Sublinear Algorithms

LECTURE 8

Last time

- Streaming
- Distinct Elements
- k-wise independent hash functions

Today

- Approximate counting
- Estimation of the 2nd moment
- Linear sketching

Sign up for project meetings, scribing, grading on Piazza

Sofya Raskhodnikova;Boston University

Frequency Moments Estimation

Input: a stream $\langle a_1, a_2, \ldots, a_m \rangle \in [n]^m$

- The frequency vector of the stream is $f = (f_1, \ldots, f_n)$, where f_i is the number of times i appears in the stream
- The p-th frequency moment is $F_p = \left\|f\right\|_p^p = \sum_{i=1}^n f_i^p$

F_0 is the number of nonzero entries of f (# of distinct elements)

$F_1 = m$ (# of elements in the stream)

$F_2 = \left\|f\right\|_2^2$ is a measure of non-uniformity

used e.g. for anomaly detection in network analysis

$F_\infty = \max_i f_i$ is the most frequent element

Goal: Estimate F_p up to a multiplicative factor $(1 \pm \varepsilon)$ with probability $\geq 2/3$

Approximate Counting: Estimating F_1

Input: a stream $\langle a_1, a_2, \ldots, a_m \rangle \in [n]^m$

Warm-up: Compute m. How much space do you need?

Goal: Estimate m up to a multiplicative factor $(1 \pm \varepsilon)$ with probability $\geq \frac{2}{3}$

Today: $O(\varepsilon^{-2} \log \log m)$ space algorithm [Morris 78]

Morris Algorithm (initial version)

1. Initialize $X \leftarrow 0$
2. For each element, increment X by 1 w. p. 2^{-X}
3. Return $\widetilde{m} = 2^X - 1$

- Intuitively, X is keeping track of $\log(m+1)$
- Intuitively, expected increment to 2^X at each step is $2^X \cdot 2^{-X} = 1$.

Based on Jelani Nelson's lecture notes: https://www.sketchingbigdata.org/fall20/lec/notes.pdf

Morris Algorithm: Analysis

Morris Algorithm (initial version)

1. Initialize $X \leftarrow 0$
2. For each element, increment X by 1 w. p. 2^{-X}
3. Return $\tilde{m} = 2^X - 1$

- Let X_i represent X after i elements.
- $2^{X_0} = 1$
- $\mathbb{E}[2^{X_i}] = \mathbb{E}\left[\mathbb{E}[\, 2^{X_i} \mid X_{i-1} \,]\right]$

By the compact form of the Law of Total Expectation

$$= \mathbb{E}[2^{X_{i-1}+1} \cdot 2^{-X_{i-1}} + 2^{X_{i-1}} \cdot (1 - 2^{-X_{i-1}})]$$

$$= \mathbb{E}[2 + 2^{X_{i-1}} - 1] = \mathbb{E}[2^{X_{i-1}}] + 1 = i + 1$$

$\mathbb{E}[2^X] = m + 1$

Claim. $\mathrm{Var}[2^X] \le m^2/2$

Variance Calculation

Claim. $\text{Var}[2^X] \le m^2/2$

Proof: $\text{Var}[2^{X_i}] = \mathbb{E}[(2^{X_i})^2] - \mathbb{E}[2^{X_i}]^2$ — by definition of variance

$= \mathbb{E}[2^{2X_i}] - (i+1)^2$ — by our calculation of expectation

$\mathbb{E}[2^{2X_i}] = \mathbb{E}[\mathbb{E}[2^{2X_i} \mid X_{i-1}]]$ — by compact form of Law of Total Expectation

$= \mathbb{E}[2^{2(X_{i-1}+1)} \cdot 2^{-X_{i-1}} + 2^{2X_{i-1}}(1-2^{-X_{i-1}})]$

$= \mathbb{E}[2^{X_{i-1}+2} + 2^{2X_{i-1}} - 2^{X_{i-1}}]$

$= \mathbb{E}[3 \cdot 2^{X_{i-1}} + 2^{2X_{i-1}}] = 3i + \mathbb{E}[2^{2X_{i-1}}]$

$= 1 + 3(1+2+\ldots+i) = 1 + 3\frac{i(i+1)}{2}$ — by induction

$\text{Var}[2^{X_i}] = \mathbb{E}[2^{2X_i}] - (i+1)^2$

$= 1 + \frac{3}{2}i^2 + \frac{3}{2}i - i^2 - 2i - 1$

$= \frac{i^2}{2} - \frac{i}{2} \le \frac{i^2}{2}$

Recall that $X = X_m$.

Morris Algorithm: Analysis

Morris Algorithm (initial version)

1. Initialize $X \leftarrow 0$
2. For each element, increment X by 1 w. p. 2^{-X}
3. Return $\tilde{m} = 2^X - 1$.

- Let X_i represent X after i elements.
- $2^{X_0} = 1$
- $\mathbb{E}[2^{X_i}] = \mathbb{E}\big[\mathbb{E}[\,2^{X_i} \mid X_{i-1}\,]\big]$

By the compact form of the Law of Total Expectation

$$= \mathbb{E}[2^{X_i} \cdot 2^{-X_{i-1}} + 2^{X_{i-1}} \cdot (1 - 2^{-X_{i-1}})]$$

$$= \mathbb{E}[2 + 2^{X_{i-1}} - 1] = \mathbb{E}[2^{X_{i-1}}] + 1 = i + 1$$

$\mathbb{E}[2^X] = m + 1$

Claim. $\mathrm{Var}[2^X] \le m^2/2$

- By Chebyshev, $\Pr[|\tilde{m} - m| \ge \varepsilon m] \le \frac{\mathrm{Var}[\tilde{m}]}{(\varepsilon \cdot m)^2} \le \frac{1}{2\varepsilon^2}$
- Idea: to reduce variance, keep t independent counters and average their estimates.

Morris Algorithm: Improvement

Morris Algorithm

1. Initialize t independent counters $X \leftarrow 0$
2. For each element, increment each X by 1 w. p. 2^{-X}
3. Return $\tilde{m} =$ the average of $2^X - 1$ over all counters

- Then $E[\tilde{m}]$ remains m
- But Var$[\tilde{m}]$ is $\frac{1}{\mathrm{t}} \cdot \mathrm{Var}[2^X]$

$\mathbb{E}[2^X] = m + 1$

Claim. $\mathrm{Var}[2^X] \leq m^2/2$

- By Chebyshev, $\Pr[|\tilde{m} - m| \geq \varepsilon m] \leq \frac{\mathrm{Var}[\tilde{m}]}{(\varepsilon \cdot m)^2} \leq \frac{1}{2t\varepsilon^2}$
- It is sufficient to set $t = O\left(\frac{1}{\varepsilon^2}\right)$

Frequency Moments Estimation

Input: a stream $\langle a_1, a_2, \dots, a_m \rangle \in [n]^m$

- The frequency vector of the stream is $f = (f_1, \dots, f_n)$, where f_i is the number of times i appears in the stream
- The p-th frequency moment is $F_p = ||f||_p^p = \sum_{i=1}^n f_i^p$

  F_0 is the number of nonzero entries of f (# of distinct elements)

  $F_1 = m$ (# of elements in the stream)

  $F_2 = ||f||_2^2$ is a measure of non-uniformity

  used e.g. for anomaly detection in network analysis

  $F_\infty = \max_i f_i$ is the most frequent element

Goal: Estimate F_p up to a multiplicative factor $(1 \pm \varepsilon)$ with probability $\geq 2/3$

Estimating F_2 [Alon Matias Szegedy 96]

Input: a stream $\langle a_1, a_2, \dots, a_m \rangle \in [n]^m$

Goal: Estimate F_2 up to a multiplicative factor $(1 \pm \varepsilon)$ with probability $\geq \frac{2}{3}$

Today: $O(\varepsilon^{-2} (\log m + \log n))$ space algorithm

AMS Algorithm (initial version)

1. Sample a hash function $h : [n] \to \{-1,1\}$ from a 4-wise independent family
2. Initialize $X \leftarrow 0$
3. For each element a, increment X by $h(a)$ ← Add or subtract 1
4. Return X^2.

- Let $Z = (z_1, \dots, z_n)$, where $z_i = h(i)$
- Then, at the end, $X = Z \cdot f = \sum_{i \in [n]} z_i f_i$
- Let's compute the expectation and variance of X^2

The expectation of X^2

AMS Algorithm (initial version)

1. Sample a hash function $h : [n] \rightarrow \{-1,1\}$ from a 4-wise independent family
2. Initialize $X \leftarrow 0$
3. For each element a, increment X by $h(a)$ ← Add or subtract 1
4. Return X^2.

$\mathbb{E}[X^2] = F_2$

- Let $Z = (z_1, \ldots, z_n)$, where $z_i = h(i)$
- Then, at the end, $X = Z \cdot f = \sum_{i \in [n]} z_i f_i$

$$X^2 = \left(\sum_{i \in [n]} z_i f_i \right)^2 = \sum_{i \in [n]} \sum_{j \in [n]} z_i z_j f_i f_j$$

$$\mathbb{E}[X^2] = \sum_{i \in [n]} \sum_{j \in [n]} \mathbb{E}[z_i z_j] \, f_i f_j$$

by linearity of expectation

$$= \sum_{i \in [n]} \mathbb{E}[z_i^2] f_i^2 + \sum_{i \neq j} \mathbb{E}[z_i] \cdot \mathbb{E}[z_j] \, f_i f_j$$

z_i's are 2-wise independent

$$= \sum_{i \in [n]} f_i^2 = F_2$$

$z_i^2 = 1$

The variance of X^2

AMS Algorithm (initial version)

1. Sample a hash function $h : [n] \to \{-1,1\}$ from a 4-wise independent family
2. Initialize $X \leftarrow 0$
3. For each element a, increment X by $h(a)$ ← Add or subtract 1
4. Return X^2.

- Let $Z = (z_1, \dots, z_n)$, where $z_i = h(i)$
- Then, at the end, $X = Z \cdot f = \sum_{i \in [n]} z_i f_i$

$\mathbb{E}[X^2] = F_2$

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

$$= \sum_{i,j,k,\ell \in [n]} \mathbb{E}[z_i z_j z_k z_\ell]\, f_i f_j\, f_k f_\ell \; - F_2^2$$

by linearity of expectation

$$= \sum_{i \in [n]} \mathbb{E}[z_i^4] f_i^4 + 6 \sum_{i<j} \mathbb{E}[z_i^2] \cdot \mathbb{E}[z_j^2]\, f_i^2 f_j^2 - F_2^2$$

z_i's are 4-wise independent

$$= \sum_{i \in [n]} f_i^4 + 6 \sum_{i<j} f_i^2 f_j^2 - F_2^2 \le 4 \sum_{i<j} f_i^2 f_j^2 \le 2F_2^2$$

$z_i^2 = 1$

Estimating F_2 [Alon Matias Szegedy 96]

AMS Algorithm

1. $t \leftarrow 20/\varepsilon^2$

Run t copies of initial algorithm and average the results

2. Sample t independent hash functions $h_i: [n] \to \{-1,1\}$ from a 4-wise independent family
3. Initialize t counters $X_i \leftarrow 0$
4. For each element a, increment each X_i by $h_i(a)$
5. Return $Y = \frac{1}{t}\sum_{i\in[t]} X_i^2$.

- We proved: $\mathbb{E}[X_i^2] = F_2$ and $\mathrm{Var}[X_i^2] \le 2F_2^2$
- Then $\mathbb{E}[Y] = \mathbb{E}[X_i^2] = F_2$ and $\mathrm{Var}[Y] = \frac{1}{t}\mathrm{Var}[X_i^2] \le \frac{2}{t}F_2^2$

X_i^2 are independent

- Correctness: $\Pr[|Y - F_2| \ge \varepsilon \cdot F_2] = \Pr[|Y - \mathbb{E}[Y]| \ge \varepsilon \cdot F_2]$

$$\le \frac{\mathrm{Var}[Y]}{(\varepsilon \cdot F_2)^2} \le \frac{2F_2^2}{t \cdot \varepsilon^2 \cdot F_2^2} = \frac{1}{10}$$

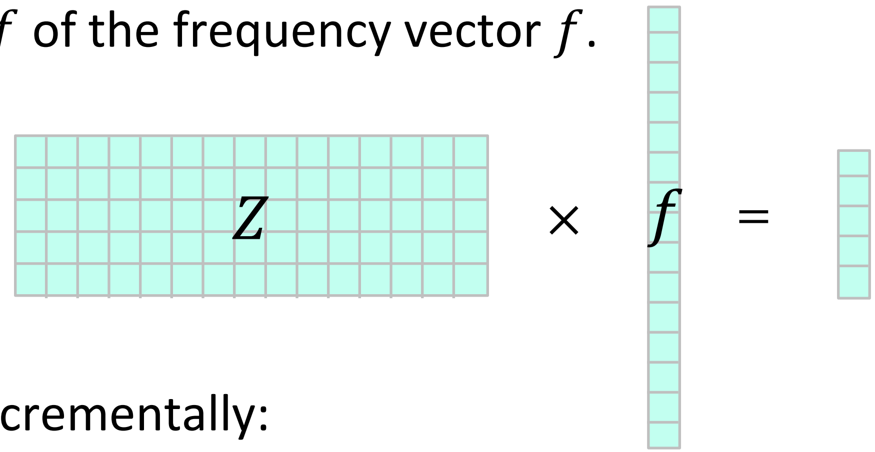
Chebyshev

- Space: $O(t \log n)$ to store hash functions + $O(t \log m)$ to store X_i's

$$O\left(\frac{1}{\varepsilon^2}(\log n + \log m)\right)$$

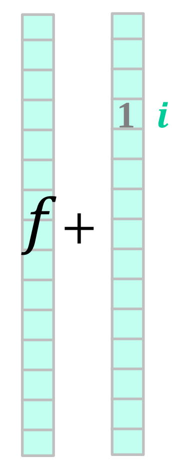
General Technique: Linear Sketching

- A sketching algorithm stores a random matrix $Z \in \mathbb{R}^{t \times n}$ where $t \ll n$ and computes projection Zf of the frequency vector f.



- Zf can be computed incrementally:
  - Suppose we have a sketch Zf of the current frequency vector f.
  - If we see an occurrence of i, the new frequency vector is $f' = f + e_i$.
  - We update the sketch by adding column i of Z to Zf :

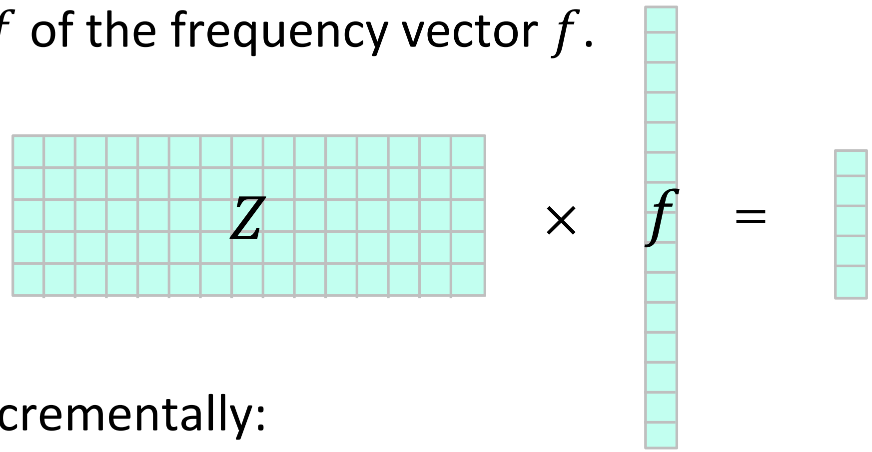
$$Zf' = Z(f + e_i) = Zf + Ze_i = Zf + (i\text{-th column of } Z)$$



- In the AMS algorithm, Z was a matrix of -1s and 1s,
  with each row chosen independently from a 4-wise independent family

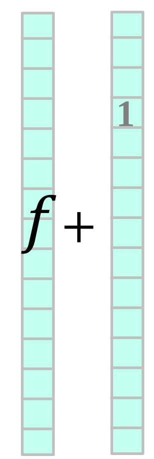
General Technique: Linear Sketching

- A sketching algorithm stores a random matrix $Z \in \mathbb{R}^{t \times n}$ where $t \ll n$ and computes projection Zf of the frequency vector f.



- Zf can be computed incrementally:
  - Suppose we have a sketch Zf of the current frequency vector f.
  - If we see an occurrence of i, the new frequency vector is $f' = f + e_i$.
  - We update the sketch by adding column i of Z to Zf :

$$Zf' = Z(f + e_i) = Zf + Ze_i = Zf + (i\text{-th column of } Z)$$



- In general: Need to chose the random matrix so that
  - relevant properties of f can be estimated with high probability from Zf
  - Z can be stored efficiently

Multipurpose Sketches: Problems

Input: a stream $\langle a_1, a_2, \dots, a_m \rangle \in [n]^m$

- The frequency vector of the stream is $f = (f_1, \dots, f_n)$, where f_i is the number of times i appears in the stream

Goal: to maintain data structures that can answer the following queries

- Point Query: For $i \in [n]$, estimate f_i
- Range Query: For $i, j \in$ [n], estimate $f_i + f_{i+1} + \dots + f_j$
- Quantile Query: For $\phi \in [0, 1]$, find j with $f_1 + \dots + f_j \approx \phi m$
- Heavy Hitters Query: For $\phi \in [0, 1]$, find all i with $f_i \geq \phi m$.

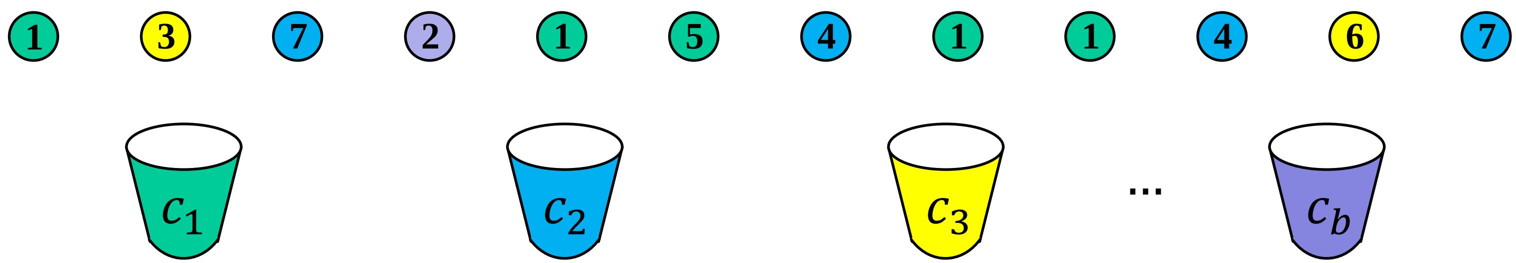
Desired accuracy: $\pm\varepsilon m$ with error probability δ

Based on Andrew McGregor's slides: https://people.cs.umass.edu/~mcgregor/711S18/vectors-3.pdf

Initial Solution to Point Queries

- We could maintain the whole frequency vector $(f_1, \dots, f_n)$
- Then, on query i, we can output f_i

Idea: Group counts for some numbers together



If i falls into bucket j, then $f_i \le c_j$.

Point Query Algorithm (initial version)

1. Sample a hash function $h : [n] \to [b]$ from a 2-wise independent family
2. Initialize counters $c_1, \dots, c_b$ to 0
3. For each element a, increment $c_{h(a)}$ by 1.
4. To answer a point query i, return $c_{h(i)}$.

Never underestimate