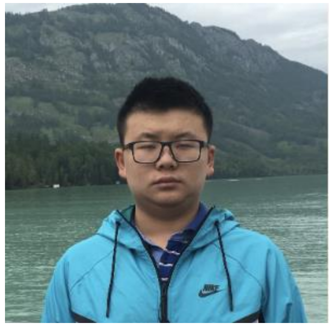
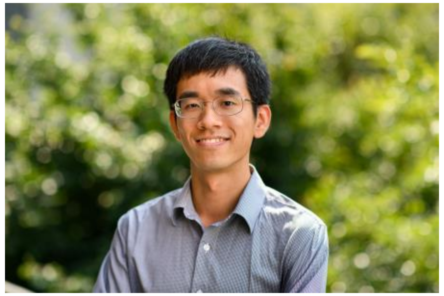
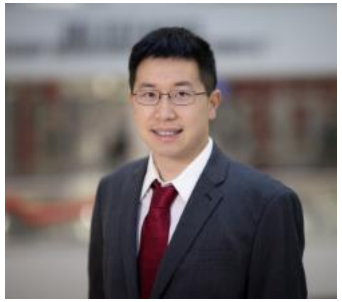
Adversarial Robustness on Insertion-Deletion Streams

Elena Gribelyuk
Princeton

Honghao Lin
CMU

David P. Woodruff
CMU

Huacheng Yu
Princeton

Samson Zhou
Texas A&M

Streaming Model

Massive Data Streams

Internet traffics

Sensor networks

Stock Markets

Streaming Model

$(a_1, w_1), (a_2, w_2), \cdots, (a_m, w_m)$

- There is an underlying frequency vector $x \in \mathbb{Z}^n$
  - Initialized to 0^n;
  - Updated in each iteration: $x_{a_t} \leftarrow x_{a_t} + w_t$.
- Insertion-Only Stream: when w_t can only be positive.
- Insertion-Deletion Stream : when w_t can be either positive or negative.

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

Frequency Moments

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

$$F_p = |f_1|^p + |f_2|^p + \ldots + |f_n|^p$$

- Goal: Given a set S of m elements from $[n]$ and an accuracy parameter ε, output a $(1+\varepsilon)$-approximation to F_p
- Motivation: Entropy estimation, linear regression

Adversarially Robust Streaming

- Input: Updates to an underlying vector x, which arrive sequentially and *adversarially*
- Output: Evaluation (or approximation) of a given function
- Goal: Use space *sublinear* in the size m of the input S

- Adversarial robustness: "Future queries may depend on previous queries"
- Motivation: Database queries, adversarial ML

Adversarially Robust Streaming

- Input: Updates to an underlying vector x, which arrive sequentially and *adversarially*
- Output: Evaluation (or approximation) of a given function
- Goal: Use space *sublinear* in the size m of the input S

Attacker

Algorithm

Adversarially Robust Streaming

- Input: Updates to an underlying vector x, which arrive sequentially and *adversarially*
- Output: Evaluation (or approximation) of a given function
- Goal: Use space *sublinear* in the size m of the input S

Attacker

$x_1 \leftarrow x_1 + 1$

1

Algorithm

Adversarially Robust Streaming

- Input: Updates to an underlying vector x, which arrive sequentially and *adversarially*
- Output: Evaluation (or approximation) of a given function
- Goal: Use space *sublinear* in the size m of the input S

$$x_1 \leftarrow x_1 + 1$$
$$x_2 \leftarrow x_2 + 1$$

2

Attacker

Algorithm

Adversarially Robust Streaming

- Input: Updates to an underlying vector x, which arrive sequentially and *adversarially*
- Output: Evaluation (or approximation) of a given function
- Goal: Use space *sublinear* in the size m of the input S

Attacker

$$x_1 \leftarrow x_1 + 1$$
$$x_2 \leftarrow x_2 + 1$$
$$x_3 \leftarrow x_3 + 1$$

3

Algorithm

Adversarially Robust Streaming

- Input: Updates to an underlying vector x, which arrive sequentially and *adversarially*
- Output: Evaluation (or approximation) of a given function
- Goal: Use space *sublinear* in the size m of the input S

Attacker

$$x_1 \leftarrow x_1 + 1$$
$$x_2 \leftarrow x_2 + 1$$
$$x_3 \leftarrow x_3 + 1$$
$$x_1 \leftarrow x_1 - 1$$

4

Algorithm

Adversarially Robust Streaming

- Input: Updates to an underlying vector x, which arrive sequentially and *adversarially*
- Output: Evaluation (or approximation) of a given function
- Goal: Use space *sublinear* in the size m of the input S

Attacker

$$x_1 \leftarrow x_1 + 1$$
$$x_2 \leftarrow x_2 + 1$$
$$x_3 \leftarrow x_3 + 1$$
$$x_1 \leftarrow x_1 - 1$$

4

Algorithm

Insertion-Only Streams

- Space $\tilde{O}\left(\frac{1}{\varepsilon^2}\log n\right)$ algorithm for $(1+\varepsilon)$-approximation to F_2 moment [AlonMatiasSzegedy99, BlasiokDingNelson17]

- Space $\tilde{O}\left(\frac{1}{\varepsilon^2}\log n\right)$ robust algorithm for $(1+\varepsilon)$-approximation to F_2 moment [WoodruffZhou21]

Insertion-Deletion Streams

- Space $O\left(\frac{1}{\varepsilon^2}\log^2 n\right)$ algorithm for $(1+\varepsilon)$-approximation to F_2 moment [AlonMatiasSzegedy99]

- Space for robust streaming algorithm for approximation to F_2 moment?

Linear Sketch

- Algorithm maintains Ax for a matrix A throughout the stream
- All insertion-deletion streaming algorithms on a sufficiently long stream might as well be linear sketches [LiNguyenWoodruff14, AiHuLiWoodruff16]
- Any multiplicative approximation algorithm for F_p estimation based on a linear sketch requires $\Omega(n)$ space [HardtWoodruff13, GribelyukLinWoodruffYuZhou25]
- Conjecture: $\Omega(n)$ space is necessary for any adversarially robust insertion deletion algorithm

Our Results, Adversarial Robustness on Insertion-Deletion Streams

- We refute the $\Omega(n)$ conjecture!! We use a non-linear sketch!!
- Space $\mathrm{poly}\left(\frac{1}{\varepsilon}, \log n\right)$ algorithm for $(1+\varepsilon)$-approximation to F_2 moment
- Space $\mathrm{poly}\left(\frac{1}{\varepsilon}, \log n\right)$ algorithm for L_2 heavy-hitters
- Space $n^{1/C}$ overhead for algorithm for $2^{O(C)}$-approximation to functions with approximate triangle inequality; trade-off between space blow-up and approximation factor

Bounded Computation Paths

- Suppose you have a randomized algorithm that, given an adaptively chosen stream of length m, outputs a b-bit number on each update, which changes at most T times

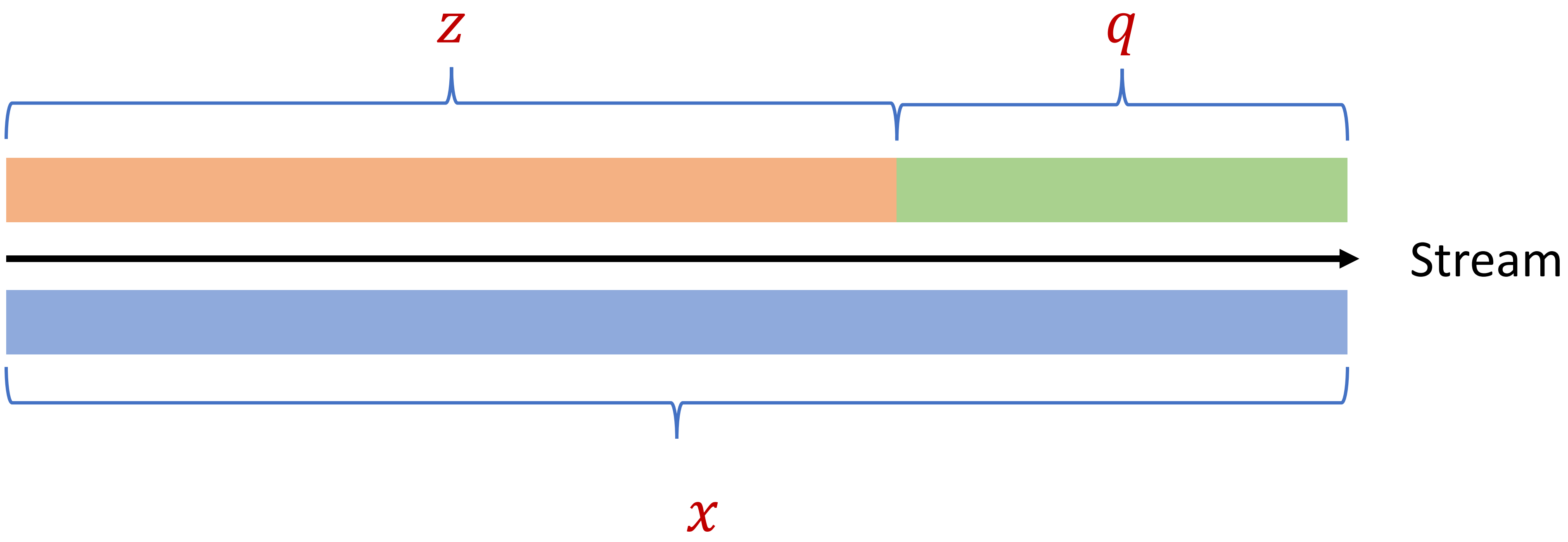
- Intuition: T positions in stream where adversary learns something

- Set failure probability of algorithm to be $\leq \frac{1}{\binom{m}{T} 2^{bT}}$

- Only an $O\big(T(\log m + b)\big)$ space overhead

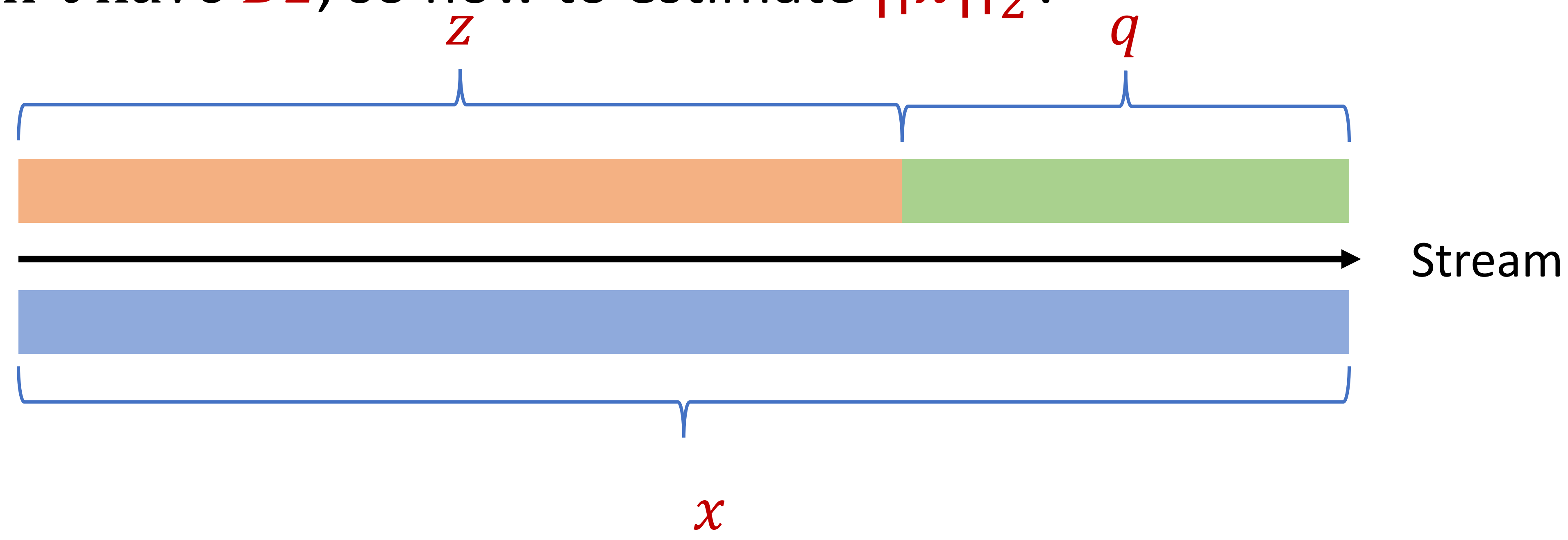
Intuition

- Decompose frequency vector x as $x = z + q$
- Use sketch matrix Az for z and sketch matrix Bq for q (independent L_2 sketches)



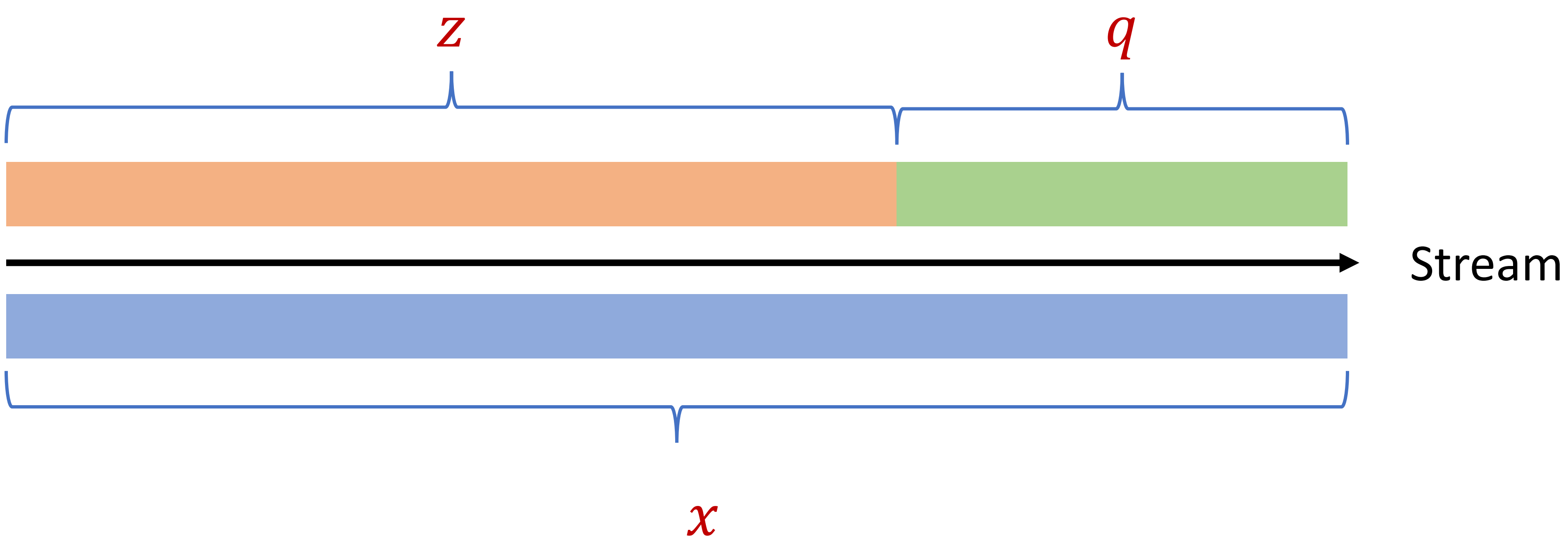
Intuition

- Since $x = z + q$, then $Bx = Bz + Bq$
- Initialized and maintained Bq at the beginning of the block
- Don't have Bz, so how to estimate $||x||_2^2$?



Intuition

- Since $x = z + q$, then $\|x\|_2^2 = \|z + q\|_2^2 = \|z\|_2^2 + 2\langle z, q\rangle + \|q\|_2^2$



Intuition

- Since $x = z + q$, then $\|x\|_2^2 = \|z + q\|_2^2 = \|z\|_2^2 + 2\langle z, q\rangle + \|q\|_2^2$

- Then EITHER:

  - $\langle z, q\rangle$ is SMALL, so that $\|x\|_2^2 \approx \|z\|_2^2 + \|q\|_2^2 \approx |Az|_2^2 + |Bq|_2^2$

  - $\langle z, q\rangle$ is LARGE, so that q is "well-aligned" with z, and we

    LEARN INFORMATION about z from q

Estimator, Corrector, Learner

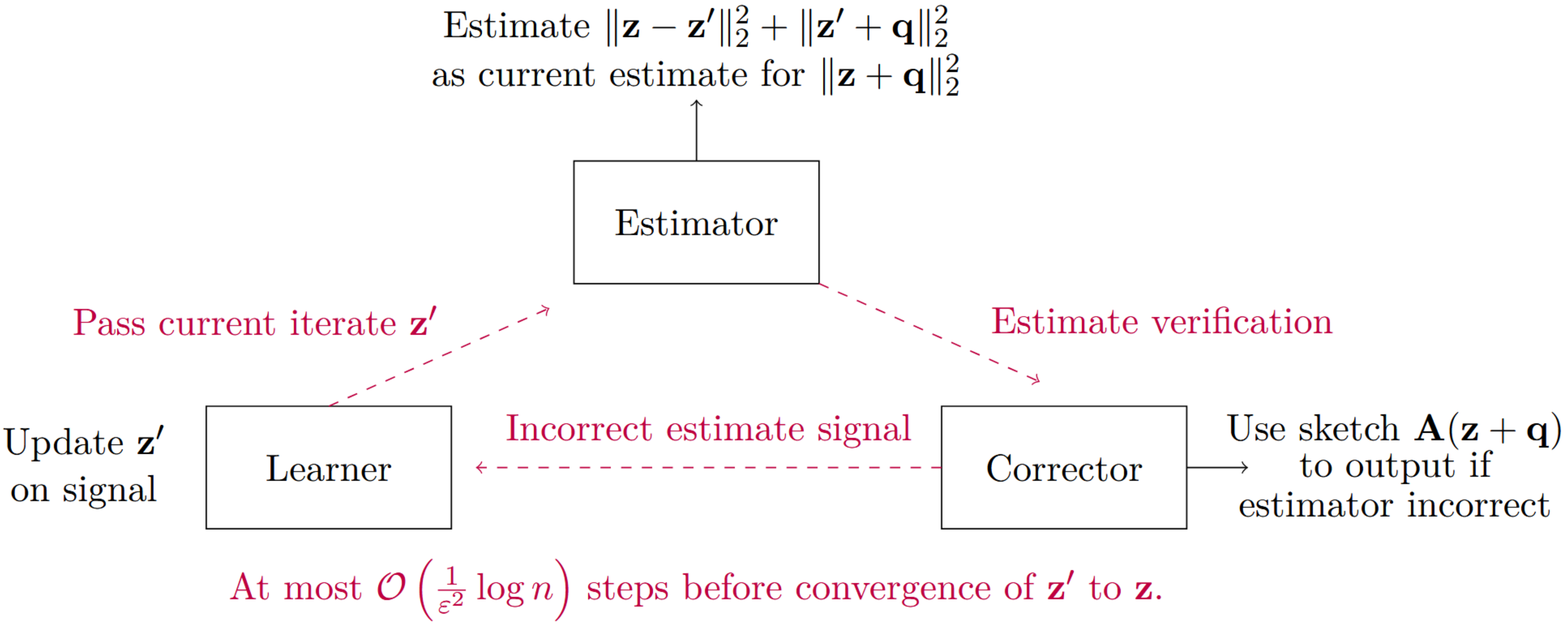
- Three ingredients:
  - An estimator, whose role is to output estimates of $\|z + q\|_2^2$
  - A learner, whose role is to learn z. When the estimator is incorrect, the learner will gain information about z from q
  - A corrector, whose role is to inform the learner/estimator when the estimate is incorrect

Estimator, Corrector, Learner

- Corrector uses Ax to detect when an estimate is incorrect

- Learner maintains a vector z', which is a linear combination of the queries q on which the algorithm was incorrect
  - Actually maintain Az' and Bz': all updates to z' are performed implicitly in the sketch space

- Estimator outputs $\|z - z'\|_2^2 + \|z' + q\|_2^2$ as estimate for $\|z + q\|_2^2$, where the first term is estimated using $Az - Az'$, and the second term is estimated using $Bz' + Bq$



The estimator outputs a current estimate, the corrector verifies its accuracy, and the learner updates its internal state z' based on incorrect estimates

Estimator, Corrector, Learner

- Sanity check #1: For $z' = 0$, then $\|z - z'\|_2^2 + \|z' + q\|_2^2$ corresponds to earlier estimator $\|z\|_2^2 + \|q\|_2^2$
- Sanity check #2: For $z' = z$, then $\|z - z'\|_2^2 + \|z' + q\|_2^2$ corresponds to correct value $\|z + q\|_2^2$
- How many interactions with corrector before $z' \approx z$?

Update Rule

- Estimator is $\|z - z'\|_2^2 + \|z' + q\|_2^2$
- Incorrect estimate implies

$$\left| \|z - z'\|_2^2 + \|z' + q\|_2^2 - \|z + q\|_2^2 \right| \geq \varepsilon \cdot \|z + q\|_2^2$$

and ultimately,

$$|\langle z - z', z' + q \rangle| \geq \frac{\varepsilon}{2} \cdot \|z + q\|_2^2$$

- Assume WLOG $\langle z - z', z' + q \rangle \geq \frac{\varepsilon}{2} \cdot \|z + q\|_2^2$ (other case similar)

Update Rule

- On query q where estimator errs, learner uses update rule $z' \leftarrow z' + \alpha(q + z')$ and truncates small coordinates
- Let z'' be the new iterate after updating (we will choose α)
- Will show: $\|z - z''\|_2^2 \leq \left(1 - \varepsilon^2\right) \cdot \|z - z'\|_2^2$
- Coordinates bounded by $\mathrm{poly}(n)$ → at most $O\left(\frac{1}{\varepsilon^2} \log n\right)$ updates

Progress

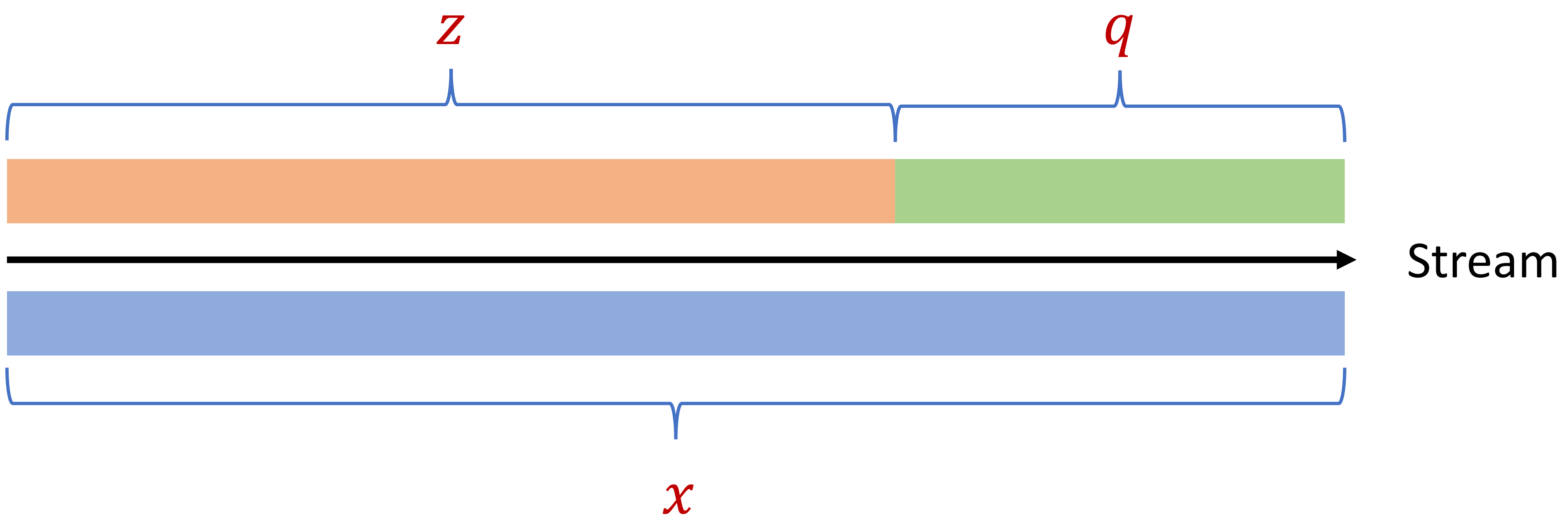
$$\|z - z''\|_2^2 = \|z - z' - \alpha(q + z')\|_2^2$$
$$= \|z - z'\|_2^2 + \alpha^2 \cdot \|q + z'\|_2^2 - 2\alpha \cdot \langle z - z', q + z' \rangle$$

- Recall: due to incorrect estimator, $\langle z - z', z' + q \rangle \geq \frac{\varepsilon}{2} \cdot \|z + q\|_2^2$
- By setting $\alpha \approx \varepsilon \cdot \frac{\|z - z'\|_2}{\|q + z'\|_2}$ (can estimate this), we have progress:

$$\|z - z''\|_2^2 \leq \left(1 - \varepsilon^2\right) \cdot \|z - z'\|_2^2$$

Progress

- Key: Can learn z using $O\left(\frac{1}{\varepsilon^2}\log n\right)$ adaptive queries to corrector
- Still need A to be robust over z and B to be robust over q
- Recurse on each block!



Recursion

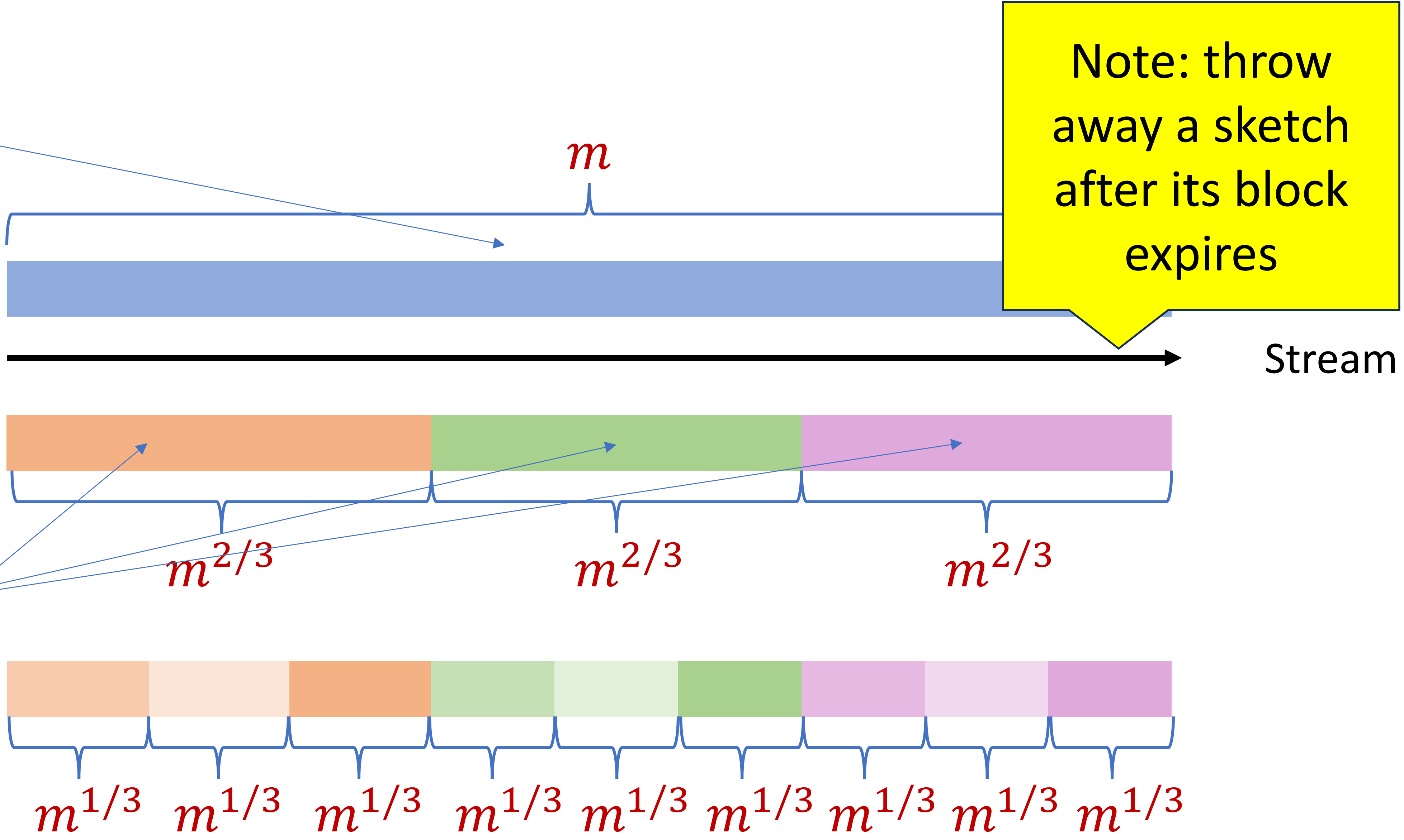
- Let $q = q_1 + q_2$, where q_2 is still evolving over the stream
- Estimator is $\|z - z'\|_2^2 + \|z' + q\|_2^2$ and suppose iterate z' is fixed
- Then:
  - First term is fixed
  - Second term is $\|z' + q_1 + q_2\|_2^2$ and estimator must be robust

Recursion

- Robust estimate for $\|z' + q_1 + q_2\|_2^2$
- Recurse using estimator-corrector-learner framework:
  - Corrector for $\|z' + q_1 + q_2\|_2^2$
  - Learner has an iterate u to learn $z' + q_1$
  - Estimator uses $\|z' + q_1 - u\|_2^2 + \|u + q_2\|_2^2$ as estimate

One F_2-sketch at top level correct on $m^{1/3}\varepsilon^{-2}\log m$ adaptive queries

Independent F_2-sketch for each block of middle level, correct on $m^{1/3}\varepsilon^{-2}\log m$ adaptive queries



Independent F_2-sketch for each block of $m^{1/3}$ adaptive queries at bottom level

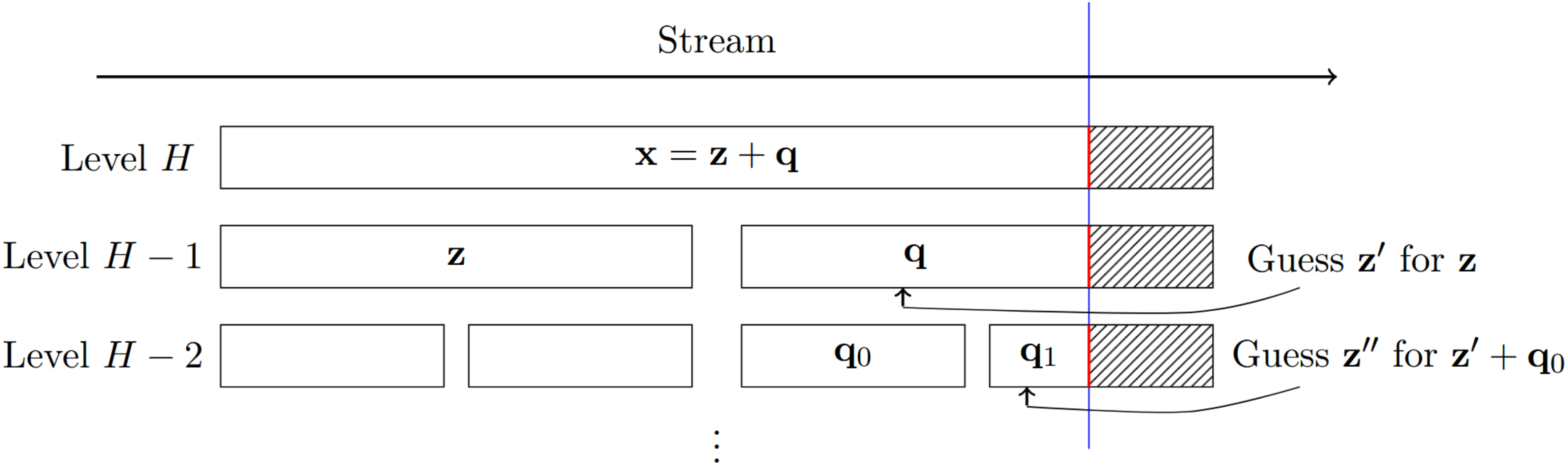


Fig. 3: Example of recursive tree structure on stream with $B = 2$ blocks and H levels. Level $H-1$ outputs $\|\mathbf{z} - \mathbf{z}'\|_2^2 + \|\mathbf{z}' + \mathbf{q}\|_2^2$ as estimate for $\|\mathbf{x}\|_2^2 = \|\mathbf{z} + \mathbf{q}\|_2^2$. Level $H-2$ outputs $\|\mathbf{z}' + \mathbf{q}_0 - \mathbf{z}''\|_2^2 + \|\mathbf{z}'' + \mathbf{q}_1\|_2^2$ as estimate for $\|\mathbf{z}' + \mathbf{q}\|_2^2 = \|\mathbf{z}' + \mathbf{q}_0 + \mathbf{q}_1\|_2^2$.

Recursion Tree

- Suppose each node in recursion tree with height H has B blocks
- Must have $B \geqslant \frac{1}{\varepsilon^2} \log n$, since adversary can force you to spawn this many children
- For $B^H \geq m$, suffices to set $H = O(\log m)$

Space Complexity

- Each corrector must be robust to $B = O\left(\frac{1}{\varepsilon^2}\log n\right)$ adaptive queries
- Each AMS algorithm uses space $O\left(\frac{1}{\varepsilon^2}\log^2 n\right)$ and only need a single active sketch at each of the $O(\log m)$ levels!
- Rescale ε to $\frac{\varepsilon}{\log m}$ as multiplicative error compounds across levels
- Total space $O\left(\frac{1}{\varepsilon^4}\log^8 n\right)$ for $\log m = O(\log n)$

Heavy-Hitters

- Let $\frac{\varepsilon}{2} \cdot e_i$ be a small scaling of the elementary vector
- If x_i is a heavy-hitter, i.e., $|x_i|^2 \geq \varepsilon \cdot \|x\|_2^2$ is a heavy-hitter, then increasing x by $\frac{\varepsilon}{2} \cdot \|x\|_2 \cdot e_i$, will increase $\|x\|_2^2$ by a "LOT"
- If x_i is NOT a heavy-hitter, i.e., $|x_i|^2 < \frac{\varepsilon}{2} \cdot \|x\|_2^2$ is a heavy-hitter, then increasing x by $\frac{\varepsilon}{2} \cdot \|x\|_2 \cdot e_i$, will increase $\|x\|_2^2$ by a "LITTLE"

Heavy-Hitters

- These two cases differ by a constant factor, and thus distinguishable by our robust F_2 estimation algorithm
- Test over all coordinates $i \in [n]$
- Space $\text{poly}\left(\frac{1}{\varepsilon}, \log n\right)$ algorithm for L_2 heavy-hitters

Triangle Inequality

- Corrector flags when $F(z - z') + F(z' + q) \geq 4 \cdot F(z + q)$
  - $F(z - z') + F(z' + q) \leq 2F(z - z') + F(z + q)$
  - $F(z + q) \leq \frac{2}{4} F(z - z') + \frac{1}{4} F(z + q)$
  - $F(z + q) \leq \frac{2}{3} F(z - z')$
- Set new z' to be $-q$
- Generalizes to approximate triangle inequality

Triangle Inequality

- Constant-factor compounding across each level, so set height of recursion tree to be a parameter C
- Each block has size $n^{1/C}$
- Algorithm must be robust to $n^{1/C}$ adaptive queries
- $n^{O(1/C)}$ space and $2^{O(C)}$-approximation

Future Directions?

- Improve the concrete $\mathrm{poly}\left(\frac{1}{\varepsilon}, \log n\right)$ factors

- Get $(1+\varepsilon)$-approximation for other functions, e.g., distinct elements, L_p estimation, etc., with $\mathrm{poly}\left(\frac{1}{\varepsilon}, \log n\right)$ overhead

- Robust algorithms on insertion-deletion streams for geometry (e.g., facility location, clustering), linear algebra (e.g., regression), graphs (e.g., cut sparsification)