Lecture 2

TTIC 41000: Algorithms for Massive Data

Toyota Technological Institute at Chicago

Spring 2021

Instructor: Sepideh Mahabadi

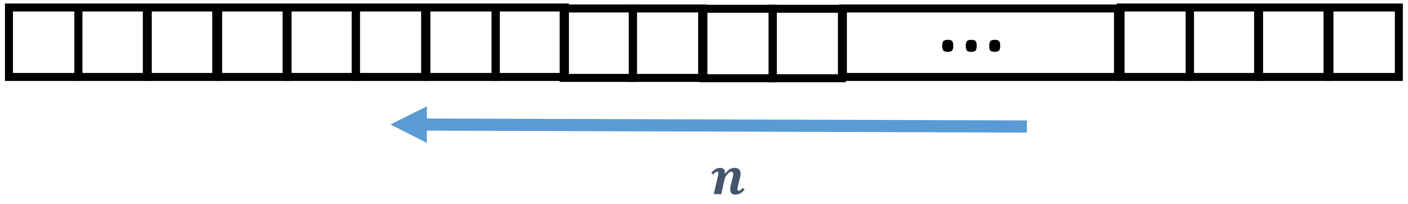
Recap from Lecture 1

Streaming Model

- Huge data set (does not fit into the main memory)
- Only sequential access to the data
  - One pass
  - Few passes (the data is stored somewhere else)
- Use little memory
  - Sublinear in input parameters
  - Sublinear in the input size
- Solve the problem (approximately)

Parameters of Interest:

1. Memory usage
2. Number of passes
3. Approximation Factor
4. (Sometimes) query/update time



Streaming Model of Computation

- ❑ Insertion-only Stream

1 2 3 4 5 6 7 8 9 10

[0,0,0,0,0,0,0,0,0,0]

Streaming Model of Computation

- ❑ Insertion-only Stream
  - Insert(3)

1 2 3 4 5 6 7 8 9 10

[0,0,**1**,0,0,0,0,0,0,0]

Streaming Model of Computation

- ❑ Insertion-only Stream
  - Insert(3), Insert(5)

1 2 3 4 5 6 7 8 9 10

[0,0,1,0,**1**,0,0,0,0,0]

Streaming Model of Computation

- ❑ Insertion-only Stream
  - Insert(3), Insert(5), Insert(7)

1	2	3	4	5	6	7	8	9	10

[0,0,1,0,1,0,**1**,0,0,0]

Streaming Model of Computation

- Insertion-only Stream
  - Insert(3), Insert(5), Insert(7), Insert(5)

1 2 3 4 5 6 7 8 9 10

[0,0,1,0,**2**,0,1,0,0,0]

Streaming Model of Computation

❑ Insertion-only Stream

- Insert(3), Insert(5), Insert(7), Insert(5), Insert(5), Insert(10)

1 2 3 4 5 6 7 8 9 10

[0,0,1,0,3,0,1,0,0,1]

Streaming Model of Computation

- ❑ Insertion-only Stream
  - Insert(3), Insert(5), Insert(7), Insert(5), Insert(5), Insert(10)
- ❑ Insertion and Deletion (Dynamic)
  - Insert(3), Insert(5), Insert(7),

1 2 3 4 5 6 7 8 9 10

[0,0,1,0,3,0,1,0,0,1]

[0,0,1,0,1,0,1,0,0,0]

Streaming Model of Computation

- Insertion-only Stream
  - Insert(3), Insert(5), Insert(7), Insert(5), Insert(5), Insert(10)

1 2 3 4 5 6 7 8 9 10

[0,0,1,0,3,0,1,0,0,1]

- Insertion and Deletion (Dynamic)
  - Insert(3), Insert(5), Insert(7), Delete(5)

[0,0,1,0,**0**,0,1,0,0,0]

Streaming Model of Computation

- ❑ Insertion-only Stream
  - Insert(3), Insert(5), Insert(7), Insert(5), Insert(5), Insert(10)

1 2 3 4 5 6 7 8 9 10

[0,0,1,0,3,0,1,0,0,1]

- ❑ Insertion and Deletion (Dynamic)
  - Insert(3), Insert(5), Insert(7), Delete(5), Insert(5), Delete(7)
  - May assume at any point #deletions(i)<=#insertions(i)
  - E.g. can be used for numbers, edges of graphs, …

[0,0,1,0,1,0,0,0,0,0]

Streaming Model of Computation

- ❑ Insertion-only Stream
  - Insert(3), Insert(5), Insert(7), Insert(5), Insert(5), Insert(10)
- ❑ Insertion and Deletion (Dynamic)
  - Insert(3), Insert(5), Insert(7), Delete(5), Insert(5), Delete(7)
  - May assume at any point #deletions(i)<=#insertions(i)
  - E.g. can be used for numbers, edges of graphs, …
- ❑ Turnstile (for vectors, and matrices)
  - Add(i,Δ): Add value Δ to the ith coordinate

1 2 3 4 5 6 7 8 9 10

[0,0,1,0,3,0,1,0,0,1]

[0,0,1,0,1,0,0,0,0,0]

[0,0,0,0,0,0,0,0,0,0]

Streaming Model of Computation

- Insertion-only Stream
  - Insert(3), Insert(5), Insert(7), Insert(5), Insert(5), Insert(10)

1 2 3 4 5 6 7 8 9 10

[0,0,1,0,3,0,1,0,0,1]

- Insertion and Deletion (Dynamic)
  - Insert(3), Insert(5), Insert(7), Delete(5), Insert(5), Delete(7)
  - May assume at any point #deletions(i)<=#insertions(i)
  - E.g. can be used for numbers, edges of graphs, …

[0,0,1,0,1,0,0,0,0,0]

- Turnstile (for vectors, and matrices)
  - Add(i,Δ): Add value Δ to the ith coordinate
  - Add(1,10),

[**10**,0,0,0,0,0,0,0,0,0]

Streaming Model of Computation

- Insertion-only Stream
  - Insert(3), Insert(5), Insert(7), Insert(5), Insert(5), Insert(10)

1 2 3 4 5 6 7 8 9 10

[0,0,1,0,3,0,1,0,0,1]

- Insertion and Deletion (Dynamic)
  - Insert(3), Insert(5), Insert(7), Delete(5), Insert(5), Delete(7)
  - May assume at any point #deletions(i)<=#insertions(i)
  - E.g. can be used for numbers, edges of graphs, …

[0,0,1,0,1,0,0,0,0,0]

- Turnstile (for vectors, and matrices)
  - Add(i,Δ): Add value Δ to the ith coordinate
  - Add(1,10), Add(4,5),

[10,0,0,**5**,0,0,0,0,0,0]

Streaming Model of Computation

1 2 3 4 5 6 7 8 9 10

- Insertion-only Stream
  - Insert(3), Insert(5), Insert(7), Insert(5), Insert(5), Insert(10)

[0,0,1,0,3,0,1,0,0,1]

- Insertion and Deletion (Dynamic)
  - Insert(3), Insert(5), Insert(7), Delete(5), Insert(5), Delete(7)
  - May assume at any point #deletions(i)<=#insertions(i)
  - E.g. can be used for numbers, edges of graphs, …

[0,0,1,0,1,0,0,0,0,0]

- Turnstile (for vectors, and matrices)
  - Add(i,Δ): Add value Δ to the ith coordinate
  - Add(1,10), Add(4,5), Add(1,-5)

[**5**,0,0,5,0,0,0,0,0,0]

Streaming Model of Computation

- ❑ Insertion-only Stream
  - Insert(3), Insert(5), Insert(7), Insert(5), Insert(5), Insert(10)

1 2 3 4 5 6 7 8 9 10

[0,0,1,0,3,0,1,0,0,1]

- ❑ Insertion and Deletion (Dynamic)
  - Insert(3), Insert(5), Insert(7), Delete(5), Insert(5), Delete(7)
  - May assume at any point #deletions(i)<=#insertions(i)
  - E.g. can be used for numbers, edges of graphs, …

[0,0,1,0,1,0,0,0,0,0]

- ❑ Turnstile (for vectors, and matrices)
  - Add(i,Δ): Add value Δ to the ith coordinate
  - Add(1,10), Add(4,5), Add(1,-5), Add(5,-2)

[5,0,0,5,**-2**,0,0,0,0,0]

Streaming Model of Computation

- Estimate #Distinct Elements (L_0 norm: #non-zero coordinates)

Basic Algorithm

- For $D \in \{(1+\epsilon)^i : 0 \le i \le \log \mathrm{n}/\epsilon\}$

Basic Algorithm

- For $D \in \{(1+\epsilon)^i : 0 \leq i \leq \log \mathrm{n}/\epsilon\}$
    - Sample each of m coordinates w.p. $\frac{1}{D}$ into set S_j
    - If all sampled coordinates are 0, return NO
    - Otherwise, return YES

Space usage:

- a single bit (for insertion only)
- A single number in [n] (to also handle deletions)

Basic Algorithm

- For $D \in \{(1+\epsilon)^i : 0 \le i \le \log \mathrm{n}/\epsilon\}$
  - For $j \in \left\{1, \dots, k = \frac{(\log 1/\delta)}{\epsilon^2}\right\}$
    - Sample each of m coordinates w.p. $\frac{1}{D}$ into set S_j
    - If all sampled coordinates are 0, return NO
    - Otherwise, return YES
  - Z=#NO
  - If $Z > k/e$ return $\mathrm{DE} < \mathrm{D}$
  - Otherwise, return $\mathrm{DE} > \mathrm{D}$

Space usage:

- k bits (for insertion only)
- k numbers in [n]
  (to also handle deletions)

Basic Algorithm

- For $D \in \{(1+\epsilon)^i : 0 \le i \le \log \mathrm{n}/\epsilon\}$
  - For $j \in \left\{1, \ldots, k = \frac{(\log 1/\delta)}{\epsilon^2}\right\}$
    - Sample each of m coordinates w.p. $\frac{1}{D}$ into set S_j
    - If all sampled coordinates are 0, return NO
    - Otherwise, return YES
  - Z=#NO
  - If $Z > k/e$ return $\mathrm{DE} < \mathrm{D}$
  - Otherwise, return $\mathrm{DE} > \mathrm{D}$
- Return smallest D for which the above reports $\mathrm{DE} < \mathrm{D}$

Space usage:

- $k \log \mathrm{n}/\epsilon$ bits (for insertion only)
- $k \log \mathrm{n}/\epsilon$ numbers in [n] (to also handle deletions)

Basic Algorithm

- For $D \in \{(1+\epsilon)^i : 0 \le i \le \log \mathrm{n}/\epsilon\}$
  - For $j \in \left\{1, \dots, k = \frac{(\log 1/\delta)}{\epsilon^2}\right\}$
    - Sample each of m coordinates w.p. $\frac{1}{D}$ into set S_j
    - If all sampled coordinates are 0, return NO
    - Otherwise, return YES
  - Z=#NO
  - If $Z > k/e$ return $\mathrm{DE} < \mathrm{D}$
  - Otherwise, return $\mathrm{DE} > \mathrm{D}$
- Return smallest D for which the above reports $\mathrm{DE} < \mathrm{D}$

Space usage:

- $k \log \mathrm{n}/\epsilon$ bits (for insertion only)
- $k \log \mathrm{n}/\epsilon$ numbers in [n] (to also handle deletions)

Assumption: access to a perfect hash function $h: [m] \to [D]$

Streaming Model of Computation

- ❑ Distinct Elements (L_0 norm)

- ❑ Morris Counter (L_1 norm in insertion-only streams)
  - Count (approximately) in space better than $O(\log n)$?

Morris Algorithm

- Let $X = 0$
- Upon receiving INCREMENT()
  - Increment X with probability $\frac{1}{2^X}$
- Upon receiving QUERY()
  - Return $\tilde{n} = 2^X$-1

Space usage:
$O(\log \log n)$

Claim 1. Let X_n denote X after n updates. Then, $\mathbb{E}[2^{X_n}] = n + 1$.

Claim 2. $\mathbb{E}[2^{2X_n}] = \frac{3}{2}n^2 + \frac{3}{2}n + 1$

$$\Pr[|\tilde{n} - n| > \epsilon n] < \frac{1}{\epsilon^2 n^2} \cdot \frac{n^2}{2} = \frac{1}{2\epsilon^2}$$

Issue

$$\Pr[|\tilde{n} - n| > \epsilon n] < \frac{1}{\epsilon^2 n^2} \cdot \frac{n^2}{2} = \frac{1}{2\epsilon^2}$$

- Not very meaningful! RHS is better than ½ only when $\epsilon > 1$ (for which we can instead always return 0 !)

- How to decrease the failure probability?

How to improve the variance

- **Morris+**

  Average of s **Morris** estimators. Variance is multiplied by $(\frac{1}{s})$.

  Setting $s = \Theta(\frac{1}{\epsilon^2 \delta})$ suffices to get failure probability δ

$$\Pr[|\tilde{n} - n| > \epsilon n] < \frac{1}{2\epsilon^2} \cdot \epsilon^2 \delta \leq \delta$$

How to improve the space

- **Morris+**

  Average of s **Morris** estimators. Variance is multiplied by $(\frac{1}{s})$.

  Setting $s = \Theta(\frac{1}{\epsilon^2 \delta})$ suffices to get failure probability δ

- **Morris++**

  Median of t **Morris+** estimators.

  Setting $s = \Theta(\frac{1}{\epsilon^2})$, each **Morris+** estimator succeeds w.p. at least $\frac{2}{3}$.

  By Chernoff and setting $t = \Theta(\log \frac{1}{\delta})$, the failure probability becomes at most δ

Improved algorithm

- **Morris+**

  Average of s **Morris** estimators. Variance is multiplied by $(\frac{1}{s})$.

  Setting $s = \Theta(\frac{1}{\epsilon^2 \delta})$ suffices to get failure probability δ

- **Morris++**

  Median of t **Morris+** estimators.

  Setting $s = \Theta(\frac{1}{\epsilon^2})$, each **Morris+** estimator succeeds w.p. at least $\frac{2}{3}$.

  By Chernoff and setting $t = \Theta(\log \frac{1}{\delta})$, the failure probability becomes at most δ

Total Space of Morris++: $\Theta(\frac{1}{\epsilon^2} \cdot \log \frac{1}{\delta} \cdot \log \log n)$ w.p. at least $1 - \delta$

This Lecture

- AMS (L_2 norm estimation)
- Count-Min (Frequency Estimation)
- Count-Sketch (Frequency Estimation)

L_2 norm Estimation

- Start with $x = \vec{\mathbf{0}} \in \mathbb{R}^m$
- Input (turnstile model): a stream of n updates (i, Δ), meaning $x_i = x_i + \Delta$
- Goal: Approximate $\|x\|_2$ at the end
- Alon-Matias-Szegedy'96 (AMS) Algorithm

Basic Algorithm

- For each of the m coordinates, independently pick a random sign $\boldsymbol{s_i} \in \{\mathbf{-1}, \mathbf{+1}\}$ with equal probability.
- Sketch: maintain $\boldsymbol{Z} = \sum_{\boldsymbol{i=1}} \boldsymbol{s_i} \cdot \boldsymbol{x_i}$ throughout the stream
- Upon receiving (i, Δ), update $\boldsymbol{Z} = \boldsymbol{Z} + (\boldsymbol{s_i} \cdot \boldsymbol{\Delta})$
- Return $\boldsymbol{Z^2}$ as an estimate for $||x||_2^2$

Basic Algorithm

- ❑ **Claim 1** (our estimator works in **expectation**): $\mathbb{E}[Z^2] = \|x\|_2^2$
- ❑ **Claim 2** (our estimator works **with good probability**)

Basic Algorithm

$$Z = \sum_{i=1}^{m} s_i x_i$$

- **Claim 1** (our estimator works in **expectation**): $\mathbb{E}[Z^2] = \|x\|_2^2$

$\mathbb{E}[Z^2] = \mathbb{E}[(\sum_i s_i x_i)^2] = \mathbb{E}\left[\sum_{i \neq j} s_i x_i s_j x_j + \sum_i s_i^2 x_i^2\right] = \sum_{i \neq j} x_i x_j \mathbb{E}[s_i s_j] + \sum_i x_i^2 \mathbb{E}[s_i^2]$

Basic Algorithm

$$Z = \sum_{i=1}^{m} s_i x_i$$

- **Claim 1** (our estimator works in **expectation**): $\mathbb{E}[Z^2] = \|x\|_2^2$

$\mathbb{E}[Z^2] = \mathbb{E}[(\sum_i s_i x_i)^2] = \mathbb{E}\left[\sum_{i \neq j} s_i x_i s_j x_j + \sum_i s_i^2 x_i^2\right] = \sum_{i \neq j} x_i x_j \mathbb{E}\left[s_i s_j\right] + \sum_i x_i^2 \mathbb{E}\left[s_i^2\right]$

- s_i and s_j are chosen independently (2-wise independence is enough)

Basic Algorithm

$$Z = \sum_{i=1}^{m} s_i x_i$$

- **Claim 1** (our estimator works in **expectation**): $\mathbb{E}[Z^2] = \|x\|_2^2$

$\mathbb{E}[Z^2] = \mathbb{E}[(\sum_i s_i x_i)^2] = \mathbb{E}\left[\sum_{i \neq j} s_i x_i s_j x_j + \sum_i s_i^2 x_i^2\right] = \sum_{i \neq j} x_i x_j \mathbb{E}[s_i s_j] +$
$\sum_i x_i^2 \mathbb{E}[s_i^2] = 0 + \sum_i x_i^2 = \|x\|_2^2$

Basic Algorithm

- **Claim 1** (our estimator works in **expectation**): $\mathbb{E}[\boldsymbol{Z}^2] = \|\boldsymbol{x}\|_2^2$

$$\mathbb{E}[Z^2] = \mathbb{E}[(\sum_i s_i x_i)^2] = \mathbb{E}\left[\sum_{i \neq j} s_i x_i s_j x_j + \sum_i s_i^2 x_i^2\right] = \sum_{i \neq j} x_i x_j \mathbb{E}[s_i s_j] + \sum_i x_i^2 \mathbb{E}[s_i^2] = 0 + \sum_i x_i^2 = \|x\|_2^2$$

- Claim 2 (our estimator works with high probability) -> Use Chebyshev
  - Need to bound the variance of the estimator

Basic Algorithm – Bounding variance

$$Z = \sum_{i=1}^{m} s_i x_i$$

- $\mathrm{V}ar(Z^2) = \mathbb{E}[Z^4] - \mathbb{E}[Z^2]^2$

Basic Algorithm – Bounding variance

$$\boldsymbol{Z} = \sum_{i=1}^{m} \boldsymbol{s_i x_i}$$

- $\mathrm{V}ar(Z^2) = \mathbb{E}[Z^4] - \mathbb{E}[Z^2]^2$
- $\left(\mathbb{E}[Z^2]\right)^2 = \left(\|x\|_2^2\right)^2 = \left(\sum_i x_i^2\right)^2 = \sum_i x_i^4 + 2 \cdot \sum_{i<j} x_i^2 x_j^2$

Basic Algorithm – Bounding variance

$$Z = \sum_{i=1}^{m} s_i x_i$$

- $Var(Z^2) = \mathbb{E}[Z^4] - \mathbb{E}[Z^2]^2$
- $\left(\mathbb{E}[Z^2]\right)^2 = \left(\|x\|_2^2\right)^2 = \left(\sum_i x_i^2\right)^2 = \sum_i x_i^4 + 2 \cdot \sum_{i<j} x_i^2 x_j^2$
- $\mathbb{E}[Z^4] = \mathbb{E}[(\sum_i s_i x_i)^4] = \mathbb{E}[\sum_i (s_i x_i)^4] + 6 \cdot \mathbb{E}\left[\sum_{i<j} \left(s_i s_j x_i x_j\right)^2\right] + 0$

e.g. $x_1 x_2 x_3 x_4 \mathbb{E}[s_1 s_2 s_3 s_4] = 0$

Basic Algorithm – Bounding variance

$$Z = \sum_{i=1}^{m} s_i x_i$$

- $Var(Z^2) = \mathbb{E}[Z^4] - \mathbb{E}[Z^2]^2$
- $(\mathbb{E}[Z^2])^2 = (\|x\|_2^2)^2 = (\sum_i x_i^2)^2 = \sum_i x_i^4 + 2 \cdot \sum_{i<j} x_i^2 x_j^2$
- $\mathbb{E}[Z^4] = \mathbb{E}[(\sum_i s_i x_i)^4] = \mathbb{E}[\sum_i (s_i x_i)^4] + 6 \cdot \mathbb{E}\left[\sum_{i<j} (s_i s_j x_i x_j)^2\right] + 0$

e.g. $x_1 x_2 x_3 x_4 \mathbb{E}[s_1 s_2 s_3 s_4] = 0$

$(1/2) x_1 x_2 x_3 x_4 \mathbb{E}[s_2 s_3 s_4 | s_1 = 1] - (1/2) x_1 x_2 x_3 x_4 \mathbb{E}[s_2 s_3 s_4 | s_1 = -1]$

Basic Algorithm – Bounding variance

$$Z = \sum_{i=1}^{m} s_i x_i$$

- $Var(Z^2) = \mathbb{E}[Z^4] - \mathbb{E}[Z^2]^2$
- $(\mathbb{E}[Z^2])^2 = (\|x\|_2^2)^2 = (\sum_i x_i^2)^2 = \sum_i x_i^4 + 2 \cdot \sum_{i<j} x_i^2 x_j^2$
- $\mathbb{E}[Z^4] = \mathbb{E}[(\sum_i s_i x_i)^4] = \mathbb{E}[\sum_i (s_i x_i)^4] + 6 \cdot \mathbb{E}\left[\sum_{i<j} (s_i s_j x_i x_j)^2\right] + 0$

e.g. $x_1 x_2^2 x_3 \mathbb{E}[s_1 s_2^2 s_3] = 0$

Basic Algorithm – Bounding variance

$$Z = \sum_{i=1}^{m} s_i x_i$$

- $Var(Z^2) = \mathbb{E}[Z^4] - \mathbb{E}[Z^2]^2$
- $\left(\mathbb{E}[Z^2]\right)^2 = \left(\|x\|_2^2\right)^2 = \left(\sum_i x_i^2\right)^2 = \sum_i x_i^4 + 2 \cdot \sum_{i<j} x_i^2 x_j^2$
- $\mathbb{E}[Z^4] = \mathbb{E}[(\sum_i s_i x_i)^4] = \mathbb{E}[\sum_i (s_i x_i)^4] + 6 \cdot \mathbb{E}\left[\sum_{i<j} \left(s_i s_j x_i x_j\right)^2\right] + 0$

e.g. $x_1 x_2^2 x_3 \mathbb{E}[s_1 s_2^2 s_3] = 0$

4-wise independence is sufficient

Basic Algorithm – Bounding variance

$$Z = \sum_{i=1}^{m} s_i x_i$$

- $Var(Z^2) = \mathbb{E}[Z^4] - \mathbb{E}[Z^2]^2$

- $\left(\mathbb{E}[Z^2]\right)^2 = \left(\|x\|_2^2\right)^2 = \left(\sum_i x_i^2\right)^2 = \sum_i x_i^4 + 2 \cdot \sum_{i<j} x_i^2 x_j^2$

- $\mathbb{E}[Z^4] = \mathbb{E}[(\sum_i s_i x_i)^4] = \mathbb{E}[\sum_i (s_i x_i)^4] + 6 \cdot \mathbb{E}\left[\sum_{i<j} \left(s_i s_j x_i x_j\right)^2\right] + 0$

$$= \mathbb{E}\left[\sum_i x_i^4\right] + \mathbb{E}\left[\sum_{i \neq j} \left(x_i x_j\right)^2\right] = \|x\|_4^4 + 6 \sum_{i<j} x_i^2 x_j^2$$

Basic Algorithm – Bounding variance

$$Z = \sum_{i=1}^{m} s_i x_i$$

- $Var(Z^2) = \mathbb{E}[Z^4] - \mathbb{E}[Z^2]^2$
- $(\mathbb{E}[Z^2])^2 = \sum_i x_i^4 + 2 \cdot \sum_{i<j} x_i^2 x_j^2$
- $\mathbb{E}[Z^4] = \mathbb{E}\|x\|_4^4 + 6\sum_{i<j} x_i^2 x_j^2$

- $Var(Z^2) = \|x\|_4^4 + 6\sum_{i<j} x_i^2 x_j^2 - \|x\|_4^4 - 2\sum_{i<j} x_i^2 x_j^2 =$

  $4\sum_{i<j} x_i^2 x_j^2 \le 2\left(\sum_i x_i^2\right)^2 = 2\|x\|_2^4$

Basic Algorithm – Bounding variance

$$Z = \sum_{i=1}^{m} s_i x_i$$

- $\text{Var}(Z^2) = \mathbb{E}[Z^4] - \mathbb{E}[Z^2]^2$
- $\left(\mathbb{E}[Z^2]\right)^2 = \sum_i x_i^4 + 2 \cdot \sum_{i<j} x_i^2 x_j^2$
- $\mathbb{E}[Z^4] = \mathbb{E}\|x\|_4^4 + 6 \sum_{i<j} x_i^2 x_j^2$

- $\text{Var}(Z^2) \leq 2\|x\|_2^4$
- $\sigma = \sqrt{Var(Z^2)} = \sqrt{2}\, \|x\|_2^2$

Basic Algorithm – Chebyshev

- $\mathbb{E}[Z^2] = \|x\|_2^2$
- $\sigma = \sqrt{Var(Z^2)} = \sqrt{2}\,\|x\|_2^2$

Basic Algorithm – Chebyshev

- $\mathbb{E}[Z^2] = \|x\|_2^2$
- $\sigma = \sqrt{Var(Z^2)} = \sqrt{2}\ \|x\|_2^2$

- Chebyshev: $\Pr\left[\left|Z^2 - \|x\|_2^2\ \right| \geq c\|x\|_2^2\right] \leq 2/c^2$
- E.g. with **constant** probability our estimator Z^2 is within **constant** factor of the true value $\|x\|_2^2$

Basic Algorithm – Chebyshev

- $\mathbb{E}[Z^2] = \|x\|_2^2$
- $\sigma = \sqrt{Var(Z^2)} = \sqrt{2}\,\|x\|_2^2$

- Chebyshev: $\Pr\left[\left|Z^2 - \|x\|_2^2\right| \geq c\|x\|_2^2\right] \leq 2/c^2$
- E.g. with **constant** probability our estimator Z^2 is within **constant** factor of the true value $\|x\|_2^2$

- We want to do better!
- Goal: get $(\mathbf{1} + \boldsymbol{\epsilon})$ approximation with constant probability
- **Repeat** Basic algorithm!

Overall AMS Algorithm

- Keep multiple estimators $\boldsymbol{Z_1, \cdots, Z_k}$
- Report $\boldsymbol{Z' = \mathrm{Avg}\left(Z_1^2, \ldots, Z_k^2\right)}$
- Does not change the expectation
- $\mathbb{E}[Z'] = \mathbb{E}\left[\frac{\sum_i Z_i^2}{k}\right] = \mathbb{E}[Z_1] = \|x\|_2^2$

Overall AMS Algorithm

- Keep multiple estimators $\boldsymbol{Z_1, \cdots, Z_k}$
- Report $\boldsymbol{Z' = \mathrm{Avg}(Z_1^2, \ldots, Z_k^2)}$
- Does not change the expectation , i.e., $\mathbb{E}[Z'] = \|x\|_2^2$
- Variance decreases by a factor of k
- $Var(Z') = \mathrm{Var}\left(\frac{\sum_i Z_i^2}{k}\right) = \frac{\sum Var(Z_i^2)}{k^2} = \frac{Var(Z_1^2)}{k} = \frac{2\|x\|_2^4}{k}$

Overall AMS Algorithm

- Keep multiple estimators $\boldsymbol{Z_1, \cdots, Z_k}$
- Report $\boldsymbol{Z' = \mathrm{Avg}(Z_1^2, \ldots, Z_k^2)}$
- Does not change the expectation , i.e., $\mathbb{E}[Z'] = \|x\|_2^2$
- Variance decreases by a factor of k, i.e., $\mathrm{V}ar(Z') = \frac{2\|x\|_2^4}{k}$
- $\sigma = \sqrt{Var(Z')} = \sqrt{2}\frac{\|x\|_2^2}{k}$

Overall AMS Algorithm

- Keep multiple estimators $\boldsymbol{Z_1, \cdots, Z_k}$
- Report $\boldsymbol{Z' = \mathrm{Avg}(Z_1^2, \ldots, Z_k^2)}$
- Does not change the expectation , i.e., $\mathbb{E}[Z'] = \|x\|_2^2$
- $\sigma = \sqrt{Var(Z')} = \sqrt{2}\,\|x\|_2^2/k$ Set $\boldsymbol{k = O(\frac{1}{\epsilon^2})}$

Overall AMS Algorithm

- Keep multiple estimators $\boldsymbol{Z_1, \cdots, Z_k}$
- Report $\boldsymbol{Z' = \mathbf{Avg}\left(Z_1^2, \ldots, Z_k^2\right)}$
- Does not change the expectation , i.e., $\mathbb{E}[Z'] = \|x\|_2^2$
- $\sigma = \sqrt{Var(Z')} = \sqrt{2}\,\|x\|_2^2/k$ Set $\boldsymbol{k = O(\frac{1}{\epsilon^2})}$

- Chebyshev $\Pr\left[\left|Z' - \|x\|_2^2\right| \geq c\epsilon\|x\|_2^2\right] \leq 1/c^2$
- get a $(1+\epsilon)$ approximation with a constant probability.

Remarks

- To get a $(1+\epsilon)$ approximation with **probability** $(\mathbf{1}-\boldsymbol{\delta})$.
  - Run $t = O(\log\frac{1}{\delta})$ instances of AMS and take the median
  - By Chernoff Bound, the median of the AMS estimators work

- Total space usage $O(\frac{\log\frac{1}{\delta}}{\epsilon^2})$ numbers.
- What about keeping the random signs s_i?
- Only need 4-wise independence of $s_1, \dots, s_m$, (in bounding $\mathbb{E}[(\sum_i s_i x_i)^4]$)
- e.g. $\mathbb{E}[s_1 s_2 s_3 s_4] = 0$
- Can generate such variables using $O(\log m)$ random bits.

Outline

- So far we learned how to maintain the norm of a vector in small space
- What else can we do in small (e.g. $\tilde{O}(k)$) space?
- We can keep track of all coordinates with additive error, i.e., for each coordinate we can report $\widetilde{x_i}$ that is within $x_i \pm \frac{\|x\|_1}{k}$
- This is specially useful if x_i is large (heavy-hitter), e.g. $|x_i| \geq \frac{\|x\|_1}{k}$
- (there are at most k such coordinates)

Outline

- So far we learned how to maintain the norm of a vector in small space
- What else can we do in small (e.g. $\tilde{O}(k)$) space?
- We can keep track of all coordinates with additive error, i.e., for each coordinate we can report $\tilde{x}_i$ that is within $x_i \pm \frac{\|x\|_1}{k}$
- This is specially useful if x_i is large (heavy-hitter), e.g. $|x_i| \geq \frac{\|x\|_1}{k}$

$$HH_\phi^p(x) = \{i: |x_i| > \phi\|x\|_p\}$$

Frequency Estimation

- Count-Min
- Count-Sketch

Goal:

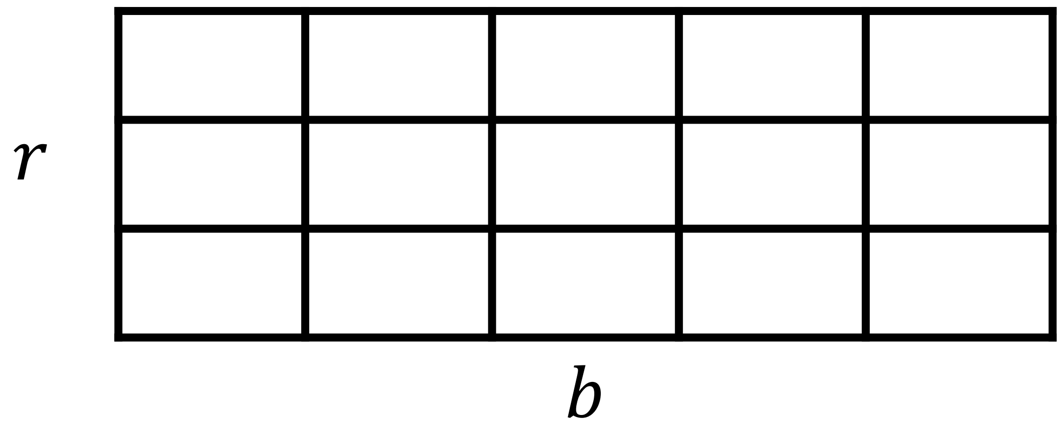
- Start with $x = \vec{\mathbf{0}} \in \mathbb{R}^m$
- Turnstile Model: input is a stream of updates (i, Δ), where $i \in [m]$
- (for now assume all coordinates remain positive at all time).

- Keep track of all coordinates with additive error, i.e.,
- for each coordinate we can report $\widetilde{\boldsymbol{x}_i}$ that is within $\boldsymbol{x}_i \pm \frac{\|\boldsymbol{x}\|_1}{\boldsymbol{k}}$

Count Min

Turnstile Model: input is a stream of updates (i, Δ), where $i \in [m]$

#rows r $= O(\log 1/\delta)$
#buckets/row b $= O(2k)$

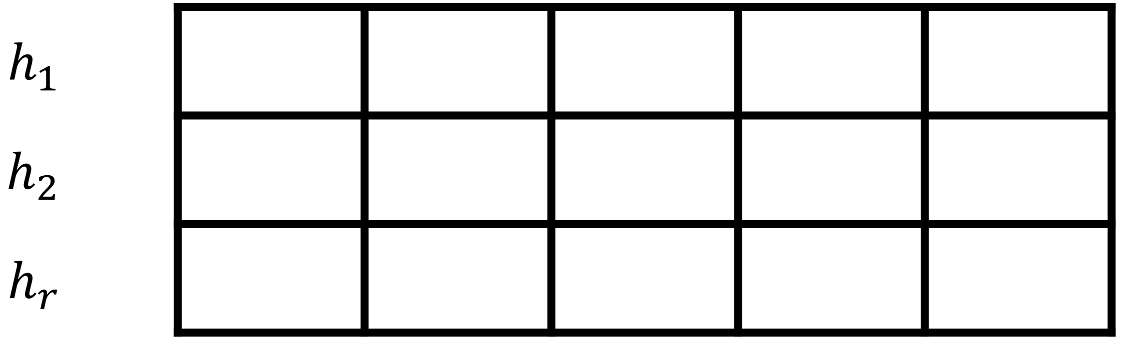


Count Min

Turnstile Model: input is a stream of updates (i, Δ), where $i \in [m]$

#rows r $= O(\log 1/\delta)$
#buckets/row b = $O(2k)$

- **Hash** $\forall j \leq r$: $h_j: [m] \rightarrow [b]$

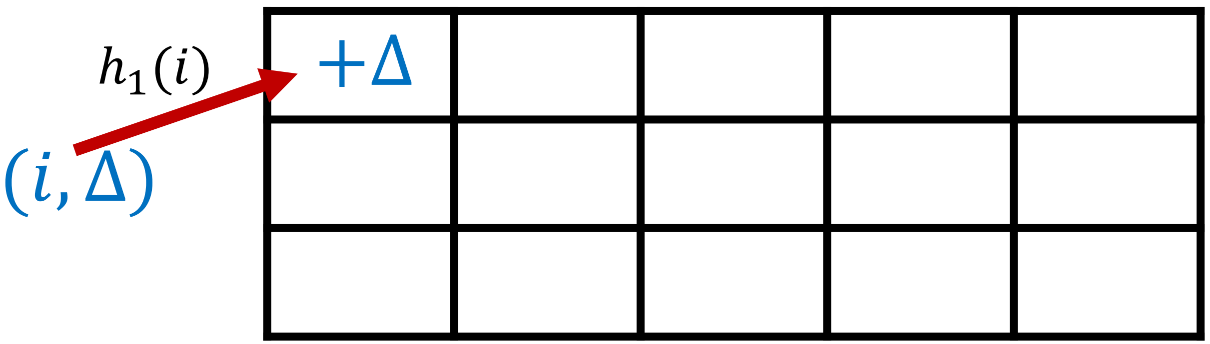


Count Min

Turnstile Model: input is a stream of updates (i, Δ), where $i \in [m]$

#rows r $= O(\log 1/\delta)$
#buckets/row b = $O(2k)$

- **Hash** $\forall j \leq r$: $h_j : [m] \to [b]$



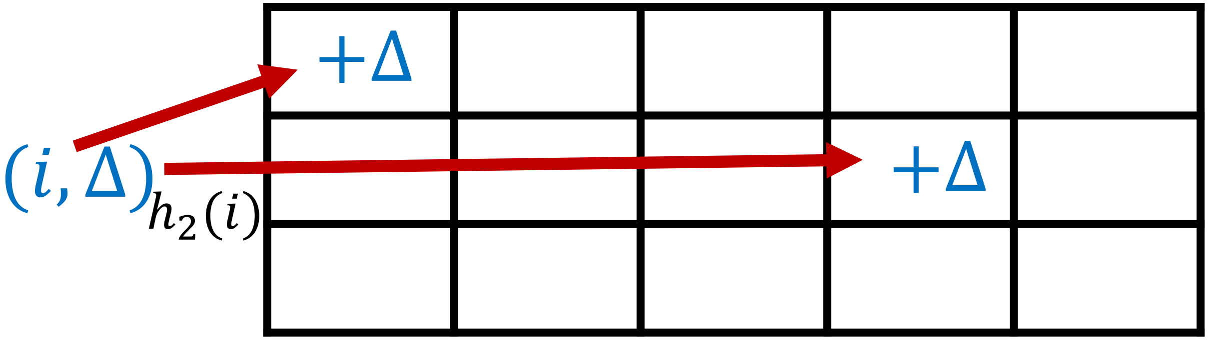
- **Update:** $C[j, h_j(i)] \mathrel{+}= \Delta$
-

Count Min

Turnstile Model: input is a stream of updates (i, Δ), where $i \in [m]$

#rows r $= O(\log 1/\delta)$
#buckets/row b = $O(2k)$

- **Hash** $\forall j \leq r$: $h_j : [m] \rightarrow [b]$

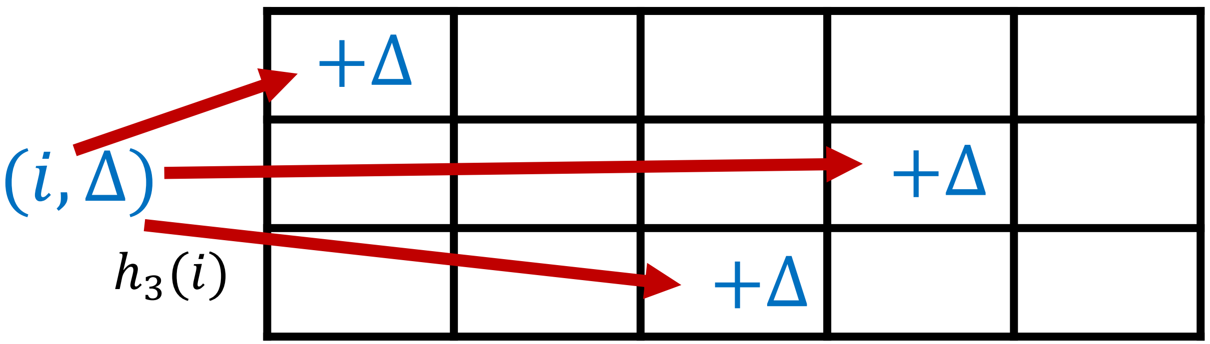


- **Update:** $C[j, h_j(i)] \mathrel{+}= \Delta$
-

Count Min

Turnstile Model: input is a stream of updates (i, Δ), where $i \in [m]$

#rows r $= O(\log 1/\delta)$
#buckets/row b = $O(2k)$



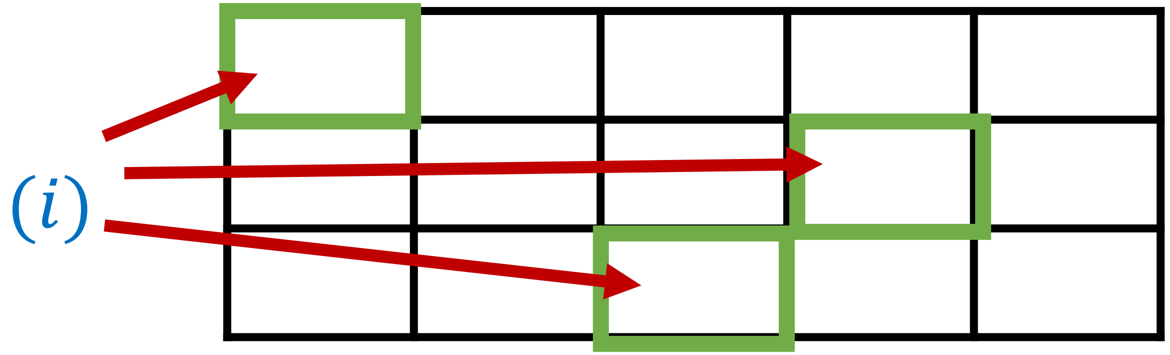
- **Hash** $\forall j \leq r$: $h_j : [m] \to [b]$

- **Update:** $C[j, h_j(i)] += \Delta$
-

Count Min

Query(i), where $i \in [m]$

Each Bucket is an over-estimation of x_i

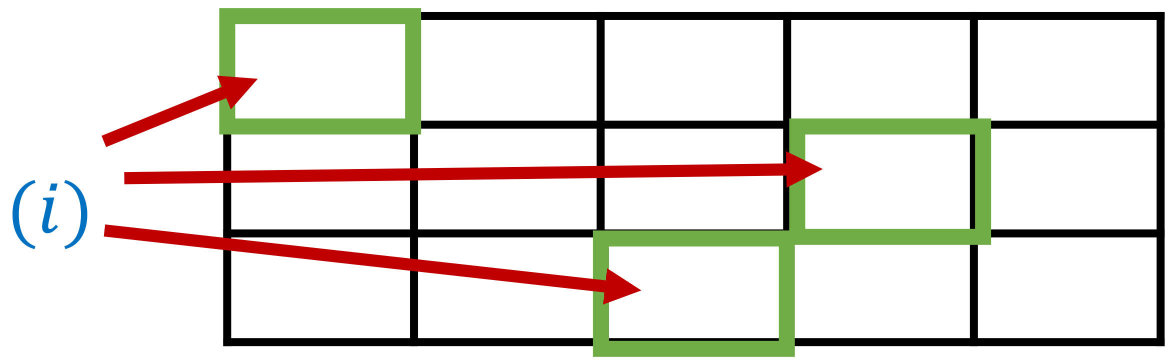


- **Update:** $C[j, h_j(i)] += \Delta$

Count Min

Query(i), where $i \in [m]$

Each Bucket is an over-estimation of x_i



- **Update:** $C[j, h_j(i)] \mathrel{+}= \Delta$
- **Estimate** $\hat{x}_i \coloneqq \min_j C[j, h_j(i)]$

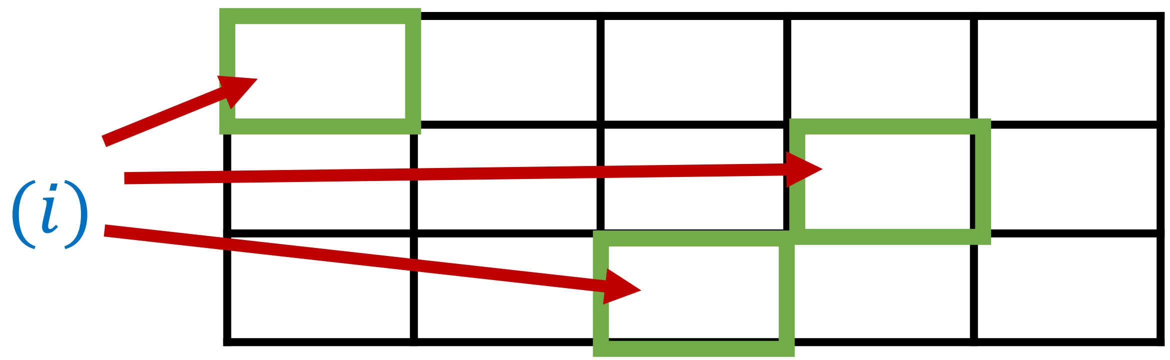
Count Min

Query(i), where $i \in [m]$

#rows r $= O(\log 1/\delta)$
#buckets/row b = $O(2k)$

Estimation guarantee: w.p $(1-\delta)$

$$|x_i - \hat{x}_i| \leq (1/k) \cdot \|\boldsymbol{x}\|_1$$



- **Update:** $C[j, h_j(i)] \mathrel{+}= \Delta$
- **Estimate** $\hat{x}_i \coloneqq \min_j C[j, h_j(i)]$

Count Min

Estimation guarantee: w.p $(1-\delta)$
$$|x_i - \hat{x}_i| \leq (1/k) \cdot \|\boldsymbol{x}\|_1$$

- Fix j, and consider h_j (which we assume is 2-wise independent)

Count Min

Estimation guarantee: w.p $(1-\delta)$
$|x_i - \hat{x}_i| \le (1/k) \cdot \|\boldsymbol{x}\|_1$

- Fix j, and consider h_j (which we assume is 2-wise independent)
- For $i' \in [m]$ Let $Z_{i'}$ be the indicator variable which is $\mathbf{1}[h_j(i') = h_j(i)]$

Count Min

Estimation guarantee: w.p $(1-\delta)$
$|x_i - \hat{x}_i| \le (1/k) \cdot \|\boldsymbol{x}\|_1$

- Fix j, and consider h_j (which we assume is 2-wise independent)
- Let $Z_{i'}$ be the indicator variable which is $\mathbf{1}[h_j(i') = h_j(i)]$
- $C[j, h_j(i)] = x_i + \sum_{i' \neq i} Z_{i'} x_{i'}$

Count Min

Estimation guarantee: w.p $(1-\delta)$

$$|x_i - \hat{x}_i| \leq (1/k) \cdot \|\boldsymbol{x}\|_1$$

- Fix j, and consider h_j (which we assume is 2-wise independent)
- Let $Z_{i'}$ be the indicator variable which is $\mathbf{1}\left[h_j(i') = h_j(i)\right]$
- $C\left[j, h_j(i)\right] = x_i + \sum_{i' \neq i} Z_{i'} x_{i'} \coloneqq x_i + \mathrm{Err}$

Count Min

Estimation guarantee: w.p $(1-\delta)$
$|x_i - \hat{x}_i| \le (1/k) \cdot \|\boldsymbol{x}\|_1$

- Fix j, and consider h_j (which we assume is 2-wise independent)
- Let $Z_{i'}$ be the indicator variable which is $\mathbf{1}\left[h_j(i') = h_j(i)\right]$
- $C\left[j, h_j(i)\right] = x_i + \sum_{i' \neq i} Z_{i'} x_{i'} \coloneqq x_i + Err$
- Thus the expected error is $\mathbb{E}[Err] = \left(\frac{1}{B}\right) \sum_{i' \neq i} x_{i'} \le \|x\|_1 / 2k$

Count Min

Estimation guarantee: w.p $(1-\delta)$
$$|x_i - \hat{x}_i| \le (1/k) \cdot \|\boldsymbol{x}\|_1$$

- Fix j, and consider h_j (which we assume is 2-wise independent)
- Let $Z_{i'}$ be the indicator variable which is $\mathbf{1}\left[h_j(i') = h_j(i)\right]$
- $C\left[j, h_j(i)\right] = x_i + \sum_{i' \neq i} Z_{i'} x_{i'} := x_i + Err$
- Thus the expected error is $\mathbb{E}[Err] = \left(\frac{1}{B}\right) \sum_{i' \neq i} x_{i'} \le \|x\|_1 / 2k$
- By Markov, $\Pr\left[Err > \frac{\|x\|_1}{k}\right] \le \frac{1}{2}$

Count Min

Estimation guarantee: w.p $(1-\delta)$
$|x_i - \hat{x}_i| \leq (1/k) \cdot \|\boldsymbol{x}\|_1$

- Fix j, and consider h_j (which we assume is 2-wise independent)
- Let $Z_{i'}$ be the indicator variable which is $\mathbf{1}[h_j(i') = h_j(i)]$
- $C[j, h_j(i)] = x_i + \sum_{i' \neq i} Z_{i'} x_{i'} \coloneqq x_i + Err$
- Thus the expected error is $\mathbb{E}[Err] = \left(\frac{1}{B}\right) \sum_{i' \neq i} x_{i'} \leq \|x\|_1 / 2k$
- By Markov, $\Pr\left[Err > \frac{\|x\|_1}{k}\right] \leq \frac{1}{2}$
- By Independence of the rows: $\Pr\left[MinErr > \frac{\|x\|_1}{k}\right] \leq \frac{1}{2^r} \leq \delta$

Outline

- We can keep track of all coordinates with additive error, i.e., for each coordinate we can report $\widetilde{x}_i$ that is within $x_i \pm \frac{\|x\|_1}{k}$
- CountMin

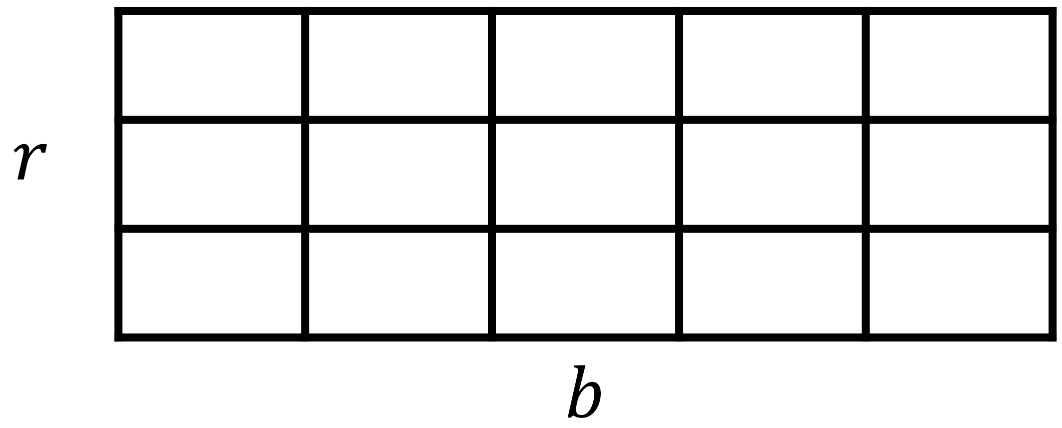
- We can keep track of all coordinates with additive error, i.e., for each coordinate we can report $\widetilde{x}_i$ that is within $x_i \pm \frac{\|\boldsymbol{x}\|_2}{\sqrt{\boldsymbol{k}}}$
- CountSketch

CountSketch

Count Sketch

Turnstile Model: input is a stream of updates (i, Δ), where $i \in [m]$

#rows r $= O(\log 1/\delta)$
#buckets/row b = $O(9k)$

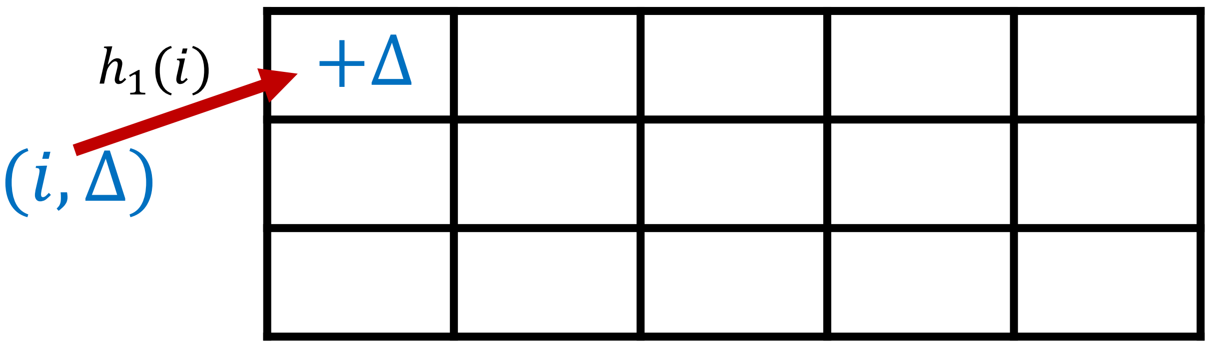


Count Sketch

Turnstile Model: input is a stream of updates (i, Δ), where $i \in [m]$

#rows r $= O(\log 1/\delta)$
#buckets/row b = $O(9k)$

- **Hash** $h_j: [m] \to [b]$
- **Sign** $\sigma_j: [m] \to \{-1, +1\}$



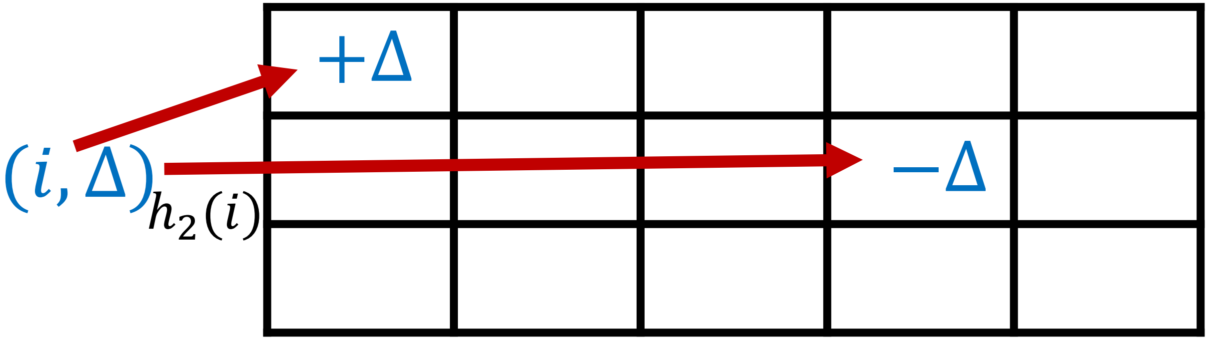
- **Update:** $C[j, h_j(i)] \mathrel{+}= \sigma_j(i) \cdot \Delta$
-

Count Sketch

Turnstile Model: input is a stream of updates (i, Δ), where $i \in [m]$

#rows r $= O(\log 1/\delta)$
#buckets/row b $= O(9k)$

- **Hash** $h_j : [m] \to [b]$
- **Sign** $\sigma_j : [m] \to \{-1, +1\}$



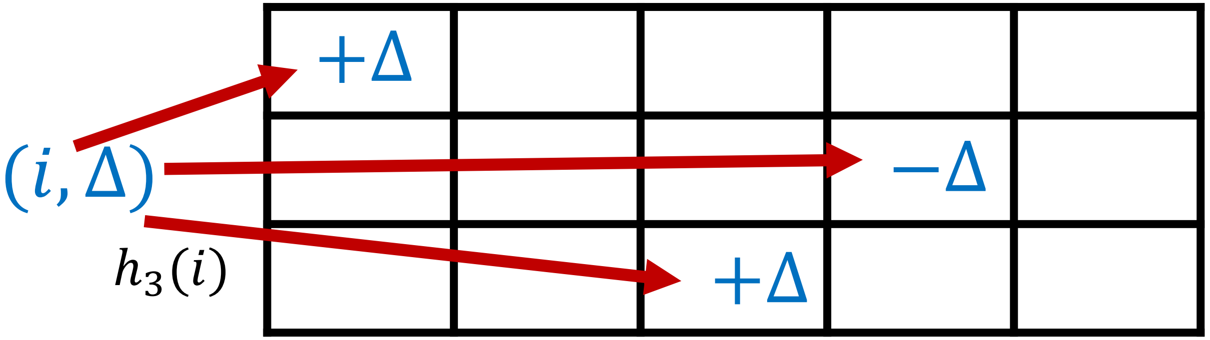
- **Update:** $C[j, h_j(i)] \mathrel{+}= \sigma_j(i) \cdot \Delta$
-

Count Sketch

Turnstile Model: input is a stream of updates (i, Δ), where $i \in [m]$

#rows r $= O(\log 1/\delta)$
#buckets/row b = $O(9k)$

- **Hash** $h_j: [m] \to [b]$
- **Sign** $\sigma_j: [m] \to \{-1, +1\}$



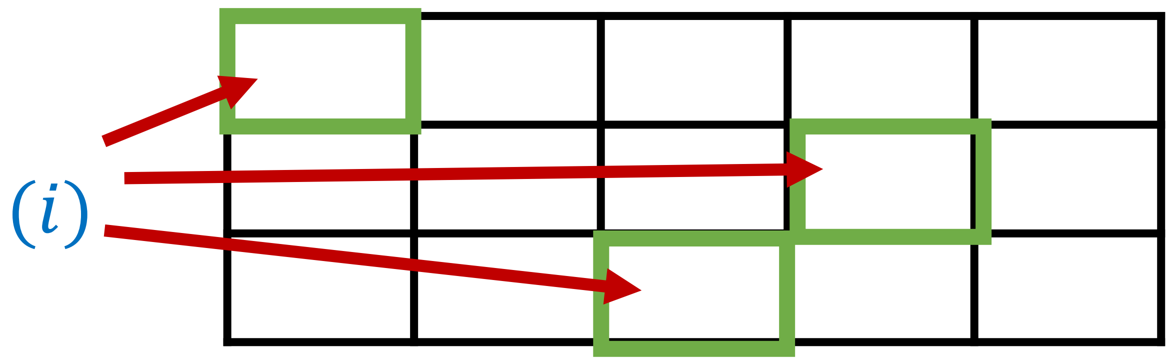
- **Update:** $C[j, h_j(i)] \mathrel{+}= \sigma_j(i) \cdot \Delta$
-

Count Sketch

Query(i), where $i \in [m]$

#rows r $= O(\log 1/\delta)$
#buckets/row b = $O(9k)$

- **Hash** $h_j : [m] \to [b]$
- **Sign** $\sigma_j : [m] \to \{-1, +1\}$



- **Update:** $C[j, h_j(i)] \mathrel{+}= \sigma_j(i) \cdot \Delta$
- **Estimate** $\hat{x}_i = \mathbf{median_j}\ \sigma_j(i) C[j, h_j(i)]$

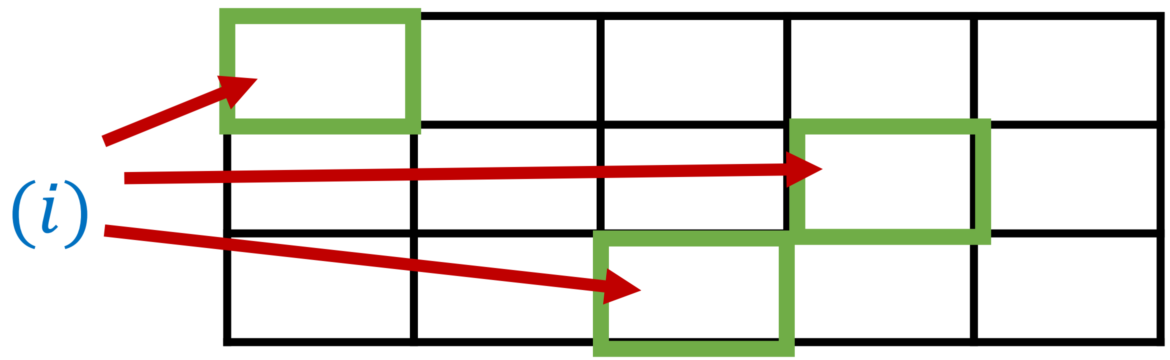
Count Sketch

Query(i), where $i \in [m]$

#rows r $= O(\log 1/\delta)$
#buckets/row b = $O(9k)$

Estimation guarantee: w.p $(1 - \delta)$

$$|x_i - \hat{x}_i| \leq (1/\sqrt{k}) \cdot \|\boldsymbol{x}\|_2$$



- **Update:** $C[j, h_j(i)] \mathrel{+}= \sigma_j(i) \cdot \Delta$
- **Estimate** $\hat{x}_i = \mathbf{median_j}\ \sigma_j(i) C[j, h_j(i)]$

Count Sketch

Estimation guarantee: w.p $(1-\delta)$
$$|x_i - \hat{x}_i| \leq (1/\sqrt{k}) \cdot \|\boldsymbol{x}\|_2$$

- Fix j, and consider h_j (which we assume is 2-wise independent)
- Let $Z_{i'}$ be the indicator variable which is $\mathbf{1}\left[h_j(i') = h_j(i)\right]$

Count Sketch

Estimation guarantee: w.p $(1-\delta)$

$$|x_i - \hat{x}_i| \leq (1/\sqrt{k}) \cdot \|\boldsymbol{x}\|_2$$

- Fix j, and consider h_j (which we assume is 2-wise independent)
- Let $Z_{i'}$ be the indicator variable which is $\mathbf{1}\left[h_j(i') = h_j(i)\right]$
- $C\left[j, h_j(i)\right] = x_i + \sum_{i' \neq i} Z_{i'} \sigma_j(i') x_{i'} := x_i + Err$

Count Sketch

Estimation guarantee: w.p $(1-\delta)$

$$|x_i - \hat{x}_i| \leq (1/\sqrt{k}) \cdot \|\boldsymbol{x}\|_2$$

- Fix j, and consider h_j (which we assume is 2-wise independent)
- Let $Z_{i'}$ be the indicator variable which is $\mathbf{1}\left[h_j(i') = h_j(i)\right]$
- $C\left[j, h_j(i)\right] = x_i + \sum_{i' \neq i} Z_{i'} \sigma_j(i') x_{i'} := x_i + Err$
- Goal: the expected error is $\mathbb{E}[|Err|] \leq \|x\|_2/(3\sqrt{k})$

Count Sketch

Estimation guarantee: w.p $(1-\delta)$

$$|x_i - \hat{x}_i| \leq (1/\sqrt{k}) \cdot \|\boldsymbol{x}\|_2$$

- Fix j, and consider h_j (which we assume is 2-wise independent)
- Let $Z_{i'}$ be the indicator variable which is $\mathbf{1}[h_j(i') = h_j(i)]$
- $C[j, h_j(i)] = x_i + \sum_{i' \neq i} Z_{i'} \sigma_j(i') x_{i'} := x_i + Err$
- Goal: the expected error is $\mathbb{E}[|Err|] \leq \|x\|_2 / (3\sqrt{k})$
- By Markov, $\Pr\left[|Err| > \frac{\|x\|_2}{\sqrt{k}}\right] \leq \frac{1}{3}$

Count Sketch

Estimation guarantee: w.p $(1-\delta)$

$$|x_i - \hat{x}_i| \le (1/\sqrt{k}) \cdot \|\boldsymbol{x}\|_2$$

- Fix j, and consider h_j (which we assume is 2-wise independent)
- Let $Z_{i'}$ be the indicator variable which is $\mathbf{1}\left[h_j(i') = h_j(i)\right]$
- $C\left[j, h_j(i)\right] = x_i + \sum_{i' \neq i} Z_{i'} \sigma_j(i') x_{i'} := x_i + Err$
- Goal: the expected error is $\mathbb{E}[|Err|] \le \|x\|_2/(3\sqrt{k})$
- By Markov, $\Pr\left[|Err| > \frac{\|x\|_2}{\sqrt{k}}\right] \le \frac{1}{3}$
- By Chernoff: $\Pr\left[MedianErr > \frac{\|x\|_2}{\sqrt{k}}\right] \le e^{-\frac{c \log\frac{1}{\delta}}{3}} \le \delta$

Count Sketch

Estimation guarantee: w.p $(1-\delta)$

$$|x_i - \hat{x}_i| \leq (1/\sqrt{k}) \cdot \|\boldsymbol{x}\|_2$$

- Fix j, and consider h_j (which we assume is 2-wise independent)
- Let $Z_{i'}$ be the indicator variable which is $\mathbf{1}\left[h_j(i') = h_j(i)\right]$
- $C\left[j, h_j(i)\right] = x_i + \sum_{i' \neq i} Z_{i'} \sigma_j(i') x_{i'} := x_i + Err$
- Goal: the expected error is $\mathbb{E}[|Err|] \leq \|x\|_2 / (3\sqrt{k})$

Count Sketch

Estimation guarantee: w.p $(1-\delta)$

$$|x_i - \hat{x}_i| \le (1/\sqrt{k}) \cdot \|\boldsymbol{x}\|_2$$

- Fix j, and consider h_j (which we assume is 2-wise independent)
- Let $Z_{i'}$ be the indicator variable which is $\mathbf{1}\left[h_j(i') = h_j(i)\right]$
- $C\left[j, h_j(i)\right] = x_i + \sum_{i' \neq i} Z_{i'} \sigma_j(i') x_{i'} := x_i + Err$
- Goal: the expected error is $\mathbb{E}[|Err|] \le \|x\|_2/(3\sqrt{k})$
- By Jensen's inequality $\mathbb{E}[|Err|] \le \sqrt{\mathbb{E}[|Err|^2]}$

Jensen's inequality

Jensen's inequality:

ϕ is convex

$$\phi(\mathbb{E}[x]) \leq \mathbb{E}[\phi(x)]$$

In our application:

$\phi(\mathrm{x}) := \mathrm{x}^2$

$$(\mathbb{E}[|Err|])^2 \leq \mathbb{E}[|Err|^2]$$

$$\mathbb{E}[|Err|] \leq \sqrt{\mathbb{E}[|Err|^2]}$$

Count Sketch

Estimation guarantee: w.p $(1-\delta)$

$|x_i - \hat{x}_i| \le (1/\sqrt{k}) \cdot \|\boldsymbol{x}\|_2$

- Fix j, and consider h_j (which we assume is 2-wise independent)
- Let $Z_{i'}$ be the indicator variable which is $\mathbf{1}[h_j(i') = h_j(i)]$
- $C[j, h_j(i)] = x_i + \sum_{i' \neq i} Z_{i'} \sigma_j(i') x_{i'} := x_i + Err$
- Goal: the expected error is $\mathbb{E}[|Err|] \le \|x\|_2/(3\sqrt{k})$
- By Jensen's inequality $\mathbb{E}[|Err|] \le \sqrt{\mathbb{E}[|Err|^2]}$
- $\le \left(\mathbb{E}\left[\sum_{i' \neq i} Z_{i'} x_{i'}^2 + \sum_{\substack{i_1, i_2 \neq i \\ i_1 \neq i_2}} Z_{i_1} Z_{i_2} \sigma_j(i_1) \sigma_j(i_2) x_{i'}^2\right]\right)^{1/2} =$

Count Sketch

Estimation guarantee: w.p $(1-\delta)$

$$|x_i - \hat{x}_i| \leq (1/\sqrt{k}) \cdot \|\boldsymbol{x}\|_2$$

- Fix j, and consider h_j (which we assume is 2-wise independent)
- Let $Z_{i'}$ be the indicator variable which is $\mathbf{1}[h_j(i') = h_j(i)]$
- $C[j, h_j(i)] = x_i + \sum_{i' \neq i} Z_{i'} \sigma_j(i') x_{i'} := x_i + Err$
- Goal: the expected error is $\mathbb{E}[|Err|] \leq \|x\|_2/(3\sqrt{k})$
- By Jensen's inequality $\mathbb{E}[|Err|] \leq \sqrt{\mathbb{E}[|Err|^2]}$
- $\leq \left(\sum_{i' \neq i} x_{i'}^2 \mathbb{E}[Z_{i'}] + \sum_{\substack{i_1, i_2 \neq i \\ i_1 \neq i_2}} x_{i'}^2 \mathbb{E}[Z_{i_1} Z_{i_2} \sigma_j(i_1) \sigma_j(i_2)] \right)^{1/2} =$

Count Sketch

Estimation guarantee: w.p $(1-\delta)$
$|x_i - \hat{x}_i| \leq (1/\sqrt{k}) \cdot \|\boldsymbol{x}\|_2$

- Fix j, and consider h_j (which we assume is 2-wise independent)
- Let $Z_{i'}$ be the indicator variable which is $\mathbf{1}\left[h_j(i') = h_j(i)\right]$
- $C\left[j, h_j(i)\right] = x_i + \sum_{i' \neq i} Z_{i'} \sigma_j(i') x_{i'} := x_i + Err$
- Goal: the expected error is $\mathbb{E}[|Err|] \leq \|x\|_2/(3\sqrt{k})$
- By Jensen's inequality $\mathbb{E}[|Err|] \leq \sqrt{\mathbb{E}[|Err|^2]}$
- $\leq \left(\sum_{i' \neq i} x_{i'}^2 \mathbb{E}[Z_{i'}] + \sum_{\substack{i_1, i_2 \neq i \\ i_1 \neq i_2}} x_{i'}^2 \mathbb{E}\left[Z_{i_1} Z_{i_2} \sigma_j(i_1) \sigma_j(i_2)\right]\right)^{1/2} =$
- $\left(\sum_{i' \neq i} \mathbb{E}(Z_{i'}) x_{i'}^2\right)^{1/2} \leq \frac{\|x\|_2}{\sqrt{B}} \leq \frac{\|x\|_2}{3\sqrt{k}}$

Outline

- We can keep track of all coordinates with additive error, i.e., for each coordinate we can report $\widetilde{x_i}$ that is within $x_i \pm \frac{\|x\|_1}{k}$
- CountMin

- We can keep track of all coordinates with additive error, i.e., for each coordinate we can report $\widetilde{x_i}$ that is within $x_i \pm \frac{\|\boldsymbol{x}\|_2}{\sqrt{\boldsymbol{k}}}$
- CountSketch

Next Lecture

- L_0 sampler
- More combinatorial Algorithms