Lecture 9

Streaming algorithms

Also good introduction to sampling and estimation algorithms.

Setting

a set of objects / items / tokens come in a stream / online fashion

$e_1, e_2, \ldots, e_m$

Examples:

- e_i is a number from $[n]$
- e_i is an edge (u,v) in a graph
- e_i is a vector / point in $\mathbb{R}^d$
- e_i is a row / column of a matrix

- e_i is a matrix.

Assumption: m is large and unknown. Cannot afford to store all of $e_1, \dots, e_m$ in memory.

We have say, B space where $B \ll m$ (assume one token is one unit for now).

What functions of the input can we compute?

Depends on B and we have various trade off between efficiency, accuracy etc.

<u>Simple</u> <u>but</u> <u>powerful</u> <u>setting</u>.

each $e_i \in [n]$ hence a non-neg integer from say large range $0..n-1$ or $1..n$.

Ex: $n = 1000$

<u>stream</u> 5, 3, 1, 1, 2, 2, 10, 5, 90

Implicitly defines a vector $\bar{f} \in Z^n$ that starts at all $\bar{0}$ vector

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Each e_i adds 1 to vector

$$f_{e_i}$$

$$5, 3, 1, 1, 2, 2, 10, 5, 90$$

$$\begin{bmatrix}0\\0\\\vdots\\0\end{bmatrix} \to \begin{bmatrix}0\\0\\0\\0\\1\\0\\\vdots\\1\end{bmatrix} \to \begin{bmatrix}0\\0\\1\\0\\1\\\vdots\end{bmatrix} \begin{bmatrix}0\\0\\\vdots\end{bmatrix} -$$

Frequency moment estimation

Given stream it define f at the end of the stream.

Want to estimate / compute kth frequency moment

$$F_k = \sum_{i=1}^{n} f_i^k \quad \text{or} \quad \left(\sum_{i=1}^{n} f_i^k \right)^{\frac{1}{k}}.$$

- $k=0$ # of distinct elements
- $k=1$ m. easy
- $k=2$ l_2 norm which is very important as we will see.
- $k=\infty$ largest coordinate / heavy hitters.

$k=0$, $k=2$, $k=\infty$ most important.

If we can store all e_i or $\bar{f}$ then problems are of course trivial.

<u>Question</u> Can we compute/estimate with $B \ll m$.

Yes! with $B = \tilde{O}(1)$!

But requires <u>randomization</u> and <u>approximation</u>. Can show that without either not feasible with $o(n)$ memory.

Reservoir Sampling

Given stream $e_1, e_2, \ldots, e_m$ with m unknown. Want to have a uniformly random sample.

- $s \leftarrow \phi$
- $m \leftarrow 0$
- While (stream has not ended)

    $m \leftarrow m+1$

    $e_m \leftarrow$ current element

    with prob $\frac{1}{m}$.

        $s \leftarrow e_m$

  end while
- output s.

If we want K samples with replacement can use above independently.

K samples <u>without</u> replacement.

$S \leftarrow \phi$, $m \leftarrow 0$

While (stream is not done)

$m \leftarrow m+1$

if $m \leq K$ add e_m to S.

else

with prob $\frac{K}{m}$

replace a random element of S with e_m

end while

Output S.

Exercise: Prove correctness.

See notes to learn about "weighted" sampling.

Distinct Element Estimation

Given stream $e_1, e_2, \ldots, e_m$ where each $e_i \in [n]$, want to know F_0 = # of distinct elements seen.

Ex: $1, 1, 3, 1, 1, 1, 1, 1, 3, 5, 5, 5, 1, 2$

4 is # of distinct elements.

n is large m is large.

Ex: An interesting application.

Packets through a high-speed internet switch.

(src, dest, content)

↗ IP address of source and destination

$\underline{\underline{128}}$ bits in IPv6.

So n is 2^{128} technically.

Q: How many distinct source addresses?

Offline solution

Can store distinct elements using a dictionary data structure.

$\Theta(k)$ space where $k = F_0$.

- k can be very large
- k unknown
- data structure may not be fast enough.

But we can get exact answer.

Can show that one cannot get $(1-\varepsilon)$ approximation deterministically with sub-linear space.

But with randomization can get ~~an~~ a $(1-\varepsilon)$ approx estimate with probability $(1-\delta)$ in

$$O\left(\frac{1}{\varepsilon^2}\log\frac{1}{\delta}\right)\cdot \text{polylog}(n) \text{ space.}$$

<u>How?</u>

- An old classical result from 1985 based on hashing and refined over years.
- A very recent surprisingly simple one based on sampling.

Hashing based Algorithm

We will assume we have access to an "ideal" hash function

$h : [n] \to [0,1]$

↗ real interval.

You can assume that instead of real interval we are hashing to $[n^3]$ or some large n^c

Distinct Elements Hashing

- Pick random hash function $h : [n] \to [0,1]$
- $z \leftarrow \infty$
- While (stream is not done)
  $z \leftarrow \min(h(e_i), z)$.
- Output $\frac{1}{z} - 1$.

Memory: Only one number!

Lemma: Suppose $X_1, X_2, \ldots, X_k$ are independent random variables that are uniformly distributed in $[0,1]$.

Let $Y = \min_i X_i$.

Then (i) $E[Y] = \frac{1}{k+1}$

(ii) $Var(Y) = \frac{1}{(k+1)(k+2)}$

Proof: Let F_Y and f_Y be the cdf and pdf of Y.

Then it is easy to see that

$F_Y(t) = 0 \quad t \le 0 \quad F_Y(t) = 1 \quad t \ge 1$

$F_Y(t) = 1 - (1-t)^k$ for t in $(0,1)$.

$$f_Y(t) = k(1-t)^{k-1}$$

$$E[Y] = \int_0^1 (1-t)^k \, dt = -\frac{(1-t)^{k+1}}{k+1}\Bigg|_0^1$$

$$= \frac{1}{k+1} = \int_0^1 t\,k\,(1-t)^{k-1} dt$$

$$E[Y^2] = \int_0^1 t^2 k (1-t)^{k-1} dt$$

$$= \frac{2}{(k+1)(k+2)}$$

$$\mathrm{Var}(Y) = \frac{2}{(k+1)(k+2)} - \frac{1}{(k+1)^2}$$

$$= \frac{2k}{(k+1)^2(k+2)} \simeq \frac{2}{(k+1)(k+2)}$$

D.

Thus the algorithm has an exact estimator for F_0 but variance is too high to get a good approximation. What does this mean?

Lemma: Suppose $F_0 = k$.

For $\frac{1}{Z} - 1$ to be within $(1 \pm \varepsilon)$ of k we need Z to be within $\frac{O(\varepsilon)}{k+1}$ of true value $\frac{1}{k+1}$.

But if we use Markov or Chebyshev one sees that it does not work.

Variance reduction

When we have an exact estimator Y for a quantity with true value α and $Var(Y)$ is large we can obtain another estimator Y' s.t. $E[Y'] = \alpha$ and $Var(Y')$ is $\leq \frac{1}{n} Var(Y)$ by averaging independent estimators.

Claim: Let $Y_1, Y_2, \ldots, Y_n$ be independent and let $Y = \frac{1}{n} \sum_{i=1}^{n} Y_i$

$E[Y] = \frac{1}{n} \sum_i E[Y_i]$

and $Var(Y) = \frac{1}{n} \sum_{i=1}^{n} Var(Y_i)$.

Hence if each Y_i's are iid with mean μ and variance σ^2

Then $E[Y] = \mu$ and $Var(Y) = \frac{\sigma^2}{n}$.

How can we apply this to F_0 estimation?

We have an exact estimator

$\mu = K$. $Var(Z) = \frac{2}{(K+1)(K+2)}$.

Want an estimator Z' such that $Pr[(1-\varepsilon)K \leq Z' \leq (1+\varepsilon)K]$

$\geq \frac{9}{10}$. say.

$\varepsilon \in (0, \frac{1}{2})$ given.

- Run basic algorithm h times independently and in parallel
- Let $Z_1, Z_2, \ldots, Z_h$ be the min values.
- Let $Z = \frac{1}{h} \sum_{j=1}^{h} Z_j$
- Output $Y = \frac{1}{Z} - 1$

<u>claim:</u> $E[Z] = \frac{1}{k+1}$

<u>claim:</u> $Var(Z) \leq \frac{2}{h(k+1)(k+2)}$.

Via Chebyshev's inequality

$$\Pr\left[\left|Z - \frac{1}{k+1}\right| \geq \frac{\varepsilon}{3(k+1)}\right] \leq \frac{\frac{2}{h(k+1)(k+2)}}{\frac{\varepsilon^2}{9(k+1)^2}}$$

$$\leq \frac{18}{h\varepsilon^2}.$$

Thus if $\frac{18}{h\varepsilon^2} \leq \frac{1}{10}$

$$\Rightarrow Z \in \left(1 \pm \frac{\varepsilon}{3}\right)\frac{1}{k+1}$$

$$\Rightarrow \frac{1}{Z} - 1 \in (1 \pm \varepsilon)k.$$

$$\Rightarrow h \geq \frac{180}{\varepsilon^2}.$$

Exercise: To get $(1 \pm \varepsilon)$ approx with probability $\geq (1-\delta)$ via chebyshev argue that

$$h = \Omega\left(\frac{1}{\varepsilon^2} \cdot \frac{1}{\delta}\right) \text{ suffices.}$$

Space usage is h times the Space usage for single estimator.

Can we do better?

Median Trick

Can use only $h = O\left(\frac{1}{\varepsilon^2} \ln \frac{1}{\delta}\right)$ estimators. How?

Suppose Y is an estimator for k such that

$$\Pr\left[(1-\varepsilon) \le Y \le (1+\varepsilon)k\right] \ge \frac{3}{4}$$

We can obtain such an estimator using $O\left(\frac{1}{\varepsilon^2}\right)$ independent values and averaging to reduce variance and using Chebyshev as above.

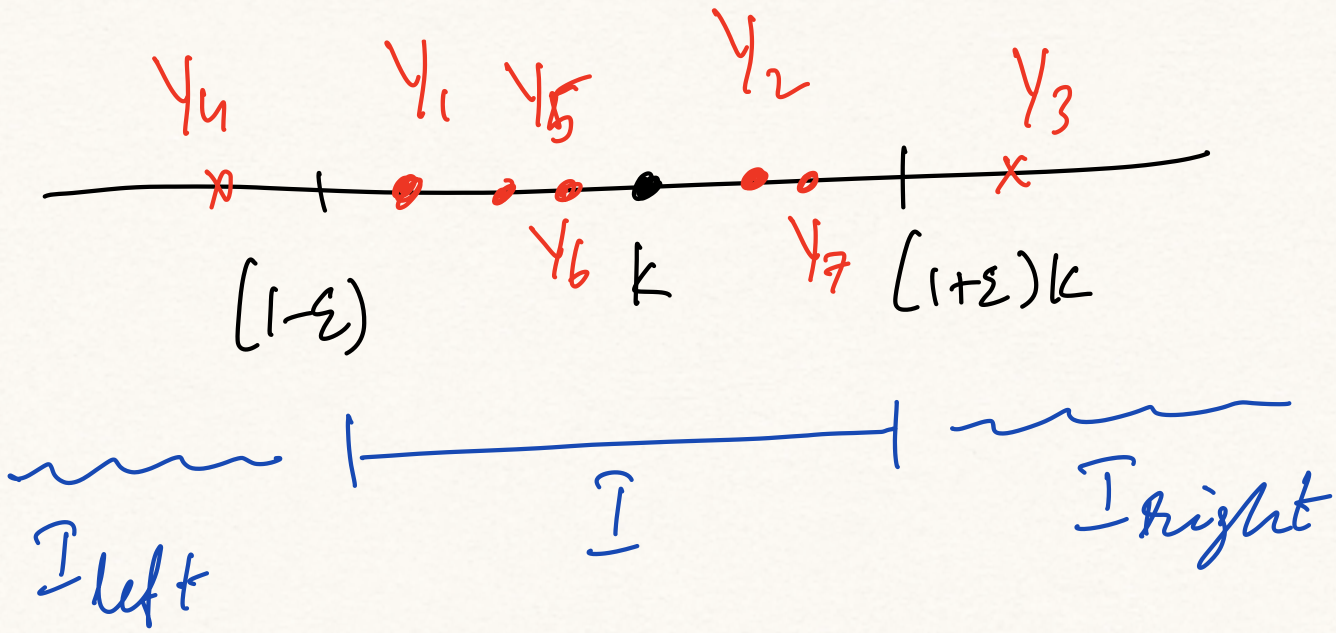
Let $Y_1, Y_2, \ldots, Y_\ell$ be independent such estimators.

<u>Lemma</u>: Let $Y = \text{median}(Y_1, \ldots, Y_\ell)$

Then $\Pr\left[(1-\varepsilon)k \le Y \le (1+\varepsilon)k\right] \ge 1-\delta$

if $l = \Omega(\log \frac{1}{\delta})$.

Proof: Use Chernoff bounds.



A_i is event that $Y_i \in I$.

$\Pr[A_i] \geq \frac{3}{4}$.

For Y to be outside I it must be that $\geq \frac{l}{2}$ fall to the right of I or $> \frac{l}{2}$ fall to the left of I.

Let R_i be indicator for $Y_i \in I_{right}$

$$\Pr[R_i] \le \frac{1}{4}.$$

$$R = \sum_{i=1}^{l} R_i \qquad E[R] \le \frac{l}{4}.$$

$$\Pr\left[R \geq \frac{l}{2}\right] \le e^{-c\frac{l}{4}}$$

$$\le \frac{\delta}{2} \quad \text{if } l = \Omega\left(\ln \frac{1}{\delta}\right)$$

Thus

$$\Pr\left[\text{median}(Y_1, \ldots, Y_l) \in I_{right}\right] \le \frac{\delta}{2}$$

By symmetric argument

$$\Pr\left[\text{median}(Y_1, \ldots, Y_l) \in I_{left}\right] \le \frac{\delta}{2}$$

$$\Rightarrow \Pr\left[\text{median}(Y_1, \ldots, Y_l) \notin I\right] \le \delta.$$

$\square$.

Final alg:

Given ε, δ.

- for $i = 1$ to l do

  $X_i \leftarrow 0$

  for $j = 1$ to h do

  Run DEHashing to obtain Z_j

  $X_i = Z_i + X_i$

  endfor

  $X_i \leftarrow \frac{1}{h} X_i$

  $Y_i = \frac{1}{X_i} - 1$

- endfor
- Output median$(Y_1, \ldots, Y_l)$.

Space usage: $\Omega\left(\frac{1}{\varepsilon^2}\log\frac{1}{\delta}\right)$ times space usage of basic estimator which is one number.

Guarantee: Output is a $(1 \pm \varepsilon)$ approx with probability $\geq (1-\delta)$.

We abstract out what we did above.

Defn: An (ε, δ) randomized scheme for a non-negative quantity α is a randomized algorithm that outputs a value X such that

$$\Pr\left[(1-\varepsilon)\alpha \leq X \leq (1+\varepsilon)X\right] \geq 1-\delta.$$

Lemma: Suppose we have a randomized algorithm that outputs $X \geq 0$ $E[X] = \alpha$ and $Var(X) \leq \beta \cdot \alpha^2$. Then one can use the algorithm $O\left(\frac{\beta}{\varepsilon^2} \ln \frac{1}{\delta}\right)$ time independently to obtain a (ε, δ)-scheme.

From ideal hashing to "actual" hashing.

In streaming we are optimizing space so also need to look at space requirement of hash function.

- Can use pairwise independent hash function.
- no real interval but can approximate $[0,1]$ by $[0, \frac{1}{n^3}, \frac{2}{n^3}, \dots, 1]$ hence $h: [n] \to [n^3]$ would be ok.
- Still need small tweak to the algorithm since we are using limited independence.

New simple algorithm due to [CMV22]

Idea:

Suppose we sample each element of the stream with some fixed probability p

Let B be the sample.

If m is the length of the stream then

$E[|B|] = pm$

$\Rightarrow \frac{|B|}{p}$ is an unbiased estimator of m.

moreover by Chernoff bounds

if $E\left[\frac{|B|}{p}\right] \geq \frac{c}{\varepsilon^2} \ln \frac{1}{\delta}$ we can

Show that $\frac{|B|}{p}$ is a $(1 \pm \varepsilon)$

approx for m with prob $\geq 1-\delta$.

But we don't want to estimate m but want to estimate F_0.

- Is there a way to sample only distinct elements?
- How do we choose p so that $\frac{|B|}{p}$ is sufficiently large but not too large for otherwise we will need to store $|B|$.

We will address first issue.

Suppose we maintain a sample B of the stream

$B \leftarrow \phi$, $m \leftarrow 0$

While (stream is not done)

$m \leftarrow m+1$

e_m current element

add e_m to B with prob p.

How do we make B sample only distinct elements?

Trick. If $e_m \in B$ remove it since we have detected a duplicate and then sample the new element with prob p.

distinct-element-sample(p)

$B \leftarrow \phi$, $m \leftarrow 0$

While (stream is not done)

$m \leftarrow m+1$

e_m current element

If $e_m \in B$, $B \leftarrow B - \{e_m\}$

Add e_m to B with prob p.

Output $\frac{|B|}{p}$ as estimate for F_0.

<u>Second issue</u>: how do we set p?

If we set p too low, $|B|$ will be too small to get accurate estimate. If we set p too high

$|B|$ will be too large.

Key idea is to "guess" F_0 and reduce p if $|B|$ gets too large.

distinct-elements-sample(n, ε)

$p \leftarrow 1$, $B \leftarrow \phi$, $m \leftarrow \phi$

$\tau \leftarrow c \dfrac{\log n}{\varepsilon^2}$ for sufficiently large c.

While (stream is not empty) do

$m \leftarrow m+1$

e_m current item

$B \leftarrow B - \{e_m\}$

With prob p:

add e_m to B

if $|B| \geq \tau$

discard each $e_i \in B$

with prob $\frac{1}{2}$.

end while

Output $\frac{|B|}{p}$

<u>Theorem</u>: The algorithm used $O\left(\frac{\log n}{\varepsilon^2}\right)$ words of memory and outputs an estimate that is within $(1 \pm \varepsilon) F_0$ with high probability.