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

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```
Streaming Model of Computation

- Insertion-only Stream
  - Insert(3)

```
[0,0,0,0,0,0,0,0,0,0]
```
Streaming Model of Computation

- Insertion-only Stream
  - Insert(3), Insert(5)
Streaming Model of Computation

- Insertion-only Stream
  - Insert(3), Insert(5), Insert(7)
Streaming Model of Computation

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

\[
[0,0,1,0,2,0,1,0,0,0]\]
Streaming Model of Computation

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

\[ [0, 0, 1, 0, 3, 0, 1, 0, 0, 1] \]
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)
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)

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

```
1 2 3 4 5 6 7 8 9 10
[0,0,1,0,3,0,1,0,0,1]
[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)

- **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, ...

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

1 2 3 4 5 6 7 8 9 10
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
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)\leq\#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
  - Add(1, 10),
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)\leq\#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
- Add(1,10), Add(4,5),

[0,0,1,0,3,0,1,0,0,1]
[0,0,1,0,1,0,0,0,0,0]
[10,0,0,5,0,0,0,0,0,0]
Streaming Model of Computation

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

- Insertion and Deletion (Dynamic)
  - Insert(3), Insert(5), Insert(7), Delete(5), Insert(5), Delete(7)
  - May assume at any point \( \#\text{deletions}(i) \leq \#\text{insertions}(i) \)
  - 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
  - Add(1,10), Add(4,5), Add(1,-5)
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 $\#\text{deletions}(i)\leq\#\text{insertions}(i)$
  - E.g. can be used for numbers, edges of graphs, ...

- **Turnstile (for vectors, and matrices)**
  - Add(i, $\Delta$): Add value $\Delta$ to the $i$th coordinate
  - Add(1,10), Add(4,5), Add(1,-5), Add(5,-2)
Streaming Model of Computation

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

• For $D \in \{(1 + \epsilon)^i : 0 \leq i \leq \log n/\epsilon\}$
Basic Algorithm

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

- 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 \( D \in \{(1 + \epsilon)^i : 0 \leq i \leq \log n/\epsilon\} \)

• For \( j \in \{1, \ldots, k = \frac{(\log 1/\delta)}{\epsilon^2}\} \)
  - 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

\( Z = \#\text{NO} \)

• If \( Z > k/e \) return DE < D
• Otherwise, return DE > D

Space usage:
- k bits (for insertion only)
- k numbers in \([n]\) (to also handle deletions)
Basic Algorithm

- For $D \in \{(1 + \epsilon)^i : 0 \leq i \leq \log n/\epsilon\}$
  - For $j \in \{1, \ldots, k = \frac{(\log 1/\delta)}{\epsilon^2}\}$
    - 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 DE < D
  - Otherwise, return DE > D

- Return smallest $D$ for which the above reports DE < D

Space usage:
- $k \log n/\epsilon$ bits (for insertion only)
- $k \log n/\epsilon$ numbers in $[n]$ (to also handle deletions)
Basic Algorithm

- For $D \in \{(1 + \epsilon)^i : 0 \leq i \leq \log n/\epsilon\}$
  - For $j \in \{1, \ldots, k = (\log 1/\delta)/\epsilon^2\}$
    - 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 $DE < D$
  - Otherwise, return $DE > D$

- Return smallest $D$ for which the above reports $DE < D$

**Space usage:**
- $k\log n/\epsilon$ bits (for insertion only)
- $k\log n/\epsilon$ numbers in $[n]$ (to also handle deletions)

**Assumption:** access to a perfect hash function $h: [m] \to [D]$
Streaming Model of Computation

- Distinct Elements  \((L_0\ \text{norm})\)

- Morris Counter  \((L_1\ \text{norm in insertion-only streams})\)
  - 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 \( \frac{1}{2} \) 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 $\left(\frac{1}{s}\right)$.
  Setting $s = \Theta\left(\frac{1}{\epsilon^2\delta}\right)$ suffices to get failure probability $\delta$

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

• **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 $\delta$
Improved algorithm

• **Morris+**
  
  Average of $s$ Morris estimators. Variance is multiplied by $\left(\frac{1}{s}\right)$.
  
  Setting $s = \Theta\left(\frac{1}{\epsilon^2 \delta}\right)$ suffices to get failure probability $\delta$

• **Morris++**
  
  Median of $t$ Morris+ estimators.
  
  Setting $s = \Theta\left(\frac{1}{\epsilon^2}\right)$, each Morris+ estimator succeeds w.p. at least $\frac{2}{3}$.
  
  By Chernoff and setting $t = \Theta\left(\log\frac{1}{\delta}\right)$, the failure probability becomes at most $\delta$

**Total Space of Morris++:** $\Theta\left(\frac{1}{\epsilon^2} \cdot \log\frac{1}{\delta} \cdot \log\log n\right)$ 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 $x = \vec{0} \in \mathbb{R}^m$

- Input (turnstile model): a stream of $n$ updates $(i, \Delta)$, 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 $s_i \in \{-1, +1\}$ with equal probability.

- Sketch: maintain $Z = \sum_{i=1} s_i \cdot x_i$ throughout the stream.

- Upon receiving $(i, \Delta)$, update $Z = Z + (s_i \cdot \Delta)$

- Return $Z^2$ 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

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

\[
\begin{align*}
\mathbb{E}[Z^2] &= \mathbb{E}[(\sum_i s_i x_i)^2] = \mathbb{E}[\sum_{i \neq j} s_i x_i s_j x_j + \sum_i s_i^2 x_i^2] = \sum_{i \neq j} x_i x_j \mathbb{E}[s_i s_j] + \\
&+ \sum_i x_i^2 \mathbb{E}[s_i^2] = Z = \sum_{i=1}^{m} s_i x_i
\end{align*}
\]
Basic Algorithm

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

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

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

- **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): $\mathbb{E}[Z^2] = ||x||_2^2$

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

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

\[ \text{Var}(Z^2) = \mathbb{E}[Z^4] - [\mathbb{E}[Z^2]]^2 \]
Basic Algorithm – Bounding variance

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

\[ (\mathbb{E}[Z^2])^2 = (\|x\|_2^2)^2 = (\sum x_i^2)^2 = \sum x_i^4 + 2 \cdot \sum_{i<j} x_i^2 x_j^2 \]
Basic Algorithm – Bounding variance

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

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

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

e.g. \( x_1 x_2 x_3 x_4 \mathbb{E}[s_1 s_2 s_3 s_4] = 0 \)
Basic Algorithm – Bounding variance

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

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

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

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

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

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

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

e.g. \( x_1 x_2^2 x_3 \mathbb{E}[s_1 s_2^2 s_3] = 0 \)
Basic Algorithm – Bounding variance

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

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

\( \mathbb{E}[Z^4] = \mathbb{E}[\sum_i s_ix_i] = \mathbb{E}[\sum_i (s_ix_i)^4] + 6 \cdot \mathbb{E} \left[ \sum_{i<j}(s_is_jx_ix_j)^2 \right] + 0 \)

e.g. \( x_1x_2^2x_3 \mathbb{E}[s_1s_2^2s_3] = 0 \)

4-wise independence is sufficient
Basic Algorithm – Bounding variance

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

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

\( \mathbb{E}[Z^4] = \mathbb{E}\left[ (\sum_i s_i x_i)^4 \right] = \mathbb{E}\left[ (\sum_i (s_i x_i))^4 \right] + 6 \cdot \mathbb{E}\left[ \sum_{i<j} (s_i s_j x_i x_j)^2 \right] + 0 \\
= \mathbb{E}\left[ \sum_i x_i^4 \right] + \mathbb{E}\left[ \sum_{i \neq j} (x_i x_j)^2 \right] = \|x\|_4^4 + 6 \sum_{i<j} x_i^2 x_j^2 \)

\( Z = \sum_{i=1}^m s_i x_i \)
Basic Algorithm – Bounding variance

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

\( (\mathbb{E}[Z^2])^2 = \sum_i x_i^4 + 2 \cdot \sum_{i<j} x_i^2 x_j^2 \)

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

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

\[ 4 \sum_{i<j} x_i^2 x_j^2 \leq 2(\sum_i x_i^2)^2 = 2\|x\|_2^4 \]
Basic Algorithm – Bounding variance

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

\[ (\mathbb{E}[Z^2])^2 = \sum_i x_i^4 + 2 \cdot \sum_{i<j} x_i^2 x_j^2 \]

\[ \mathbb{E}[Z^4] = \mathbb{E}\|x\|^4 + 6 \sum_{i<j} x_i^2 x_j^2 \]

\[ \text{Var}(Z^2) \leq 2\|x\|^4 \]

\[ \sigma = \sqrt{\text{Var}(Z^2)} = \sqrt{2} \|x\|^2 \]
Basic Algorithm – Chebyshev

\[ E[Z^2] = \|x\|_2^2 \]
\[ \sigma = \sqrt{Var(Z^2)} = \sqrt{2} \|x\|_2^2 \]
Basic Algorithm – Chebyshev

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

- Chebyshev: \( \Pr[|Z^2 - \|x\|_2^2| \geq c\|x\|_2^2] \leq 2/c^2 \)
- E.g. with constant probability our estimator \( Z^2 \) is within constant factor of the true value \( \|x\|_2^2 \)
Basic Algorithm – Chebyshev

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

- Chebyshev: \( \Pr[|Z^2 - \|x\|_2^2| \geq c\|x\|_2^2] \leq \frac{2}{c^2} \)
- 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 \((1 + \epsilon)\) approximation with constant probability
- Repeat Basic algorithm!
Overall AMS Algorithm

- Keep multiple estimators $Z_1, \ldots, Z_k$
- Report $Z' = \text{Avg}(Z_1^2, \ldots, Z_k^2)$
- 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 $Z_1, \ldots, Z_k$
- Report $Z' = \text{Avg}(Z_1^2, \ldots, Z_k^2)$
- Does not change the expectation, i.e., $\mathbb{E}[Z'] = \|x\|_2^2$
- Variance decreases by a factor of $k$

$$Var(Z') = Var\left(\frac{\sum_i Z_i^2}{k}\right) = \frac{\sum Var(Z_i^2)}{k^2} = \frac{Var(Z_1^2)}{k} = \frac{2\|x\|_2^4}{k}$$
Overall AMS Algorithm

- Keep multiple estimators $Z_1, \ldots, Z_k$
- Report $Z' = \text{Avg}(Z_1^2, \ldots, Z_k^2)$
- Does not change the expectation, i.e., $\mathbb{E}[Z'] = \|x\|^2$
- Variance decreases by a factor of $k$, i.e., $\text{Var}(Z') = \frac{2\|x\|_4}{k}$
- $\sigma = \sqrt{\text{Var}(Z')} = \sqrt{2} \frac{\|x\|_2^3}{k}$
Overall AMS Algorithm

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

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

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

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

- Total space usage $O\left(\frac{\log^1}{\epsilon^2 \delta}\right)$ numbers.

- What about keeping the random signs $s_i$?

- Only need 4-wise independence of $s_1, \ldots, s_m$, (in bounding $\mathbb{E}[\left(\sum_i s_i x_i\right)^4]$)
  - 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 $\tilde{x}_i$ that is within $x_i \pm \frac{\|x\|_1}{k}$
• This is specially useful if $x_i$ is large (heavy-hitter), e.g. $|x_i| \geq \frac{\|x\|_1}{k}$
• (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. $|x_i| \geq \frac{\|x\|_1}{k}$

$$HH^p_\phi(x) = \{i: |x_i| > \phi \|x\|_p\}$$
Frequency Estimation

• Count-Min
• Count-Sketch
Goal:

- Start with $x = \vec{0} \in \mathbb{R}^m$
- Turnstile Model: input is a stream of updates $(i, \Delta)$, 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 $\tilde{x}_i$ that is within $x_i \pm \frac{\|x\|_1}{k}$
Count Min

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

\#rows \quad r = O(\log 1/\delta)

\#buckets/row \quad b = O(2k)
Count Min

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

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

\[ \begin{array}{cccc}
  h_1 & & & \\
  h_2 & & & \\
  h_r & & & \\
\end{array} \]

- Hash \(\forall j \leq r: h_j: [m] \rightarrow [b]\)
Count Min

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

<table>
<thead>
<tr>
<th>#rows</th>
<th>(r = O(\log 1/\delta))</th>
</tr>
</thead>
<tbody>
<tr>
<td>#buckets/row</td>
<td>(b = O(2k))</td>
</tr>
</tbody>
</table>

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

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

\[h_1(i)\]
Count Min

Turnstile Model: input is a stream of updates \((i, \Delta)\), 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:** \( C[j, h_j(i)] += \Delta \)
Count Min

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

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

\begin{align*}
\text{Hash } & \forall j \leq r: h_j: [m] \to [b] \\
\text{Update: } & C[j, h_j(i)] += \Delta
\end{align*}
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: \(C[j, h_j(i)] += \Delta\)
- Estimate \(\hat{x}_i := \min_j C[j, h_j(i)]\)
Count Min

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

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

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

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

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

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

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

$$|x_i - \hat{x}_i| \leq \frac{1}{k} \cdot ||x||_1$$
Count Min

• 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 $1[h_j(i') = h_j(i)]$

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

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

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

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

$|x_i - \hat{x}_i| \leq \left(\frac{1}{k}\right) \cdot \|x\|_1$
Count Min

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

• Let $Z_{i'}$ be the indicator variable which is $1[h_{j}(i') = h_j(i)]$

• $C[j, h_j(i)] = x_i + \sum_{i' \neq i} Z_{i'} x_{i'} := x_i + \text{Err}$

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

$|x_i - \hat{x}_i| \leq \frac{1}{k} \cdot \|x\|_1$
Count Min

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

Estimation guarantee: \( \text{w.p } (1 - \delta) \)
\[ |x_i - \hat{x}_i| \leq \left(\frac{1}{k}\right) \cdot \|x\|_1 \]
Count Min

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

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

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

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

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

• CountSketch
CountSketch
Count Sketch

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

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

**Hash** \(h_j: [m] \rightarrow [b]\)

**Sign** \(\sigma_j: [m] \rightarrow \{-1, +1\}\)

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

\(\text{Turnstile Model: input is a stream of updates } (i, \Delta), \text{ where } i \in [m]\)
Count Sketch

Turnstile Model: input is a stream of updates \((i, \Delta)\), 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:** \(C[j, h_j(i)] += \sigma_j(i) \cdot \Delta\)
Count Sketch

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

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

- **Hash** \(h_j : [m] \rightarrow [b]\)
- **Sign** \(\sigma_j : [m] \rightarrow \{-1, +1\}\)

- **Update:** \(C[j, h_j(i)] += \sigma_j(i) \cdot \Delta\)
Count Sketch

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

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

**Rows** $r = O(\log 1/\delta)$

**Buckets/Row** $b = O(9k)$

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

**Estimate** $\hat{x}_i = \text{median}_j \sigma_j(i) C[j, h_j(i)]$
Count Sketch

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

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

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

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

- **Update:** $C[j, h_j(i)] += \sigma_j(i) \cdot \Delta$
- **Estimate** $\hat{x}_i = \text{median}_j \sigma_j(i) C[j, h_j(i)]$
Count Sketch

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

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

$$|x_i - \hat{x}_i| \leq \frac{1}{\sqrt{k}} \cdot \|x\|_2$$
Count Sketch

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

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

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

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

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

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

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

• Let \( Z_{i'} \) be the indicator variable which is \( 1[ h_j(i') = h_j(i) ] \)

• \( C[j, h_j(i)] = x_i + \sum_{i' \neq i} Z_{i'} \sigma_j(i') x_{i'} \): \( := x_i + \text{Err} \)

• **Goal:** the expected error is \( \mathbb{E}[|\text{Err}|] \leq \|x\|_2/(3\sqrt{k}) \)

• By Markov, \( \Pr \left[ |\text{Err}| > \frac{\|x\|_2}{\sqrt{k}} \right] \leq \frac{1}{3} \)

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

\[ |x_i - \hat{x}_i| \leq \left( \frac{1}{\sqrt{k}} \right) \cdot \|x\|_2 \]
Count Sketch

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

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

$$|x_i - \hat{x}_i| \leq \left(\frac{1}{\sqrt{k}}\right) \cdot \|x\|_2$$
Count Sketch

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

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

Estimation guarantee: w.p \( (1 - \delta) \)
\[ |x_i - \hat{x}_i| \leq (1/\sqrt{k}) \cdot ||x||_2 \]
## Jensen’s inequality

### Jensen’s inequality:

- $\phi$ is convex

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

### In our application:

- $\phi(x) := x^2$

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

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

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

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

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

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

**Estimation guarantee:** w.p $1 - \delta$

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

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

Estimation guarantee: w.p $(1 - \delta)$
\[|x_i - \hat{x}_i| \leq (1/\sqrt{k}) \cdot \|x\|_2\]
We can keep track of all coordinates with additive error, i.e., for each coordinate we can report $\tilde{x}_i$ that is within $x_i \pm \frac{\|x\|_1}{k}$.

- **CountMin**

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

- **CountSketch**
Next Lecture

• $L_0$ sampler
• More combinatorial Algorithms