Adversarially Robust Streaming Algorithms

Omri Ben-Eliezer
Part 1: Model and motivation
Streaming: The classical model

- Stream \((x_1, \Delta_1), \ldots, (x_m, \Delta_m)\) of updates, \(x_i \in [n], \Delta_i \in \mathbb{R}\)
- Goal: compute \((1 + \epsilon)\)-approx of \(f\) over (frequency vector \(v\) of) stream
  \((f = \#\) distinct elements / \(\ell_2\)-norm / entropy / heavy hitters / ...\)
Streaming: The classical model

- Stream \((x_1, \Delta_1), \ldots, (x_m, \Delta_m)\) of updates, \(x_i \in [n], \Delta_i \in \mathbb{R}\)
- Goal: compute \((1 + \epsilon)\)-approx of \(f\) over (frequency vector \(v\) of) stream
  \((f = \# \text{ distinct elements} / \ell_2\text{-norm} / \text{entropy} / \text{heavy hitters} / \ldots)\)
Streaming: The classical model

• Stream \((x_1, \Delta_1), \ldots, (x_m, \Delta_m)\) of updates, \(x_i \in [n], \Delta_i \in \mathbb{R}\)

• Goal: compute \((1 + \epsilon)\)-approx of \(f\) over (frequency vector \(v\) of) stream
  \((f = \# \text{ distinct elements} / \ell_2\text{-norm} / \text{entropy} / \text{heavy hitters} / \ldots)\)
Streaming: The classical model

- Stream $(x_1, \Delta_1), \ldots, (x_m, \Delta_m)$ of updates, $x_i \in [n], \Delta_i \in \mathbb{R}$
- Goal: compute $(1 + \epsilon)$-approx of $f$ over (frequency vector $v$ of) stream
  ($f = \#$ distinct elements / $\ell_2$-norm / entropy / heavy hitters / ...)

delete seven units of $x_2$
Streaming: The classical model

- Stream \((x_1, \Delta_1), \ldots, (x_m, \Delta_m)\) of updates, \(x_i \in [n], \Delta_i \in \mathbb{R}\)
- Goal: compute \((1 + \epsilon)\)-approx of \(f\) over (frequency vector \(v\) of) stream
  \((f = \# \text{ distinct elements} / \ell_2\text{-norm} / \text{entropy} / \text{heavy hitters} / \ldots)\)
Streaming: The classical model

- Stream \((x_1, \Delta_1), \ldots, (x_m, \Delta_m)\) of updates, \(x_i \in [n], \Delta_i \in \mathbb{R}\)
- Goal: Compute \((1 + \epsilon)\)-approx of \(f\) over (frequency vector \(v\) of) stream
  \(f = \#\) distinct elements / \(\ell_2\)-norm / entropy / heavy hitters / ...

\[ F_p(v) = \sum_{i=1}^{n} v_i^p \]
Randomized streaming algorithms

<table>
<thead>
<tr>
<th>Problem</th>
<th>Space complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>distinct elements ($F_0$)</td>
<td>$\tilde{O}(\epsilon^{-2} + \log n)$ [IW’03, KNW’10, B’18, W’23, …]</td>
</tr>
<tr>
<td>$F_2$-estimation</td>
<td>$\tilde{O}(\epsilon^{-2} \log n)$ [AMS’96, KNW’10, BDN’17, …]</td>
</tr>
<tr>
<td>$F_2$-heavy hitters</td>
<td>$\tilde{O}(\epsilon^{-2} \log^2 n)$ [BCINWW’17]</td>
</tr>
<tr>
<td>entropy estimation</td>
<td>$\tilde{O}(\epsilon^{-2} \log^3 n)$ [CC’13]</td>
</tr>
</tbody>
</table>

*Deterministic algorithms: exponentially less efficient*
Randomized streaming algorithms

<table>
<thead>
<tr>
<th>Problem</th>
<th>Space complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>distinct elements ($F_0$)</td>
<td>$\tilde{O}(\epsilon^{-2} + \log n)$ [IW’03, KNW’10, B’18, W’23, ...]</td>
</tr>
<tr>
<td>$F_2$-estimation</td>
<td>$\tilde{O}(\epsilon^{-2} \log n)$ [AMS’96, KNW’10, BDN’17, ...]</td>
</tr>
<tr>
<td>$F_2$-heavy hitters</td>
<td>$\tilde{O}(\epsilon^{-2} \log^2 n)$ [BCINWW’17]</td>
</tr>
<tr>
<td>entropy estimation</td>
<td>$\tilde{O}(\epsilon^{-2} \log^3 n)$ [CC’13]</td>
</tr>
</tbody>
</table>

But... proofs assume stream is fixed in advance!

*Deterministic algorithms: exponentially less efficient
Streaming: The adaptive/adversarial model

- Feedback loop
- World’s goal: break $(1 + \epsilon)$-approximation
Streaming: The **adaptive/adversarial** model

- Feedback loop
- World’s goal: break \((1 + \epsilon)\)-approximation
Streaming: The adaptive/adversarial model

- Feedback loop
- World’s goal: break \((1 + \epsilon)\)-approximation
Streaming: The adaptive/adversarial model

- Feedback loop
- World’s goal: break \((1 + \epsilon)\)-approximation
Streaming: The adaptive/adversarial model

- Feedback loop
- World’s goal: break \((1 + \epsilon)\)-approximation
Streaming: The adaptive/adversarial model

- Feedback loop
- World’s goal: break $(1 + \epsilon)$-approximation
Streaming: The **adaptive/adversarial** model

- Feedback loop
- World’s goal: break \((1 + \epsilon)\)-approximation
Key streaming technique: Linear sketching

**$F_2$-estimation** [Alon-Matias-Szegedy’96]:

Use $|Sv|_2$ to estimate $|v|_2$.

Given stream update $(x_t, \Delta)$, update: $Sv \leftarrow Sv + \Delta \cdot S_{*,x_t}$

$$
\mathbb{E}_S[|Sv|_2^2] = \frac{1}{\epsilon^2} \mathbb{E}_S \left[ \left( \sum_{j=1}^{n} S_{1j}v_j \right)^2 \right] = \mathbb{E}_S \left[ \sum_{j=1}^{n} v_j^2 + \sum_{i \neq j} S_{1i}S_{1j}v_iv_j \right] = |v|_2^2
$$
Linear sketching: not robust [Hardt-Woodruff’13]
Also [AGM’12, BJWY’20, CLNSSS’22, CNSS’23,…]

Use $|Sv|^2$ to estimate $|v|^2$.

What if $S_1^*$ correlations learned by adversary?

$$
\mathbb{E}_S[|Sv|^2] = \frac{1}{\epsilon^2} \mathbb{E}_S \left[ \left( \sum_{j=1}^{n} S_{1j}v_j \right)^2 \right] = \mathbb{E}_S \left[ \sum_{j=1}^{n} v_j^2 + \sum_{i \neq j} S_{1i}S_{1j}v_iv_j \right] \neq |v|^2
$$
Linear sketching: not robust [Hardt-Woodruff’13]
Also [AGM’12, BJWY’20, CLNSSS’22, CNSS’23, …]

Use $|Sv|_2$ to estimate $|v|_2$.

$S = \begin{bmatrix} s_{11} & \ldots & \ldots \\ \vdots & \ddots & \vdots \\ \ldots & \ldots & s_{nn} \end{bmatrix}$

• Actual attack of [HW13] works for general $r \times n$ sketch matrices and breaks $B$-approx. in poly$(rB)$ rounds

• Proof idea: find vectors correlated with row space, boost, “peel” dimension. Iterate until remainder of sketch is constant dimensional
The need for robustness - examples

Actual adversarial attacks

Search Engine Optimization (SEO)

Gradient descent

Alexander "alech" Klink
n.runs AG

Julian "zeri" Wälde
TU Darmstadt
#hashDoS
Part 2: Frameworks

How to make streaming algorithms robust?

• Sketch Switching [B., Jayaram, Woodruff, Yogev ‘20]
• Differential privacy [Hassidim, Kaplan, Mansour, Matias, Stemmer ‘20]
• Difference estimators [Woodruff, Zhou ‘21]
• “Best of both worlds” [Attias, Cohen, Shechner, Stemmer ‘23]
Sketch switching [B., Jayaram, Woodruff, Yogev ‘20]

- Run $\lambda$ copies of algorithm
- Maintain frozen-output visible copy + active copy.
- If frozen output incorrect:
  - $\text{visible copy} \leftarrow \text{active copy}$
  - $\text{active copy} \leftarrow \text{next copy}$

\[ (x_1, \Delta_1), (x_2, \Delta_2), \ldots \]
Key notion: **Flip number**

\[
\lambda_{\epsilon, m}(f) := \max \left\{ k \mid \exists j_1, \ldots, j_k \in [m] \forall l \in [k - 1] : f(v^{(j_{l+1})}) \notin (1 \pm \epsilon) \cdot f(v^{(j_l)}) \right\}
\]

**Theorem ([BJWY’20]):**
Let \( \mathcal{A} \) be a (“classical”) \( \epsilon \)-tracking streaming algorithm for \( f \).

There exists an adversarially robust algorithm \( \mathcal{A}' \) for \( \epsilon \)-tracking \( f \) using space \( \lambda(f) \cdot \text{Space}(\mathcal{A}) \).
Key notion: **Flip number**

\[ \lambda_{\epsilon, m}(f) := \max \left\{ k \mid \exists j_1, \ldots, j_k \in [m] \forall l \in [k - 1] : f(v^{(j_{l+1})}) \notin (1 \pm \epsilon) \cdot f(v^{(j_l)}) \right\} \]

**Theorem** ([BJWY’20]):
Let \( \mathcal{A} \) be a (“classical”) \( \epsilon \)-tracking streaming algorithm for \( f \).

There exists an adversarially robust algorithm \( \mathcal{A}' \) for \( \epsilon \)-tracking \( f \) using space \( \lambda(f) \cdot \text{Space}(\mathcal{A}) \).

\[ f(\{x_1, \ldots, x_m\}) \leq n \]

\[ f(\{x_1\}) = 1 \]

\[ f = \text{distinct elements} \]

Insertion only setting \( \Rightarrow \)

\[ \lambda(f) = O \left( \frac{\log n}{\epsilon} \right) \]
Key notion: **Flip number**

**Insertion only** ($\Delta > 0$):

\[ \lambda = O\left(\frac{\log n}{\varepsilon}\right) \]

**Turnstile model** (general $\Delta$, insertion/deletion):

\[ \lambda = m \text{ in worst case!} \]
Sketch switching: Proof idea

1. Assume deterministic (Yao’s minimax [Yao’77])

2. Only need active copy $A_i$ to be correct over specific (fixed) stream:

   “$A_i$-oblivious part”
   $+$
   “frozen output part”
Generic frameworks: flip number overhead

$O(\lambda)$ [BJWY’20]

$\tilde{O}(\sqrt{\lambda})$
Differential privacy *

$O(\epsilon \cdot \lambda)$
Difference estimators **

$\tilde{O}(\sqrt{\epsilon \cdot \lambda})$
“best of both worlds” ***

$\tilde{\Omega}(\sqrt{\lambda})$
for specific problem ****

* Hassidim-Kaplan-Mansour-Matias-Stemmer ’20

** Woodruff-Zhou ’21

*** Attias-Cohen-Shechner-Stemmer ’23

**** Kaplan-Mansour-Nissim-Stemmer ’21
Differential privacy framework
[Hassidim-Kaplan-Mansour-Matias-Stemmer’20]

$(x_1, \Delta_1), (x_2, \Delta_2), ...$

$A_1$

11 → Private threshold
(sparse vector technique)

$A_2$

15

$A_t$

$13$

$t \approx \sqrt{\lambda}$ copies

Private median
Differential privacy framework
[Hassidim-Kaplan-Mansour-Matias-Stemmer’20]

Why it works:

• Key idea -- **Advanced composition** [Dwork-Rothblum-Vadhan’10] in differential privacy supports $\approx \frac{1}{\varepsilon^2}$ adaptive interactions provided “privacy level” $\varepsilon$.

• Interaction with each flip $\Rightarrow \varepsilon \approx \frac{1}{\sqrt{\lambda}}$

• $\varepsilon$-DP median requires $\approx \frac{1}{\varepsilon} \approx \sqrt{\lambda}$ copies to be accurate

*Extremely useful* in applications!
(e.g., robust count sketch [Cohen, Lyu, Nelson, Sarlós, Shechner, Stemmer ’22], robust dynamic graph algorithms [Beimel, Kaplan, Mansour, Nissim, Saranurak, Stemmer ‘22], ...)
Difference estimators framework

[Woodruff, Zhou ‘21]

- Key notion: $\varepsilon$-difference estimator for vectors $u$ and $v$ approximates $f(u + v) - f(u)$ to $\pm \varepsilon \cdot f(u)$ error

- Framework stitches difference estimators at different scales
Difference estimators framework

[Woodruff, Zhou ‘21]

- Key notion: $\varepsilon$-difference estimator for vectors $u$ and $v$ approximates $f(u + v) - f(u)$ to $\pm \varepsilon \cdot f(u)$ error

- Framework stitches difference estimators at different scales

- Near-optimal space complexity for insertion only + sliding window model, resolving [Braverman-Ostrovsky’07]

<table>
<thead>
<tr>
<th>Problem</th>
<th>[BJYW’20]</th>
<th>[HKMMS’20]</th>
<th>[WZ’21]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distinct Elements ($F_0$)</td>
<td>$\tilde{O}\left(\frac{1}{\varepsilon^3} + \frac{\log n}{\varepsilon}\right)$</td>
<td>$O\left(\frac{\log^{1.5} n}{\varepsilon^{2.5}} + \frac{\log^{2.5} n}{\varepsilon^{1.5}}\right)$</td>
<td>$\tilde{O}\left(\frac{1}{\varepsilon^2} + \frac{\log n}{\varepsilon}\right)$</td>
</tr>
<tr>
<td>$F_p$ estimation, $p \in (0,2]$</td>
<td>$\tilde{O}\left(\frac{\log n}{\varepsilon^3}\right)$</td>
<td>$O\left(\frac{\log^{1.5} n}{\varepsilon^{2.5}}\right)$</td>
<td>$\tilde{O}\left(\frac{\log n}{\varepsilon^2}\right)$</td>
</tr>
<tr>
<td>$F_p$ with deletions, flip number $\lambda$</td>
<td>$\tilde{O}\left(\frac{\lambda \cdot \log^2 n}{\varepsilon^2}\right)$</td>
<td>$O\left(\frac{\log^3 n \sqrt{\lambda}}{\varepsilon^2}\right)$</td>
<td>$\tilde{O}\left(\frac{\lambda \cdot \log^2 n}{\varepsilon}\right)$</td>
</tr>
</tbody>
</table>
Is flip number always the right parameter?
[B., Eden, Onak '22]: No. for insertion-deletion streams ($\lambda = m$) we get

$\tilde{O}(m^{1/3})$ for distinct elements, $\tilde{O}(m^{2/5})$ for $\ell_2$-estimation

Technique: differential privacy + sparse recovery
[B., Eden, Onak ‘22]: No. for insertion-deletion streams ($\lambda = m$) we get

\[ \tilde{O}(m^{1/3}) \] for distinct elements, \[ \tilde{O}(m^{2/5}) \] for $\ell_2$-estimation

Technique: differential privacy + sparse recovery

Best known, but exponentially worse than static.
Dense-sparse tradeoff [B., Eden, Onak ‘22]

privacy regime, \( \lambda \leq m^{2/3} \)

Sparse recovery regime, use “for all” solution [Gilbert, Strauss, Tropp, Vershynin ‘07]
Part 3: Lessons
Lesson 1: Sampling over sketching

• **Linear sketching**: extremely useful in static settings, but breaks in $r^{O(1)}$ rounds in adversarial settings [HW’13, BJWY’20, CLNSSS’22, CNSS’23,…]

• **Random sampling**: another fundamental technique, more reliable in adversarial settings?

$$S = \begin{bmatrix} s_{11} & \ldots & \vdots & \ldots & s_{1n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ s_{r1} & \ldots & \vdots & \ldots & s_{rn} \end{bmatrix}$$
Lesson 1: Sampling over sketching

How many uniform samples needed so that sample will be representative of data?

Oblivious case: $\Theta \left( \frac{\text{VC-DIM}}{\epsilon^2} \right)$

[B., Yogev ‘19] Not enough for robust case!

But $\text{VC} \cdot \frac{\log m}{\epsilon^2}$ suffices.

[Alon, B., Dagan, Moran, Naor, Yogev ‘21]:

$\Theta \left( \frac{\text{Littlestone-DIM}}{\epsilon^2} \right)$
Lesson 1: Sampling over sketching

[Braverman, Hassidim, Matias, Schain, Silwal, Zhou ‘21]:

- Importance sampling & merge and reduce robust “for free”

⇒ Robustness of many existing algorithms!

<table>
<thead>
<tr>
<th>Meta-approach</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Merge and reduce (Theorem 1.1)</td>
<td>Coreset construction, support vector machine, Gaussian mixture models, $k$-means clustering, $k$-median clustering, projective clustering, principal component analysis, $M$-estimators, Bayesian logistic regression, generative adversarial networks (GANs), $k$-line center, $j$-subspace approximation, Bregman clustering</td>
</tr>
<tr>
<td>Row sampling (Theorem 1.2)</td>
<td>Linear regression, generalized regression, spectral approximation, low-rank approximation, projection-cost preservation, $L_1$-subspace embedding</td>
</tr>
<tr>
<td>Edge sampling (Theorem 1.3)</td>
<td>Graph sparsification</td>
</tr>
</tbody>
</table>

(Taken from [BHMSSZ’21])
Lesson 1: Sampling over sketching

Formal separation between “sketching type” vs. “sampling type” algs?

Yes! For ”missing item finding” [Stoeckl’23, Chakrabarti-Stoeckl’23, Magen’24] (see also Menuhin-Naor’22)

- Random oracle = fixed random string, no storage cost
- Random tape = randomness on the fly, costs to store it
- Random seed = randomness at beginning
- Pseudo-deterministic [GGMW’19] = “deterministic with prob. $1 - \delta$”

(Taken from [CS’23])
Speaking about separations: robust graph coloring

**Static** streams: $(\Delta + 1)$-coloring in $\tilde{O}(n)$ space [Assadi, Chen, Khanna’19]

**Adversarially robust** [Chakrabarti, Ghosh, Stoeckl’22], [Assadi, Chakrabarti, Ghosh, Stoeckl’23]

”No free lunch” lower bound 😐:

$O(\Delta)$ colors requires $\Omega(n\Delta)$ space, $O(n)$ space implies $\Omega(\Delta^2)$ colors

Upper bounds via sketch switching + additional techniques:

$O(\Delta^{2.5})$ colors in $\tilde{O}(n)$ space, or $O(\Delta^2)$ colors in $\tilde{O}(n^{\Delta^{1/3}})$ space

**Deterministic** [Assadi, Chen, Sun ‘22]: $\tilde{O}(n)$ space implies $\exp(\Delta^{\Omega(1)})$ colors!
Lesson 2: White box vs black box

**White box**: adversary can see internal state of algorithm

[Ajtaï, Braverman, Jayram, Silwal, Sun, Woodruff, Zhou ‘22], [Feng, Woodruff ‘23]

< 01101011 ... >
Lesson 2: White box vs black box

**White box:** adversary can see internal state of algorithm [Ajtai, Braverman, Jayram, Silwal, Sun, Woodruff, Zhou ‘22], [Feng, Woodruff ‘23]

• Algorithms (e.g., $n^\epsilon$-approx. $F_0$) from cryptographic primitives (e.g., SIS) and computationally-bounded adversary assumptions

• $\Omega(n)$ lower bound for approximate white-box $F_p$-estimation via reductions from deterministic GAP-EQUALITY problem in communication complexity
Lesson 3: Connections and applications [partial list]

• **Techniques/notions:**
  “new”: differential privacy [HKMMS’20], cryptography [ABJSSW’21], statistics / learning theory [ABDMNY’21], adaptive data analysis [KMNS’21]
  “standard”: sparse recovery [BEO’22], communication complexity [CGS’22]
  + novel techniques like difference estimators [WZ’21]

• **Applications/implications** in:
  dynamic data structures [BKMNSS’22], graph coloring [CGS’22, ACGS’23], sliding window streaming [WZ’21], machine learning [BHMSSZ’21, WZZ’23, CSWZZZ’23], analytics [RZCP’24], …

• Many connections waiting to be explored!
  A few examples: algorithmic game theory, cognitive science, reinforcement learning
Lesson 4 (personal): randomness of history

Wenn ich nur erst die Sätze habe! Die Beweise werde ich schon finden.
-- Bernhard Riemann (1826-1866)
A few (important/favorite) open questions

• $m^{o(1)}$ space for robust streaming $F_0$ or $F_2$?
• Alternatively, $m^{\Omega(1)}$ lower bound for same problems? (Not known even for pseudo-deterministic algorithms!)

• Interesting beyond worst case adversarial models [Cherapanamjeri, Silwal, Woodruff, Zhang, Zhang, Zhou’23, Sedigurchi, Stemmer, Shechner’23]
  • streaming vs. streaming?
  • Distribution-maintaining adversaries?
  • Models for robust dynamic graph algorithms?
• “Practical deployment”