\log n\right)$ [Bar-YoseffJayramKumarSivakumar02]
- $O\left(\frac{1}{\varepsilon^2}\log\log n + \log n\right)$ assumes random oracle, additive error, i.e., HyperLogLog [FlajoletFusyGandouetMeunier07]
- $O\left(\frac{1}{\varepsilon^2} + \log n\right)$ [KaneNelsonWoodruff10], [Blasiok20]

# Streaming Algorithms for $\ell_0$ Estimation

- Sample the elements of the universe $[n]$ at rate $\frac{1}{2^i}$ into set $S_i$ for $i = 0,1, \ldots, O(\log n)$

- Pick set $S_i$ with roughly $\frac{1}{\varepsilon^2} \log n$ items in the stream

- Output $|S_i| \cdot 2^i$ as constant-factor approximation to the number of distinct elements

# Adversarially Robust Streaming

- Input: Elements of an underlying data set $S$, which arrives sequentially and *adversarially*
- Output: Evaluation (or approximation) of a given function
- Goal: Use space *sublinear* in the size $m$ of the input $S$

1

1

# Adversarially Robust Streaming

- Input: Elements of an underlying data set $S$, which arrives sequentially and *adversarially*
- Output: Evaluation (or approximation) of a given function
- Goal: Use space *sublinear* in the size $m$ of the input $S$

1 4

2

# Adversarially Robust Streaming

- Input: Elements of an underlying data set $S$, which arrives sequentially and *adversarially*
- Output: Evaluation (or approximation) of a given function
- Goal: Use space *sublinear* in the size $m$ of the input $S$

1 4 2

3

# Adversarially Robust Streaming

- Input: Elements of an underlying data set $S$, which arrives sequentially and *adversarially*
- Output: Evaluation (or approximation) of a given function
- Goal: Use space *sublinear* in the size $m$ of the input $S$

1 4 2 1

4

# Adversarially Robust Streaming

- Input: Elements of an underlying data set $S$, which arrives sequentially and *adversarially*
- Output: Evaluation (or approximation) of a given function
- Goal: Use space *sublinear* in the size $m$ of the input $S$

1 4 2 1

4

# Adversarially Robust Streaming

- Input: Elements of an underlying data set $S$, which arrives sequentially and *adversarially*
- Output: Evaluation (or approximation) of a given function
- Goal: Use space *sublinear* in the size $m$ of the input $S$

- Adversarially Robust: "Future queries may depend on previous queries"
- Motivation: Database queries, adversarial ML

# Robust Algorithms for $\ell_0$ Estimation

$(1+\varepsilon)$-multiplicative approximation adversarially robust streaming algorithms for distinct elements estimation using space:

- $\tilde{O}\left(\frac{1}{\varepsilon^3}\right) \cdot \mathrm{polylog}(n)$, via sketch switching [Ben-EliezerJayaramWoodruffYogev20]
- $\tilde{O}\left(\frac{1}{\varepsilon^{2.5}}\right) \cdot \mathrm{polylog}(n)$, via differential privacy [HassidimKaplanMansourMatiasStemmer20]
- $\tilde{O}\left(\frac{1}{\varepsilon^2}\right) \cdot \mathrm{polylog}(n)$, via difference estimators [WoodruffZhou21]

# Robust Algorithms for $\ell_0$ Estimation

$(1+\varepsilon)$-multiplicative appr ... streaming
algorithms fo

- $\tilde{O}\left(\frac{1}{\varepsilon^3}\right) \cdot$
  Elie
- $\tilde{O}\left(\frac{}{\varepsilon^2}\right)$
  [Has
- $\tilde{O}\left(\frac{1}{\varepsilon^2}\right) \cdot \text{poly}$ ... ors [ ... druffZhou21]

Nearly tight results for
adversarial robustness on
insertion-only streams

# Insertion-Deletion Streams

- Each update $u_t = (a_t, \Delta_t)$ can increase or decrease a coordinate $a_t \in [n]$ of the underlying frequency vector $x \in \mathbb{R}^n$ by $\Delta_t \in \mathbb{Z}$
- For simplicity, we assume $\Delta_t \in \{-1, +1\}$
- In the robust setting, each update $u_t$ can be chosen adversarially

# Insertion-Deletion Streams

- $\tilde{O}(m^{1/3})$ space algorithm for distinct element estimation, where $m$ is the length of the stream [Ben-EliezerEdenOnak22]
- Nothing known for constant-factor approximation in space polynomial in $n$

# Linear Sketch

- Algorithm maintains $Ax$ for a matrix $A$ throughout the stream
- The algorithm then outputs $f(Ax)$ for some post-processing function $f$

- All insertion-deletion streaming algorithms on a sufficiently long stream might as well be linear sketches [LiNguyenWoodruff14, AiHuLiWoodruff16]

# Reconstruction Attack on Linear Sketches

- Linear sketches are "not robust" to adversarial attacks, must use $\Omega(n)$ space [HardtWoodruff13]
- Approximately learn sketch matrix $A$, query something in the kernel of $A$
- Iterative process, start with $V_1, \ldots, V_r$
- Correlation finding: Find vectors weakly correlated with $A$ orthogonal to $V_{i-1}$
- Boosting: Use these vectors to find strongly correlated vector $v$
- Progress: Set $V_i = \text{span}(V_{i-1}, v)$

# Reconstruction Attack on Linear Sketches

- Attack randomly generates Gaussian vectors
- Analysis uses rotational invariance of Gaussians to observe which directions have larger $\ell_2$
- Attack ONLY works on *real-valued inputs* and ONLY against $\ell_2$ norm estimation

# Our Contribution

- There exists a constant $\varepsilon = \Omega(1)$ such that any linear sketch that produces $(1+\varepsilon)$-approximation to $\ell_0$ on an adversarial insertion-deletion stream on universe $n$ requires $\mathrm{poly}(n)$ rows

- There exists a constant $\varepsilon = \Omega(1)$ such that any linear skech that produces $(1+\varepsilon)$-approximation to $\ell_0$ on an adversarial insertion-deletion stream using $r \ll n$ rows can be broken in $\tilde{O}(r^8)$ queries.

# Upcoming

- Attack intuition

# Questions?

# Gap $\ell_0$ Norm Problem

- Let $\alpha$ and $\beta$ be fixed constants
- Distinguish between the case where $\|x\|_0 < \alpha n$ or $\|x\|_0 > \beta n$
- Algorithm allowed to arbitrarily output when neither case holds
- Any multiplicative $(1+\varepsilon)$-approximation algorithm to $\ell_0$ can solve the gap problem, for sufficiently small $\varepsilon \approx \sqrt{\frac{\beta}{\alpha}} - 1$

# Attack Outline

- Intuitively, a sketch matrix $A$ may preserve a “large” amount of information about some coordinates and a “small” amount of information about other coordinates
  - There can be a row of $A$ that is nonzero in only a single column
  - $A$ can be sampled such that a random set of $O(1)$ coordinates has large information
  - There can be coordinates that only appears in columns with a large number of nonzero entries

$$Ax := \overset{A}{\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & -1 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}} \underset{x}{\begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$x := [0,1,0,1,0,0,0]$$

$$A_1 := [0,0,0,1,0,0,0] \rightarrow \langle A_1, x \rangle = 1$$

$$A_2 := [1,-1,1,1,0,1,1] \rightarrow \langle A_2, x \rangle = 0$$

# Attack Outline

- Adversary wants to gradually learn the sketching matrix

- Strategy:
    1. Iteratively identify the significant coordinates and set them to zero in all future queries
    2. After we have learned all such coordinates, the query algorithm must rely on the other coordinates, which the sketch $Ax$ only has “small” information

# Attack Outline

- Consider an extreme example where the sketch $Ax$ is a subset $S$ of $r$ coordinates of $x$, unknown to the adversary

- Attack:
  1. Identify $S$
  2. Place zeros in $S$ and nonzeros elsewhere

# Interactive Fingerprinting Code Problem

- An algorithm $\mathcal{P}$ selects a set $S \subset [n]$ of coordinates unknown to the fingerprinting code $\mathcal{F}$

- $\mathcal{F}$ must identify $S$ by making adaptive queries $c^t \in \{\pm 1\}^n$
- $\mathcal{P}$ must answer consistently with some coordinate in $c^t$, i.e., $a^t = c_i^t$ for some $i \in [n]$
- BUT $\mathcal{P}$ can only observe $c_i^t$ for $i \in S$ → needs to distinguish between inputs that are all zeros and all ones restricted to $S$

- Used for watermarking, traitor-tracing schemes [BonehShaw98]

# Interactive Fingerprinting Codes

- There exists an interactive fingerprinting code with length $\tilde{O}(n^2)$ [SteinkeUllman15]
- Gap $\ell_0$ norm problem needs to distinguish between $\|x\|_0 < \alpha n$ or $\|x\|_0 > \beta n$
- Stronger requirement than fingerprinting code (which just needs to distinguish between all zeros and all ones)

# Significant Coordinates (I)

- How to quantify significant coordinates?
- $i$ is significant if there exists:
  - an elementary vector $e_i$ that is a row of $A$

$$Ax := \overset{A}{\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 999 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}} \; \begin{array}{c} \\ \\ \\ \\ x \end{array} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1000 \end{bmatrix}$$

$$x := [0,1,0,1,0,0,0]$$

$$A_1 := [0,0,0,1,0,0,0] \rightarrow \langle A_1, x \rangle = 1$$

$$A_2 := [1,999,1,1,0,1,1] \rightarrow \langle A_2, x \rangle = 1000$$

# Significant Coordinates (II)

- Since the algorithm has $Ax$, it can recover $y^\top Ax$ for any vector $y \in \mathbb{R}^r$
- If there exists $y \in \mathbb{R}^r$ such that $(y^\top Ax)_i^2 \geq \frac{1}{s}\|y^\top A\|_2^2$, then $i$ is significant (leverage score of column $i$ is large)

# Significant Coordinates (II)

- How to quantify significant coordinates?
- $i$ is significant if there exists:
  - an elementary vector $e_i$ that is a row of $A$
  - $y \in \mathbb{R}^r$ such that $(y^\top A)_i^2 \geq \frac{1}{s} \|y^\top A\|_2^2$

$$A_1 := [10, 10, 10, 10, 10, 10, 3] \rightarrow \langle A_1, x \rangle = 103$$

$$x := [2,3,5,0,0,0,1]$$

Reveals information about $x_n$ modulo 10

$$A_1 := \left[1, 1, 1, 1, 1, 1, \frac{3}{10}\right] \rightarrow \langle A_1, x \rangle = 10.3$$

$$x := [2,3,5,0,0,0,1]$$

**Fractional** part of $(y^\top A)_n$ is large, for $y$ selecting the first row of $A$

# Significant Coordinates (III)

- $i$ is significant if there exists $y \in \mathbb{R}^r$ such that $(\mathrm{FRAC}(y^\top Ax)_i)^2 \geq \frac{1}{s}\sum_i (\mathrm{FRAC}(y^\top Ax)_i)^2$

# Significant Coordinates

- How to quantify significant coordinates?
- $i$ is significant if there exists:
  - an elementary vector $e_i$ that is a row of $A$
  - $y \in \mathbb{R}^r$ such that $(y^\top A)_i^2 \geq \frac{1}{s}\|y^\top A\|_2^2$
  - $y \in \mathbb{R}^r$ such that $(\mathrm{FRAC}(y^\top Ax)_i)^2 \geq \frac{1}{s}\sum_i(\mathrm{FRAC}(y^\top Ax)_i)^2$

# Pre-processing the Sketch Matrix

- The algorithm has access to linear sketch $Ax$
- Pre-process the matrix $A$ into a larger matrix $A'$ that separates the significant coordinates
- Only gives the algorithm "more" information

$$\underbrace{\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 999 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}}_{A} \longrightarrow \underbrace{\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{A'}$$

# Pre-processing the Sketch Matrix

- Resulting matrix $A'$ is a combination of a sparse part $S$ and a dense part $D$

$$A' = \begin{bmatrix} S \\ D \end{bmatrix}$$

$$A' = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

# Pre-processing the Sketch Matrix

- Sparse part $S$ has at most one nonzero entry per column
- Dense part $D$ has no significant columns
- Show only $O(rs\log n)$ rows added to $A$

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

- Note that if there were no dense part, we can use fingerprinting code to attack $S$

# Overall Attack

1. Pre-process the matrix $A$ into a matrix $A'$ that is a combination of a sparse part $S$ and a dense part $D$
2. Attack sparse part $S$ using fingerprinting code
3. Attack dense part $D$

# Upcoming

- Attack on dense part

# Questions?

# Attacking the Dense Part

- Design a family of distributions $\mathcal{D}$ over $[-R, \dots, -1,0,1, \dots, R]$ with $R = \mathrm{poly}(n)$ such that:
  - For $D_p \in \mathcal{D}$ with $p \in [a, b]$, we have $\Pr_{X \sim D_p}[X = 0] = p$
  - For any $q, p \in [a, b]$, the total variation distance between $Dx_p$ and $Dx_q$ is small, i.e., $\frac{1}{\mathrm{poly}(n)}$

# Attacking the Dense Part

- Design a family of distributions $\mathcal{D}$ over $[-R, \ldots, -1,0,1, \ldots, R]$ with $R = \mathrm{poly}(n)$ such that:
  - For $D_p \in \mathcal{D}$ with $p \in [a, b]$, we have $\Pr_{X \sim D_p}[X = 0] = p$
  - For any $q, p \in [a, b]$, the total variation distance between $Dx_p$ and $Dx_q$ is small, i.e., $\frac{1}{\mathrm{poly}(n)}$

- If $x \sim D_p^n$, then $\mathrm{Ex}[\|x\|_0] = pn$ and if $x \sim D_q^n$, then $\mathrm{Ex}[\|x\|_0] = qn$, *but the marginal distribution of $Dx$ is nearly identical for $x \sim D_p^n$ and $x \sim D_q^n$, so the algorithm must use $Sx$*

# Overall Attack

1. Pre-process the matrix $A$ into a matrix $A'$ that is a combination of a sparse part $S$ and a dense part $D$
2. Attack sparse part $S$ using fingerprinting code
3. Attack dense part $D$ using the family of distributions $\mathcal{D}$

# Bounding the Total Variation Distance

- Let $P$ be the probability distribution corresponding to $Dx_p$ and $Q$ be the probability distribution corresponding to $Dx_q$
- To bound the total variation distance between $P$ and $Q$, note

$$|P(x) - Q(x)| = \left| \frac{1}{(2\pi)^r} \int_{u \in [-2\pi, 2\pi]^r} e^{i\langle u, x\rangle} \left( \hat{P}(u) - \hat{Q}(u) \right) \mathrm{d}u \right|$$

$$\leq \frac{1}{(2\pi)^r} \int_{u \in [-2\pi, 2\pi]^r} \left| \hat{P}(u) - \hat{Q}(u) \right| \mathrm{d}u$$

# Bounding the Total Variation Distance

- For a symmetric distribution, we can write

$$\hat{P}(u) = \mathrm{Ex}_{z\sim P}\left[e^{-i\langle u,z\rangle}\right]$$

$$= \prod_{j\in[n]} \sum_{k\geq 0} \left(M_p(2k)\right) \cdot f(D, u, k)$$

where $M_p(2k) = \left(\sum_{m\geq 0} P_m m^{2k}\right)$ is the $2k$-th moment of the distribution and $f$ is a rapidly decaying function independent of $P$

# Bounding the Total Variation Distance

- To analyze the total variation distance, we have

$$\left|\hat{P}(u) - \hat{Q}(u)\right| = \prod_{j \in [n]} \sum_{k \geq 0} \left(M_p(2k) - M_q(2k)\right) \cdot f(D, u, k)$$

so if the first $2k$ moments of the distributions of $P$ and $Q$ match, for a sufficiently large $k$, then the TVD is small

# Constructing Hard Distributions

- Design a family of distributions $\mathcal{D}$ over $[-R, \ldots, -1,0,1, \ldots, R]$ with $R = \text{poly}(n)$ such that:
  - For $D_p \in \mathcal{D}$ with $p \in [a, b]$, we have $\Pr_{X \sim D_p}[X = 0] = p$
  - For any $q, p \in [a, b]$, the total variation distance between $Dx_p$ and $Dx_q$ is small, i.e., $\frac{1}{\text{poly}(n)}$
  - The first $2k$ moments of the distributions of $D_p$ and $D_q$ match

# Moment Matching

- Want $\mathrm{Ex}_{X\sim D_p}[X^k] = \mathrm{Ex}_{X\sim D_q}[X^k]$ for all $k \leq K = O(r\log n)$
- There exists [LarsenWeinsteinYu20] a polynomial $Q$ such that $|Q(0)| = \Omega(1)$ and for all $t < R - \deg(Q)$ :

$$\sum_{i=0}^{R}\left|\binom{R}{i}\cdot Q(i)\right| = O(1)$$

$$\sum_{i=0}^{R}(-1)^i\binom{R}{i}\cdot Q(i)\cdot i^t = 0$$

Set $D_p(i)$ to be

$D(i) + c_p \cdot (-1)^i\ (-1)^i\binom{R}{i}\cdot Q(i)\cdot i^t$

# Overall Attack

1. Pre-process the matrix $A$ into a matrix $A'$ that is a combination of a sparse part $S$ and a dense part $D$
2. Attack sparse part $S$ using fingerprinting code
3. Attack dense part $D$ using the family of distributions $\mathcal{D}$

# Main Results

- There exists a constant $\varepsilon = \Omega(1)$ such that any linear sketch that produces $(1+\varepsilon)$-approximation to $\ell_0$ on an adversarial insertion-deletion stream on universe $n$ requires $\text{poly}(n)$ rows

- There exists a constant $\varepsilon = \Omega(1)$ such that any linear skech that produces $(1+\varepsilon)$-approximation to $\ell_0$ on an adversarial insertion-deletion stream using $r \ll n$ rows can be broken in $\tilde{O}(r^8)$ queries.

# Other Results

- Any linear skech that produces $1.1$-approximation to $\ell_0$ on an adversarial insertion-deletion stream using $r \ll n$ rows can be broken in $\tilde{O}(r^3)$ queries, if the calculations are performed on finite fields $\mathbb{F}_p$

- There exists an attack on any real-valued linear skech that produces $O(1)$-approximation to $\ell_0$ on an adversarial insertion-deletion stream with $r \ll n$ rows, using $\mathrm{poly}(r)$ queries

# Future Directions

- Attacks with a smaller number of queries?

- Attacks against pseudo-deterministic algorithms?

# Future Directions


Attacks on linear-sketches for $\ell_0$ estimation on adversarial insertion-deletion streams

Attacks on streaming algorithms for $\ell_0$ estimation on adversarial insertion-deletion streams

Attacks on linear-sketches for $\ell_p$ estimation on adversarial insertion-deletion streams

Attacks on streaming algorithms for $\ell_p$ estimation on adversarial insertion-deletion streams