# CIS 700: "algorithms for Big Data"

# Lecture 2: Streaming

Slides at http://grigory.us/big-data-class.html


**Grigory Yaroslavtsev**

http://grigory.us

Penn
UNIVERSITY of PENNSYLVANIA

# Recap

- (Markov) For every $c > 0$ (and non-negative $\boldsymbol{X}$):
$$\Pr\left[\boldsymbol{X} \geq c\ \mathbb{E}[\boldsymbol{X}]\right] \leq \frac{1}{c}$$
- (Chebyshev) For every $c > 0$:
$$\Pr\left[|\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}]| \geq c\ \mathbb{E}[\boldsymbol{X}]\right] \leq \frac{Var[\boldsymbol{X}]}{(c\ \mathbb{E}[\boldsymbol{X}])^2}$$
- (Chernoff) Let $\boldsymbol{X}_1 \dots \boldsymbol{X}_{\boldsymbol{t}}$ be independent and identically distributed r.vs with range [0, c] and expectation $\mu$. Then if $X = \frac{1}{\boldsymbol{t}} \sum_i X_i$ and $1 > \delta > 0$,
$$\Pr[|X - \mu| \geq \delta\mu] \leq 2\exp\left(-\frac{\boldsymbol{t}\mu\delta^2}{3c}\right)$$

# This week

- Approximate counting (Morris's alg.) continued
- Approximate Median
- Alon-Mathias-Szegedy Sampling
- Frequency Moments
- Distinct Elements
- Count-Min

# Morris's Algorithm

- (**Very hard, "Count the number of items"**)
  - What is the total number of items up to error $\pm\, \epsilon n$?
  - You have $O(\log \log n / \epsilon^2)$ space and can be completely wrong with some small probability

# Morris's Algorithm: Alpha-version

Maintains a counter $X$ using $\log \log n$ bits

- Initialize $X$ to 0
- When an item arrives, increase X by 1 with probability $\frac{1}{2^X}$
- When the stream is over, output $2^X - 1$

Claim: $\mathbb{E}[2^X] = n + 1$

# Morris's Algorithm: Alpha-version

Maintains a counter $X$ using $\log\log n$ bits

- Initialize $X$ to 0, when an item arrives, increase X by 1 with probability $\frac{1}{2^X}$

Claim: $\mathbb{E}[2^X] = n + 1$

- Let the value after seeing $n$ items be $X_n$

$$\mathbb{E}[2^{X_n}] = \sum_{j=0}^{\infty} \Pr[X_{n-1} = j\,]\,\mathbb{E}[2^{X_n} | X_{n-1} = j]$$

$$= \sum_{j=0}^{\infty} \Pr[X_{n-1} = j\,]\left(\frac{1}{2^j}\,2^{j+1} + \left(1 - \frac{1}{2^j}\right)2^j\right)$$

$$= \sum_{j=0}^{\infty} \Pr[X_{n-1} = j\,]\left(2^j + 1\right) = 1 + \mathbb{E}[2^{X_{n-1}}]$$

# Morris's Algorithm: Alpha-version

Maintains a counter $X$ using $\log\log n$ bits

- Initialize $X$ to 0, when an item arrives, increase X by 1 with probability $\frac{1}{2^X}$

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

$$\mathbb{E}[2^{2X_n}] = \sum_{j=0}^{\infty} \Pr[2^{X_{n-1}} = j\,]\mathbb{E}[2^{2X_n}|2^{X_{n-1}} = j]$$

$$= \sum_{j=0}^{\infty} \Pr[2^{X_{n-1}} = j\,]\left(\frac{1}{j}\,4\,j^2 + \left(1\ - \frac{1}{j}\right)j^2\right)$$

$$= \sum_{j=0}^{\infty} \Pr[2^{X_{n-1}} = j\,](j^2 + 3j) = \mathbb{E}[2^{2X_{n-1}}] + 3\mathbb{E}[2^{X_{n-1}}]$$

$$= 3\,\frac{(\mathrm{n}\ - 1)^2}{2} + 3(\mathrm{n} - 1)/2\ + 1 + 3\mathrm{n}$$

# Morris's Algorithm: Alpha-version

Maintains a counter $X$ using $\log\log n$ bits

- Initialize $X$ to 0, when an item arrives, increase X by 1 with probability $\frac{1}{2^X}$
- $\mathbb{E}[2^X] = \mathrm{n} + 1, Var[2^X] = O(n^2)$
- Is this good?

# Morris's Algorithm: Beta-version

Maintains $t$ counters $X^1, \ldots, X^t$ using $\log \log n$ bits for each

- Initialize $X^{i'}s$ to 0, when an item arrives, increase each $X^i$ by 1 independently with probability $\frac{1}{2^{X^i}}$
- Output $\mathrm{Z} = \frac{1}{t}(\sum_{i=1}^{t} 2^{X^i} - 1)$
- $\mathbb{E}[2^{X_i}] = \mathrm{n} + 1, Var[2^{X_i}] = O(n^2)$
- $Var[Z] = Var\left(\frac{1}{t}\sum_{j=1}^{t} 2^{X^j} - 1\right) = O\left(\frac{n^2}{t}\right)$
- Claim: If $t \geq \frac{c}{\epsilon^2}$ then $\Pr[|Z - n| > \epsilon n] < 1/3$

# Morris's Algorithm: Beta-version

Maintains $t$ counters $X^1, \ldots, X^t$ using $\log \log n$ bits for each

- Output $Z = \frac{1}{t}(\sum_{i=1}^{t} 2^{X^i} - 1)$
- $Var[Z] = Var\left(\frac{1}{t}\sum_{j=1}^{t} 2^{X^j} - 1\right) = O\left(\frac{n^2}{t}\right)$
- Claim: If $t \geq \frac{c}{\epsilon^2}$ then $\Pr[|Z - n| > \epsilon n] < 1/3$
  - $\Pr[|Z - n| > \epsilon\, n] < \frac{Var[Z]}{\epsilon^2 n^2} = O\left(\frac{n^2}{t}\right) \cdot \frac{1}{\epsilon^2 n^2}$
  - If $t \geq \frac{c}{\epsilon^2}$ we can make this at most $\frac{1}{3}$

# Morris's Algorithm: Final

- What if I want the probability of error to be really small, i.e. $\Pr[|Z - n| > \epsilon\, n] \le \delta$?
- Same Chebyshev-based analysis: $t = O\left(\frac{1}{\epsilon^2 \delta}\right)$
- Do these steps $m = O\left(\log \frac{1}{\delta}\right)$ times independently in parallel and output the median answer.
- Total space: $O\left(\frac{\log \log n \cdot \log \frac{1}{\delta}}{\epsilon^2}\right)$

# Morris's Algorithm: Final

- Do these steps $m = O\left(\log \frac{1}{\delta}\right)$ times independently in parallel and output the median answer

$$Z^{med} = median(Z_1, \ldots, Z_m)$$

- Each $Z_i$ computed as before:

Maintain $t$ counters $X^1, \ldots, X^t$ using $\log \log n$ bits for each

- Initialize $X^{i'}s$ to 0, when an item arrives, increase each $X^i$ by 1 independently with probability $\frac{1}{2^{X^i}}$
- Output $\mathrm{Z} = \frac{1}{t}\left(\sum_{i=1}^{t} 2^{X^i} - 1\right)$

# Morris's Algorithm: Final Analysis

Claim: $\Pr\left[\left|Z^{med} - n\right| > \epsilon\, n\right] \le \delta$

- Let $Y_i$ be an indicator r.v. for the event that $|Z_i - n| \le \epsilon n$, where $Z_i$ is the i-th trial.
- Let $Y = \sum_i Y_i$.
- $\Pr\left[\left|Z^{med} - n\right| > \epsilon n\right] \le \Pr\left[Y \le \frac{m}{2}\right] \le \Pr\left[|Y - \mathbb{E}[Y]| \ge \frac{m}{6}\right] \le \Pr\left[|Y - \mathbb{E}[Y]| \ge \frac{\mu}{4}\right] \le \exp\left(-c\,\frac{1}{4^2}\frac{2m}{3}\right) < \exp\left(-c' \log\frac{1}{\delta}\right) < \delta$

# Data Streams

- Stream: $\boldsymbol{m}$ elements from universe $[\boldsymbol{n}] = \{1, 2, \dots, \boldsymbol{n}\}$, e.g.

$$\langle x_1, x_2, \dots, x_{\boldsymbol{m}} \rangle = \langle 5, 8, 1, 1, 1, 4, 3, 5, \dots, 10 \rangle$$

- Example:

# Approximate Median

- $S = \{x_1, \ldots, x_m\}$ (all distinct) and let
$$rank(y) = |x \in S : x \leq y|$$
- Problem: Find $\epsilon$-approximate median, i.e. $y$ such that
$$\frac{m}{2} - \epsilon m < rank(y) < \frac{m}{2} + \epsilon m$$
- Exercise: Can we approximate the value of the median with additive error $\pm\epsilon n$ in sublinear time?
- Algorithm: Return the median of a sample of size $t$ taken from $S$ (with replacement).

# Approximate Median

- Problem: Find $\epsilon$-approximate median, i.e. $y$ such that
$$\frac{m}{2} - \epsilon m < rank(y) < \frac{m}{2} + \epsilon m$$
- Algorithm: Return the median of a sample of size $t$ taken from $S$ (with replacement).
- Claim: If $t = \frac{7}{\epsilon^2} \log \frac{2}{\delta}$ then this algorithm gives $\epsilon$-median with probability $1 - \delta$

# Approximate Median

- Partition $S$ into 3 groups

$$\boldsymbol{S_L} = \left\{x \in S: rank(x) \le \frac{m}{2} - \epsilon m\right\}$$

$$\boldsymbol{S_M} = \left\{x \in S: \frac{m}{2} - \epsilon m \le rank(x) \le \frac{m}{2} + \epsilon m\right\}$$

$$\boldsymbol{S_U} = \left\{x \in S: rank(x) \ge \frac{m}{2} + \epsilon m\right\}$$

- **Key fact**: If less than $\frac{t}{2}$ elements from each of $\boldsymbol{S_L}$ and $\boldsymbol{S_U}$ are in sample then its median is in $\boldsymbol{S_M}$
- Let $\boldsymbol{X}_i = 1$ if $i$-th sample is in $\boldsymbol{S_L}$ and 0 otherwise.
- Let $\boldsymbol{X} = \sum_i \boldsymbol{X}_i$. By Chernoff, if $t > \frac{7}{\epsilon^2} \log \frac{2}{\delta}$

$$\Pr\left[\boldsymbol{X} \ge \frac{t}{2}\right] \le \Pr[\boldsymbol{X} \ge (1+\epsilon)\mathbb{E}[\boldsymbol{X}]] \le e^{-\frac{\epsilon^2\left(\frac{1}{2}-\epsilon\right)t}{3}} \le \frac{\delta}{2}$$

- Same for $\boldsymbol{S_U}$ + union bound $\Rightarrow$ error probability $\le \delta$

# Data Streams

- Stream: $\boldsymbol{m}$ elements from universe $[\boldsymbol{n}] = \{1, 2, \ldots, \boldsymbol{n}\}$, e.g.

$$\langle x_1, x_2, \ldots, x_{\boldsymbol{m}} \rangle = \langle 5, 8, 1, 1, 1, 4, 3, 5, \ldots, 10 \rangle$$

- $f_i$ = frequency of $i$ in the stream = # of occurrences of value $i$

$$f = \langle f_1, \ldots, f_{\boldsymbol{n}} \rangle$$

# AMS Sampling

- Problem: Estimate $\sum_{i\in[n]} g(f_i)$, for an arbitrary function $g$ with $g(0) = 0$.
- Estimator: Sample $x_{\boldsymbol{J}}$, where $\boldsymbol{J}$ is sampled uniformly at random from $[m]$ and compute:

$$r = \left|\{j \geq \boldsymbol{J} : x_j = x_{\boldsymbol{J}}\}\right|$$

Output: $\boldsymbol{X} = m(g(r) - g(r - 1))$

- Expectation:

$$\mathbb{E}[\boldsymbol{X}] = \sum_i \Pr\left[x_{\boldsymbol{J}} = i\right] \mathbb{E}[\boldsymbol{X} | x_{\boldsymbol{J}} = i]$$

$$= \sum_i \frac{f_i}{m}\left(\sum_{r=1}^{f_i} \frac{m\big(g(r) - g(r-1)\big)}{f_i}\right) = \sum_i g(f_i)$$

# Frequency Moments

- Define $F_k = \sum_i f_i^k$ for $k \in \{0,1,2,\ldots\}$
  - $F_0 =$ # number of distinct elements
  - $F_1 =$ # elements
  - $F_2$ = "Gini index", "surprise index"

# Frequency Moments

- Define $F_k = \sum_i f_i^k$ for $k \in \{0,1,2, \dots\}$
- Use AMS estimator with $\boldsymbol{X} = m\left(r^k - (r-1)^k\right)$
$$\mathbb{E}[\boldsymbol{X}] = F_k$$
- Exercise: $0 \leq \boldsymbol{X} \leq m\,k\,f_*^{k-1}$, where $f_* = \max_i f_i$
- Repeat $t$ times and take average $\widehat{\boldsymbol{X}}$. By Chernoff:
$$\Pr\left[\left|\widehat{\boldsymbol{X}} - F_k\right| \geq \epsilon F_k\right] \leq 2\exp\left(-\frac{tF_k\epsilon^2}{3m\,k\,f_*^{k-1}}\right)$$
- Taking $t = \frac{3mkf_*^{k-1}\log\frac{1}{\delta}}{\epsilon^2 F_k}$ gives $\Pr\left[\left|\widehat{\boldsymbol{X}} - F_k\right| \geq \epsilon F_k\right] \leq \delta$

# Frequency Moments

- Lemma:

$$\frac{m f_*^{k-1}}{F_k} \leq n^{1-1/k}$$

- Result: $t = \frac{3mk f_*^{k-1} \log\frac{1}{\delta}}{\epsilon^2 F_k} = O\left(\frac{k n^{1-\frac{1}{k}} \log\frac{1}{\delta}}{\epsilon^2} \log n\right)$ memory suffices for $(\epsilon, \delta)$-approximation of $F_k$
- Question: What if we don't know $m$?
- Then we can use probabilistic guessing (similar to Morris's algorithm), replacing $\log n$ with $\log nm$.

# Frequency Moments

- Lemma:

$$\frac{m f_*^{k-1}}{F_k} \leq n^{1-1/k}$$

- Exercise: $F_k \geq n \left(\frac{m}{n}\right)^k$ (Hint: worst-case when $f_1 = \cdots = f_n = \frac{m}{n}$. Use convexity of $g(x) = x^k$).
- Case 1: $f_*^k \leq n \left(\frac{m}{n}\right)^k$

$$\frac{m f_*^{k-1}}{F_k} \leq \frac{m n^{1-\frac{1}{k}} \left(\frac{m}{n}\right)^{k-1}}{n \left(\frac{m}{n}\right)^k} = n^{1-\frac{1}{k}}$$

# Frequency Moments

- Lemma:

$$\frac{mf_*^{k-1}}{F_k} \leq n^{1-1/k}$$

- Case 2: $f_*^k \geq n\left(\frac{m}{n}\right)^k$

$$\frac{mf_*^{k-1}}{F_k} \leq \frac{mf_*^{k-1}}{f_*^k} \leq \frac{m}{f_*} \leq \frac{m}{n^{\frac{1}{k}}\left(\frac{m}{n}\right)} = n^{1-\frac{1}{k}}$$

# Hash Functions

- Definition: A family $H$ of functions from $A \rightarrow B$ is $k$-wise independent if for any distinct $x_1, \ldots, x_k \in A$ and $i_1, \ldots i_k \in B$:

$$\Pr_{h \in_R H}[h(x_1) = i_1, h(x_2) = i_2, \ldots, h(x_k) = i_k] = \frac{1}{|B|^k}$$

- Example: If $A \subseteq \{0, \ldots, p-1\}, B = \{0, \ldots, p-1\}$ for prime $p$

$$H = \left\{ h(x) = \sum_{i=0}^{k-1} a_i x^i \ mod\ p: 0 \leq a_0, a_1, \ldots, a_{k-1} \leq p-1 \right\}$$

is a $k$-wise independent family of hash functions.

# Linear Sketches

- Sketching algorithm: picks a random matrix $Z \in R^{k \times n}$, where $k \ll n$ and computes $Zf$.
- Can be incrementally updated:
  - We have a sketch $Zf$
  - When $i$ arrives, new frequencies are $f' = f + e_i$
  - Updating the sketch:

$$\begin{aligned} Zf' &= Z(f + e_i) = Zf + Ze_i \\ &= Zf + (i - th\ column\ of\ Z) \end{aligned}$$

- Need to choose random matrices carefully

# $F_2$

- Problem: $(\epsilon, \delta)$-approximation for $F_2 = \sum_i f_i^2$
- Algorithm:
  - Let $Z \in \{-1,1\}^{k \times n}$, where entries of each row are 4-wise independent and rows are independent
  - Don't store the matrix: $k$ 4-wise independent hash functions $\sigma$
  - Compute $Zf$, average squared entries "appropriately"
- Analysis:
  - Let $s$ be any entry of $Zf$.
  - Lemma: $\mathbb{E}[s^2] = F_2$
  - Lemma: $\mathrm{Var}[s^2] \leq 4F_2^2$

# $F_2$: Expectaton

- Let $\sigma$ be a row of $Z$ with entries $\sigma_i \in_R \{-1,1\}$.

$$\mathbb{E}[s^2] = \mathbb{E}\left[\left(\sum_{i=1}^{n} \sigma_i f_i\right)^2\right]$$

$$= \mathbb{E}\left(\sum_{i=1}^{n} \sigma_i^2 f_i^2 + \sum_{i \neq j} \mathbb{E}[\sigma_i \sigma_j f_i f_j]\right)$$

$$= \mathbb{E}\left(\sum_{i=1}^{n} f_i^2 + \sum_{i \neq j} \mathbb{E}[\sigma_i \sigma_j] f_i f_j\right)$$

$$= F_2 + \sum_{i \neq j} \mathbb{E}[\sigma_i] \mathbb{E}\left[\sigma_j\right] f_i \; f_j = F_2$$

- We used 2-wise independence for $\mathbb{E}\left[\sigma_i \sigma_j\right] = \mathbb{E}[\sigma_i] \mathbb{E}\left[\sigma_j\right]$.

# $F_2$: Variance

$$\mathbb{E}[(X^2 - \mathbb{E}X^2)^2] = \mathbb{E}\left(\sum_{i \neq j} \sigma_i \sigma_j f_i f_j\right)^2$$

$$= \mathbb{E}\left(2 \sum_{i \neq j} \sigma_i^2 \sigma_j^2 f_i^2 f_j^2 + 4 \sum_{i \neq j \neq k} \sigma_i^2 \sigma_j \sigma_k f_i^2 f_j f_k\right.$$

$$\left. + 24 \sum_{i<j<k<l} \sigma_i \sigma_j \sigma_k \sigma_l f_i f_j f_k f_l\right)$$

$$= 2 \sum_{i \neq j} f_i^2 f_j^2 + 4 \sum_{i \neq j \neq k} \mathbb{E}\left[\sigma_j \sigma_k\right] f_i^2 f_j f_k$$

$$+ 24 \sum_{i<j<k<l} \mathbb{E}[\sigma_i \sigma_j \sigma_k \sigma_l] f_i f_j f_k f_l \leq 2\, F_2^2$$

- $\mathbb{E}\left[\sigma_i \sigma_j \sigma_k \sigma_l\right] = \mathbb{E}\left[\sigma_i\right] \mathbb{E}\left[\sigma_j\right] \mathbb{E}\left[\sigma_k\right] \mathbb{E}\left[\sigma_l\right] = 0$ by 4-wise independence

# $F_0$: Distinct Elements

- Problem: $(\epsilon, \delta)$-approximation for $F_0 = \sum_i f_i^0$
- Simplified: For fixed $T > 0$, with prob. $1 - \delta$ distinguish:

$$F_0 > (1+\epsilon)T \text{ vs. } F_0 < (1-\epsilon)T$$

- Original problem reduces by trying $O\left(\frac{\log n}{\epsilon}\right)$ values of T:

$$T = 1, (1+\epsilon), (1+\epsilon)^2, \ldots, n$$

# $F_0$: Distinct Elements

- Simplified: For fixed $T > 0$, with prob. $1 - \delta$ distinguish:

$$F_0 > (1+\epsilon)T \text{ vs. } F_0 < (1-\epsilon)T$$

- Algorithm:
  - Choose random sets $S_1, \ldots, S_k \subseteq [n]$ where $\Pr\left[i \in S_j\right] = \frac{1}{T}$
  - Compute $s_j = \sum_{i \in S_j} f_i$
  - If at least $k/e$ of the values $s_j$ are zero, output $F_0 < (1-\epsilon)T$

# $F_0 > (1+\epsilon)T$ vs. $F_0 < (1-\epsilon)T$

- Algorithm:
  - Choose random sets $S_1, \ldots, S_k \subseteq [n]$ where $\Pr[i \in S_j] = \frac{1}{T}$
  - Compute $s_j = \sum_{i \in S_j} f_i$
  - If at least $k/e$ of the values $s_j$ are zero, output $F_0 < (1-\epsilon)T$
- Analysis:
  - If $F_0 > (1+\epsilon)T$, then $\Pr[s_j = 0] < \frac{1}{e} - \frac{\epsilon}{3}$
  - If $F_0 < (1-\epsilon)T$, then $\Pr[s_j = 0] > \frac{1}{e} + \frac{\epsilon}{3}$
  - Chernoff: $k = O\left(\frac{1}{\epsilon^2} \log \frac{1}{\delta}\right)$ gives correctness w.p. $1-\delta$

# $F_0 > (1+\epsilon)T$ vs. $F_0 < (1-\epsilon)T$

- Analysis:
  - If $F_0 > (1+\epsilon)T$, then $\Pr[s_j = 0] < \frac{1}{e} - \frac{\epsilon}{3}$
  - If $F_0 < (1-\epsilon)T$, then $\Pr[s_j = 0] > \frac{1}{e} + \frac{\epsilon}{3}$
- If $T$ is large and $\epsilon$ is small then:

$$\Pr[s_j = 0] = \left(1 - \frac{1}{T}\right)^{F_0} \approx e^{-\frac{F_0}{T}}$$

- If $F_0 > (1+\epsilon)T$:

$$e^{-\frac{F_0}{T}} \le e^{-(1+\epsilon)} \le \frac{1}{e} - \frac{\epsilon}{3}$$

- If $F_0 < (1-\epsilon)T$:

$$e^{-\frac{F_0}{T}} \ge e^{-(1-\epsilon)} \ge \frac{1}{e} + \frac{\epsilon}{3}$$