Week 11

Streaming Algorithms and the JL Lemma

Ryan Ziegler

Outline

Background

- Probability
- Streaming and Sketching Algorithms

Streaming ℓ_2 Estimation

From Stream to Matrix

Conclusion

Section 1

Background

Subsection 1

Probability

A Probability Refresher

- (Discrete) probability distribution: given a set S assign some probability p_i to each element, so that $\sum p_i = 1$

$S = \{R, G, B\}$

$\mathbb{P}(R) = .5$

$\mathbb{P}(G) = .25$

$\mathbb{P}(B) = .25$

1

A Probability Refresher

- (Discrete) probability distribution: given a set S assign some probability p_i to each element, so that $\sum p_i = 1$
- A *random variable* X *from a distribution* D is a variable whose value is randomly chosen according to some probability distribution D. Often denoted $X \sim D$.

$$\mathbb{E}[X] = \sum p_i s_i$$

$$\mathbb{E}[X+Y] = \mathbb{E}[X] + \mathbb{E}[Y]$$

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

A Probability Refresher

- (Discrete) probability distribution: given a set S assign some probability p_i to each element, so that $\sum p_i = 1$
- A *random variable* X *from a distribution* D is a variable whose value is randomly chosen according to some probability distribution D. Often denoted $X \sim D$.
- Expected value: suppose $S \subseteq \mathbb{R}$, then $\mathbb{E}[X] = \sum p_i S_i$. Intuitively, if we picked a bunch of X following D, this is the average value we'd see.

A Probability Refresher

- (Discrete) probability distribution: given a set S assign some probability p_i to each element, so that $\sum p_i = 1$
- A *random variable* X *from a distribution* D is a variable whose value is randomly chosen according to some probability distribution D. Often denoted $X \sim D$.
- Expected value: suppose $S \subseteq \mathbb{R}$, then $\mathbb{E}[X] = \sum p_i S_i$. Intuitively, if we picked a bunch of X following D, this is the average value we'd see.
- Expectation is a linear operator: $\mathbb{E}[X+Y] = \mathbb{E}[X] + \mathbb{E}[Y]$

A Probability Refresher

- (Discrete) probability distribution: given a set S assign some probability p_i to each element, so that $\sum p_i = 1$
- A *random variable X from a distribution D* is a variable whose value is randomly chosen according to some probability distribution D. Often denoted $X \sim D$.
- Expected value: suppose $S \subseteq \mathbb{R}$, then $\mathbb{E}[X] = \sum p_i S_i$. Intuitively, if we picked a bunch of X following D, this is the average value we'd see.
- Expectation is a linear operator: $\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y]$
- Variance: $\mathrm{Var}(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2$, a low variance indicates that most of the time, when we pick X it will be close to $\mathbb{E}[X]$

A Probability Refresher

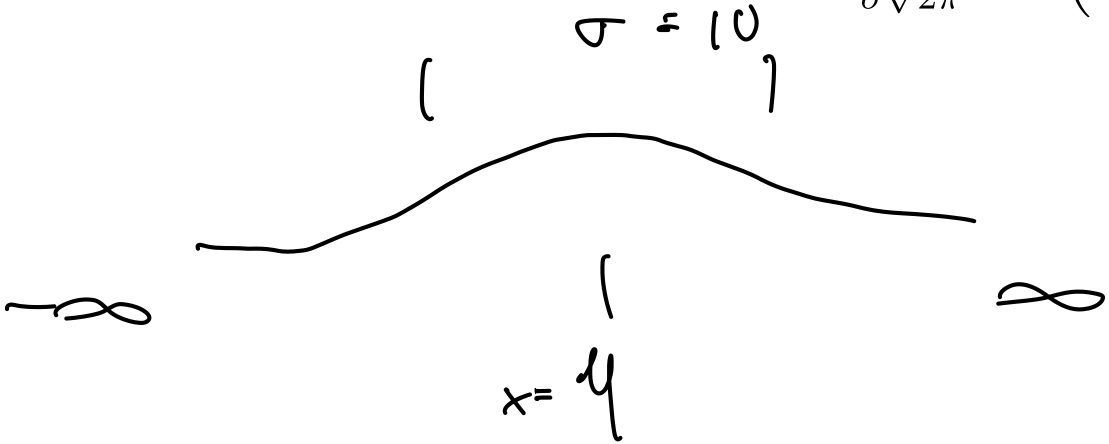
- (Discrete) probability distribution: given a set S assign some probability p_i to each element, so that $\sum p_i = 1$
- A *random variable X from a distribution D* is a variable whose value is randomly chosen according to some probability distribution D. Often denoted $X \sim D$.
- Expected value: suppose $S \subseteq \mathbb{R}$, then $\mathbb{E}[X] = \sum p_i S_i$. Intuitively, if we picked a bunch of X following D, this is the average value we'd see.
- Expectation is a linear operator: $\mathbb{E}[X+Y] = \mathbb{E}[X] + \mathbb{E}[Y]$
- Variance: $\mathrm{Var}(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2$, a low variance indicates that most of the time, when we pick X it will be close to $\mathbb{E}[X]$
  - Note that for $c \in \mathbb{R}$, $\mathrm{Var}(cX) = c^2\mathrm{Var}(X)$

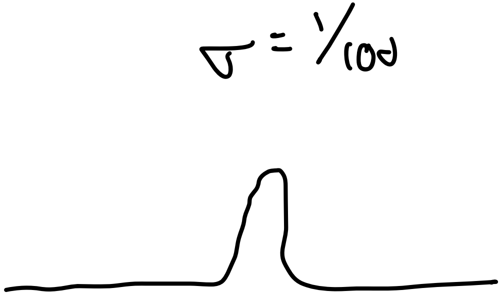
Σ

Even More Probability

$\mathcal{N}(0,1)$ $\mathbb{E}=0$ Var $=1$

- Normal distribution: $\mathcal{N}(\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)$





$X, Y \sim \mathcal{N}(\mu, \sigma)$ $X+Y$ is also Normal

Even More Probability

- Normal distribution: $\mathcal{N}(\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)$
- Normal distribution is 2-stable: for $X \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $Y \sim \mathcal{N}(\mu_2, \sigma_2^2)$, $X + Y \sim \mathcal{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$

Even More Probability

- Normal distribution: $\mathcal{N}(\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)$
- Normal distribution is 2-stable: for $X \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $Y \sim \mathcal{N}(\mu_2, \sigma_2^2)$, $X + Y \sim \mathcal{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$
- $\chi^2(k)$ distribution: Sum of k $\mathcal{N}(0,1)$ random variables, has expected value k

Even More Probability

- Normal distribution: $\mathcal{N}(\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)$
- Normal distribution is 2-stable: for $X \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $Y \sim \mathcal{N}(\mu_2, \sigma_2^2)$, $X + Y \sim \mathcal{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$
- $\chi^2(k)$ distribution: Sum of k $\mathcal{N}(0, 1)$ random variables, has expected value k
- Bernoulli distribution: If $X \sim \text{Bernoulli}(p)$, X is 1 with probability p and 0 with probability $(1 - p)$

Independence and Inequalities

$X_1, \ldots, X_n$

- A set of random variables is k-wise independent iff for any k variables in the set, $f(x_1, \ldots, x_k) = f(x_1) \cdots f(x_k)$

Independence and Inequalities

- A set of random variables is k-wise independent iff for any k variables in the set, $f(x_1, \ldots, x_k) = f(x_1) \cdots f(x_k)$
- For k-wise independent random variables, $\mathbb{E}\left[\prod_{i=1}^{k} X_i\right] = \prod_{i=1}^{k} \mathbb{E}[x_i]$

Independence and Inequalities

- A set of random variables is k-wise independent iff for any k variables in the set, $f(x_1, \ldots, x_k) = f(x_1) \cdots f(x_k)$
- For k-wise independent random variables, $\mathbb{E}\left[\prod_{i=1}^{k} X_i\right] = \prod_{i=1}^{k} \mathbb{E}[x_i]$
    - Important: k-wise independence implies $(k-1)$-wise independence

Independence and Inequalities

- A set of random variables is k-wise independent iff for any k variables in the set, $f(x_1, \ldots, x_k) = f(x_1) \cdots f(x_k)$
- For k-wise independent random variables, $\mathbb{E}\left[\prod_{i=1}^{k} X_i\right] = \prod_{i=1}^{k} \mathbb{E}[x_i]$
  - Important: k-wise independence implies $(k-1)$-wise independence
- Chebyshev's inequality: $\mathrm{P}(|X - \mathbb{E}[X]| \geq k\sigma) \leq \frac{1}{k^2}$

↳ $\sqrt{\text{Var}}$

Independence and Inequalities

- A set of random variables is k-wise independent iff for any k variables in the set, $f(x_1, \ldots, x_k) = f(x_1) \cdots f(x_k)$
- For k-wise independent random variables, $\mathbb{E}\left[\prod_{i=1}^{k} X_i\right] = \prod_{i=1}^{k} \mathbb{E}[x_i]$
  - Important: k-wise independence implies $(k-1)$-wise independence
- Chebyshev's inequality: $\mathrm{P}(|X - \mathbb{E}[X]| \geq k\sigma) \leq \frac{1}{k^2}$
- Chernoff bound: Let X be sum of h fully independent Bernoulli RVs, and $\delta \geq 1$. $\mathrm{P}(X > (1+\delta)\mathbb{E}[X]) \leq e^{-\delta^2 \mu/3}$

$Y_1, \ldots, Y_n \sim \text{Bernoulli}(p)$ $\quad X = \sum Y_i$

$\mathbb{E}[X] = np$

Subsection 2

Streaming and Sketching Algorithms

Intro to Streaming Algorithms

- Streaming model: your algorithm receives inputs one-by-one, and you don't know how many inputs you'll receive. Too many inputs to store them all and compute later

too many elements to store in memory!

⇒ sublinear space

ex: you receive a stream of Youtube video views. want the k most watched videos today

keep a data structure with $O(k)$ space, and update that when you see a video

Misra-Gries!

Σ

Intro to Streaming Algorithms

- Streaming model: your algorithm receives inputs one-by-one, and you don't know how many inputs you'll receive. Too many inputs to store them all and compute later
- (*) Example: suppose you want to calculate the k most watched YouTube videos today. It takes too much space to store all the YouTube videos and associated view counters, so you want an algorithm that does the following: upon recieving a YouTube video ID, update some data structure and continue without storing anything on disk. At the end of the day, this data structure should tell you the k most viewed videos.

Intro to Streaming Algorithms

- Streaming model: your algorithm receives inputs one-by-one, and you don't know how many inputs you'll receive. Too many inputs to store them all and compute later
- (*) Example: suppose you want to calculate the k most watched YouTube videos today. It takes too much space to store all the YouTube videos and associated view counters, so you want an algorithm that does the following: upon recieving a YouTube video ID, update some data structure and continue without storing anything on disk. At the end of the day, this data structure should tell you the k most viewed videos.
- (*) The above is possible to do *exactly* with only $O(k)$ space, but this is rare. Most streaming algorithms will only output approximates that are good with some probability

output a random variable Z

$\mathbb{E}[Z] = g(\sigma)$

↳ σ is our stream

generally, $Var(Z) \leq g(\sigma)$

$Z^* = \frac{1}{n} \sum Z_i$ where Z_i is an IID copy of alg.

$Var(Z^*) = \frac{1}{n} Var(Z)$

$\mathbb{P}\{|Z^* - g(\sigma)| \geq \varepsilon g(\sigma)\} \leq \frac{1}{4}$ ⟹ Chebyshev

Q: how big should n be?

$n = O(\frac{1}{\varepsilon^2})$ if $\varepsilon \leq .001$, $n \leq 1000$

next goal: we want to output some Z' so that

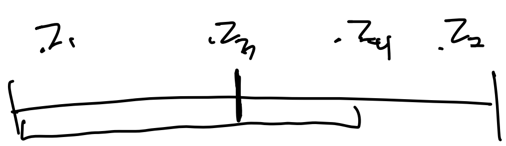
$\mathbb{P}\{|Z' - g(\sigma)| \geq \varepsilon g(\sigma)\} \leq \delta$

$Z_1^*, \ldots, Z_k^*$ that fail w/ prob. 1/4

↓

$X_1, \ldots, X_k \sim$ Bernolli(1/4)

$X_i = 1$ if Z_i is "bad", 0 otherwise



median is bad iff $n/2$ bad values

$N = \sum X_i$ $\mathbb{P}\{|Z' - g(\sigma)| \geq \varepsilon g(\sigma)\} = \mathbb{P}\{N \geq \frac{n}{2}\}$

$= \mathbb{P}\{N \geq (1+1)\frac{n}{4}\}$

$\leq \exp(-k/12) \Rightarrow k = O(\log 1/\delta)$

$O(\frac{\ln(1/\delta)}{\varepsilon^2})$ parallel copies!

A Template for Sketching Algorithms

- First, output a random variable Z such that $\mathbb{E}[Z] = g(\sigma)$ where $g(\sigma)$ is the function we're estimating for the stream σ
- Usually Z will have high variance, typically $\text{Var}(Z) \leq g(\sigma)$
- How to reduce variance? Run the streaming algorithm h times in parallel, and let $Z^* = \frac{1}{h}\sum Z_i$

$$\text{Var}(Z^*) = \frac{1}{h}\text{Var}(Z_1) \text{ and } \mathbb{E}[Z^*] = \mathbb{E}[Z_1]$$

- (*) By Chebyshev's inequality,

$$\text{P}\left(|Z^* - g(\sigma)| > \epsilon g(\sigma)\right) \leq \frac{\epsilon^2}{h}$$

- (*) So, pick $h = \frac{4}{\epsilon^2}$ for constant failure probability of $\frac{1}{4}$

The Median Trick

- Next goal: $|Z^* - g(\sigma)| > \epsilon g(\sigma)$ with some small probability δ

The Median Trick

- Next goal: $|Z^* - g(\sigma)| > \epsilon g(\sigma)$ with some small probability δ
- Naive approach: do Chebyshev's again. Requires $O\left(\frac{1}{\delta\epsilon^2}\right)$ parallel copies. We want to do better

The Median Trick

- Next goal: $|Z^* - g(\sigma)| > \epsilon g(\sigma)$ with some small probability δ
- Naive approach: do Chebyshev's again. Requires $O\left(\frac{1}{\delta\epsilon^2}\right)$ parallel copies. We want to do better
- Consider parallel copies $Z_1^*, \ldots, Z_k^*$ that each fail with probability $1/4$

The Median Trick

- Next goal: $|Z^* - g(\sigma)| > \epsilon g(\sigma)$ with some small probability δ
- Naive approach: do Chebyshev's again. Requires $O\left(\frac{1}{\delta\epsilon^2}\right)$ parallel copies. We want to do better
- Consider parallel copies $Z_1^*, \ldots, Z_k^*$ that each fail with probability $1/4$
- Our intuition tells us the median of these estimators should be "good" but how good?

The Median Trick

- Next goal: $|Z^* - g(\sigma)| > \epsilon g(\sigma)$ with some small probability δ
- Naive approach: do Chebyshev's again. Requires $O\left(\frac{1}{\delta\epsilon^2}\right)$ parallel copies. We want to do better
- Consider parallel copies $Z_1^*, \ldots, Z_k^*$ that each fail with probability 1/4
- Our intuition tells us the median of these estimators should be "good" but how good?
- (*) Let $X_i = 1$ iff the ith parallel copy fails, so then $X_i \sim \text{Bernoulli}(1/4)$

The Median Trick

- Next goal: $|Z^* - g(\sigma)| > \epsilon g(\sigma)$ with some small probability δ
- Naive approach: do Chebyshev's again. Requires $O\left(\frac{1}{\delta\epsilon^2}\right)$ parallel copies. We want to do better
- Consider parallel copies $Z_1^*, \ldots, Z_k^*$ that each fail with probability 1/4
- Our intuition tells us the median of these estimators should be "good" but how good?
- (*) Let $X_i = 1$ iff the ith parallel copy fails, so then $X_i \sim \text{Bernoulli}(1/4)$
- (*) Define $X = \sum X_i$, so then $\mathbb{E}[X] = \frac{k}{4}$

The Median Trick

- Next goal: $|Z^* - g(\sigma)| > \epsilon g(\sigma)$ with some small probability δ
- Naive approach: do Chebyshev's again. Requires $O\left(\frac{1}{\delta\epsilon^2}\right)$ parallel copies. We want to do better
- Consider parallel copies $Z_1^*, \ldots, Z_k^*$ that each fail with probability $1/4$
- Our intuition tells us the median of these estimators should be "good" but how good?
- (*) Let $X_i = 1$ iff the ith parallel copy fails, so then $X_i \sim \text{Bernoulli}(1/4)$
- (*) Define $X = \sum X_i$, so then $\mathbb{E}[X] = \frac{k}{4}$
- (*) By Chernoff bound,

$$\mathrm{P}\left(X \geq (1+1)\frac{k}{4}\right) \leq e^{-k/12}$$

The Median Trick

- Next goal: $|Z^* - g(\sigma)| > \epsilon g(\sigma)$ with some small probability δ
- Naive approach: do Chebyshev's again. Requires $O\left(\frac{1}{\delta\epsilon^2}\right)$ parallel copies. We want to do better
- Consider parallel copies $Z_1^*, \ldots, Z_k^*$ that each fail with probability 1/4
- Our intuition tells us the median of these estimators should be “good" but how good?
- (*) Let $X_i = 1$ iff the ith parallel copy fails, so then $X_i \sim \text{Bernoulli}(1/4)$
- (*) Define $X = \sum X_i$, so then $\mathbb{E}[X] = \frac{k}{4}$
- (*) By Chernoff bound,

$$\mathrm{P}\left(X \geq (1+1)\frac{k}{4}\right) \leq e^{-k/12}$$

- (*) So, pick $k = O(\log(1/\delta))$. Only running $O\left(\frac{\log\left(\frac{1}{\delta}\right)}{\epsilon^2}\right)$

Section 2

Streaming ℓ_2 Estimation

Frequency Moment Estimation

- Problem: we receive a stream σ of values $e_1, \cdots \in \mathbb{Z}$ where $1 \leq e_i \leq n$ for some n we know apriori

Frequency Moment Estimation

- Problem: we receive a stream σ of values $e_1, \cdots \in \mathbb{Z}$ where $1 \le e_i \le n$ for some n we know apriori
- Define the frequency vector to be $f(\sigma) = (f_1, \ldots, f_n)$ where f_i is the number of times we've seen i

$$\sigma = \{1, 1, 5, 7\}$$

$$f = (2, 0, 0, 0, 1, 0, 1)$$

Frequency Moment Estimation

- Problem: we receive a stream σ of values $e_1, \dots \in \mathbb{Z}$ where $1 \leq e_i \leq n$ for some n we know apriori
- Define the frequency vector to be $f(\sigma) = (f_1, \dots, f_n)$ where f_i is the number of times we've seen i
- Goal: estimate $||f(\sigma)||_2^2$ with only $O(\text{polylog}(n))$ space

Frequency Moment Estimation

- Problem: we receive a stream σ of values $e_1, \cdots \in \mathbb{Z}$ where $1 \leq e_i \leq n$ for some n we know apriori
- Define the frequency vector to be $f(\sigma) = (f_1, \ldots, f_n)$ where f_i is the number of times we've seen i
- Goal: estimate $||f(\sigma)||_2^2$ with only $O(\text{polylog}(n))$ space
- Recall the definition of L_2 norm:

$$||f(\sigma)||_2^2 = \sum_{i=1}^{n} f_i^2$$

AMS F2 Estimation

- Intuition: keep a single variable Z so that we can output Z^2 as our estimate of $||f(\sigma)||_2^2$

$$\mathbb{E}[Z^4] \leq \|f(\sigma)\|_2^2$$

$$Z = \sum f_i Y_i$$

$$\mathbb{E}[Z] = \sum f_i \mathbb{E}[Y_i]$$

$$\mathbb{E}[Z^2] = \underbrace{\sum f_i^2 \mathbb{E}[Y_i]^2} + 2 \sum_{i \neq j} \mathbb{E}[Y_i Y_j] f_i f_j$$

$$\mathbb{E}[Y_i^2] = 1 \qquad \mathbb{E}[Y_i Y_j] = 0 \qquad \mathbb{E}[Z^2] = \|f(\sigma)\|_2^2$$

let Y_i be pw ind $Y_i = 1$ w/ prob $.5$

$Y_i = -1$ w/ prob $.5$

AMS F2 Estimation

- Intuition: keep a single variable Z so that we can output Z^2 as our estimate of $||f(\sigma)||_2^2$
- (*) Idea: create some random variable Y_i for each index so that $\mathbb{E}[Z^2] = ||f(\sigma)||_2^2$. In particular, $Z = \sum Y_i f_i$

$$\mathbb{E}[Z^2] = \sum f_i^2 Y_i^2 + 2 \sum_{i \neq j} f_i f_j Y_i Y_j$$

- (*) We need Y_i to be pairwise independent and satisfy $\mathbb{E}[Y_i Y_j] = 0$ and $\mathbb{E}[Y_i^2] = 1$

AMS F2 Estimation

- Intuition: keep a single variable Z so that we can output Z^2 as our estimate of $||f(\sigma)||_2^2$
- (*) Idea: create some random variable Y_i for each index so that $\mathbb{E}[Z^2] = ||f(\sigma)||_2^2$. In particular, $Z = \sum Y_i f_i$

$$\mathbb{E}[Z^2] = \sum f_i^2 Y_i^2 + 2 \sum_{i \neq j} f_i f_j Y_i Y_j$$

- (*) We need Y_i to be pairwise independent and satisfy $\mathbb{E}[Y_i Y_j] = 0$ and $\mathbb{E}[Y_i^2] = 1$
- (*) Solution: $Y_i = 1$ with probability $\frac{1}{2}$ and $Y_i = -1$ with probability $\frac{1}{2}$

AMS F2 Estimation Continued

- Creating $O(n)$ random variables takes up too much space!
- Solution: $O(1)$-wise independent hash family of functions $[n] \to \{-1, 1\}$ can be stored in $O(\text{polylog}(n))$ space

$Z = \sum f_i h(i)$

when we see $e \in [n]$

$z \mathrel{+}= h(e)$

AMS F2 Estimation Continued

- Creating $O(n)$ random variables takes up too much space!
- Solution: $O(1)$-wise independent hash family of functions $[n] \to \{-1, 1\}$ can be stored in $O(\text{polylog}(n))$ space
- (*) Replace each Y_i with $h(i)$, and the analysis is the exact same

AMS F2 Estimation Continued

- Creating $O(n)$ random variables takes up too much space!
- Solution: $O(1)$-wise independent hash family of functions $[n] \to \{-1, 1\}$ can be stored in $O(\text{polylog}(n))$ space
- (*) Replace each Y_i with $h(i)$, and the analysis is the exact same
- (*) Similar analysis shows $\mathbb{E}[Z^4] \leq 2||f(\sigma)||_2^2$, so we can apply average and median idea from before

AMS F2 Estimation Continued

- Creating $O(n)$ random variables takes up too much space!
- Solution: $O(1)$-wise independent hash family of functions $[n] \to \{-1, 1\}$ can be stored in $O(\text{polylog}(n))$ space
- (*) Replace each Y_i with $h(i)$, and the analysis is the exact same
- (*) Similar analysis shows $\mathbb{E}[Z^4] \leq 2||f(\sigma)||_2^2$, so we can apply average and median idea from before

```
def ams_f2:
    let h be a hash function from hash family H
    let z = 0
    while i is an item from stream
        z = z + h(i)
    output z
```

$O\left(\frac{n^{1/8} \log n}{\varepsilon^2}\right)$

Extending F2 Estimation

- Note that we never used the fact that f_i was positive or integral
- Richer model: receive a stream of updates of the form (i, Δ_i) representing a change to the ith coordinate of our vector

Extending F2 Estimation

- Note that we never used the fact that f_i was positive or integral
- Richer model: receive a stream of updates of the form (i, Δ_i) representing a change to the ith coordinate of our vector

```
def l2_estimate:
    let h be a hash function from hash family H
    let z = 0
    while (i,d) is an item from stream
        z = z + h(i)d
    output z²
```

alg is a func C let σ_1 and σ_2 be streams

$$C(\sigma_1 + \sigma_2) = C(\sigma_1) + C(\sigma_2)$$

$$
O\left(\frac{\ln 1/\delta}{\varepsilon^2}\right)\left\{
\overbrace{\begin{bmatrix}
h_1(1) & h_1(2) & \cdots & \cdots & h_1(n) \\
 & & \vdots & & \\
h_k(1) & \cdots & & & h_k(n)
\end{bmatrix}}^{n}
\right.
\begin{bmatrix} \\ f \\ \\ \end{bmatrix}
=
\begin{bmatrix} \\ \\ \\ \end{bmatrix}
O\left(\frac{\ln 1/\delta}{\varepsilon^2}\right)
$$

Section 3

From Stream to Matrix

Linear Sketching

- What we just created is a linear sketch: call our algorithm C. We can show that $C(\sigma_1 + \sigma_2) = C(\sigma_1) + C(\sigma_2)$, since each iteration we add to Z
- (*) Geometric interpretation: our algorithm is an $\frac{\log(1/\delta)\log n}{\epsilon^2} \times n$ matrix M of $\{-1, 1\}$ values, each row is a parallel copy of the streaming algorithm

Linear Sketching

- What we just created is a linear sketch: call our algorithm C. We can show that $C(\sigma_1 + \sigma_2) = C(\sigma_1) + C(\sigma_2)$, since each iteration we add to Z
- (*) Geometric interpretation: our algorithm is an $\frac{\log(1/\delta)\log n}{\epsilon^2} \times n$ matrix M of $\{-1, 1\}$ values, each row is a parallel copy of the streaming algorithm
- (*) Now we have $Mx = y$ where y is a vector whose length is similar to that of x but is in lower dimension

Linear Sketching

- What we just created is a linear sketch: call our algorithm C. We can show that $C(\sigma_1 + \sigma_2) = C(\sigma_1) + C(\sigma_2)$, since each iteration we add to Z
- (*) Geometric interpretation: our algorithm is an $\frac{\log(1/\delta)\log n}{\epsilon^2} \times n$ matrix M of $\{-1, 1\}$ values, each row is a parallel copy of the streaming algorithm
- (*) Now we have $Mx = y$ where y is a vector whose length is similar to that of x but is in lower dimension
- (*) Next goal: generalize this idea so that we can reduce the dimension of a *set* of vectors while preserving pairwise distances

Linear Sketching

- What we just created is a linear sketch: call our algorithm C. We can show that $C(\sigma_1 + \sigma_2) = C(\sigma_1) + C(\sigma_2)$, since each iteration we add to Z
- (*) Geometric interpretation: our algorithm is an $\frac{\log(1/\delta)\log n}{\epsilon^2} \times n$ matrix M of $\{-1, 1\}$ values, each row is a parallel copy of the streaming algorithm
- (*) Now we have $Mx = y$ where y is a vector whose length is similar to that of x but is in lower dimension
- (*) Next goal: generalize this idea so that we can reduce the dimension of a *set* of vectors while preserving pairwise distances
- (*) Useful in real-world applications such as nearest neighbors, ML, etc

The JL Lemma

$\mathbb{E}[X] = 0 \quad \mathbb{E}[X^2] = 1$

- Let M be an $k \times n$ matrix where each entry is chosen independently from $\mathcal{N}(0,1)$
- Claim: for $k = \Omega\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$, we have that with probability $1 - \delta$, $||\frac{1}{\sqrt{k}} Mx||_2 = (1 \pm \epsilon)||x||_2$ for fixed $x \in \mathbb{R}^n$

Let S be a set of n vectors.

If $\delta = 1/n^2$, JL matrix will preserve pairwise distances

The JL Lemma

- Let M be an $k \times n$ matrix where each entry is chosen independently from $\mathcal{N}(0,1)$
- Claim: for $k = \Omega\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$, we have that with probability $1-\delta$, $||\frac{1}{\sqrt{k}}Mx||_2 = (1 \pm \epsilon)||x||_2$ for fixed $x \in \mathbb{R}^n$
- Immediate corollary: Let S be a set of k vectors in $\mathbb{R}^n$, we can preserve pairwise distances with high probability by picking $k = \Omega\left(\frac{\log n}{\epsilon^2}\right)$

The JL Lemma

- Let M be an $k \times n$ matrix where each entry is chosen independently from $\mathcal{N}(0, 1)$
- Claim: for $k = \Omega\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$, we have that with probability $1 - \delta$, $||\frac{1}{\sqrt{k}}Mx||_2 = (1 \pm \epsilon)||x||_2$ for fixed $x \in \mathbb{R}^n$
- Immediate corollary: Let S be a set of k vectors in $\mathbb{R}^n$, we can preserve pairwise distances with high probability by picking $k = \Omega\left(\frac{\log n}{\epsilon^2}\right)$

JL Lemma: Idea of Proof

- Fix some vector x (wlog, let $||x|| = 1$) and use 2-stability of Normal distribution

JL Lemma: Idea of Proof

- Fix some vector x (wlog, let $||x|| = 1$) and use 2-stability of Normal distribution
- (*) Let $y = Mx$, so then $y_i = \sum_{j=1}^{k} M_{ij} x_i$

JL Lemma: Idea of Proof

- Fix some vector x (wlog, let $||x|| = 1$) and use 2-stability of Normal distribution
- (*) Let $y = Mx$, so then $y_i = \sum_{j=1}^{k} M_{ij} x_i$
- (*) y is a Normal vector in $\mathbb{R}^k$, and each y_i is $\mathcal{N}(0, 1)$ (variance because $\sum x_i^2 = 1$)

JL Lemma: Idea of Proof

- Fix some vector x (wlog, let $||x|| = 1$) and use 2-stability of Normal distribution
- (*) Let $y = Mx$, so then $y_i = \sum_{j=1}^{k} M_{ij} x_i$
- (*) y is a Normal vector in $\mathbb{R}^k$, and each y_i is $\mathcal{N}(0,1)$ (variance because $\sum x_i^2 = 1$)
- (*) Let $\alpha = \sum y_i^2$, so then $\alpha \sim \chi^2(k)$

JL Lemma: Idea of Proof

- Fix some vector x (wlog, let $||x|| = 1$) and use 2-stability of Normal distribution
- (*) Let $y = Mx$, so then $y_i = \sum_{j=1}^{k} M_{ij} x_i$
- (*) y is a Normal vector in $\mathbb{R}^k$, and each y_i is $\mathcal{N}(0,1)$ (variance because $\sum x_i^2 = 1$)
- (*) Let $\alpha = \sum y_i^2$, so then $\alpha \sim \chi^2(k)$
- (*) Thus $\mathrm{P}((1-\epsilon)^2 k \leq \alpha \leq (1+\epsilon)^2 k) \geq 1 - 2e^{O(1)\epsilon^2 k}$

JL Lemma: Idea of Proof

- Fix some vector x (wlog, let $||x|| = 1$) and use 2-stability of Normal distribution
- (*) Let $y = Mx$, so then $y_i = \sum_{j=1}^{k} M_{ij} x_i$
- (*) y is a Normal vector in $\mathbb{R}^k$, and each y_i is $\mathcal{N}(0, 1)$ (variance because $\sum x_i^2 = 1$)
- (*) Let $\alpha = \sum y_i^2$, so then $\alpha \sim \chi^2(k)$
- (*) Thus $\mathrm{P}((1-\epsilon)^2 k \leq \alpha \leq (1+\epsilon)^2 k) \geq 1 - 2e^{O(1)\epsilon^2 k}$
- (*) Picking $k = \Omega\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$ gets us the probability we want

Section 4

Conclusion

JL Lemma: Intuition and Application

- Why does projecting to a random subspace work? A large enough random subspace means errors induced by “bad vectors" (i.e. those orthogonal to many rows in the matrix) have extremely low probability of ocurring

Σ

JL Lemma: Intuition and Application

- Why does projecting to a random subspace work? A large enough random subspace means errors induced by “bad vectors" (i.e. those orthogonal to many rows in the matrix) have extremely low probability of ocurring
- Useful for tasks such as clustering/ML: things closer together/more similar in low dimension will be close in high dimension, so can reduce dimension and speed up clustering

JL Lemma: Intuition and Application

- Why does projecting to a random subspace work? A large enough random subspace means errors induced by “bad vectors" (i.e. those orthogonal to many rows in the matrix) have extremely low probability of ocurring
- Useful for tasks such as clustering/ML: things closer together/more similar in low dimension will be close in high dimension, so can reduce dimension and speed up clustering
- Coreset generation: Many hard geometric problems have fast approximate solutions via coreset technique, which generates a set S' from input S so that running an exact algorithm on S' generates a high accuracy approximation for that algorithm on S. JL technique can be used in generating coresets

JL Lemma: Intuition and Application

- Why does projecting to a random subspace work? A large enough random subspace means errors induced by “bad vectors" (i.e. those orthogonal to many rows in the matrix) have extremely low probability of ocurring
- Useful for tasks such as clustering/ML: things closer together/more similar in low dimension will be close in high dimension, so can reduce dimension and speed up clustering
- Coreset generation: Many hard geometric problems have fast approximate solutions via coreset technique, which generates a set S' from input S so that running an exact algorithm on S' generates a high accuracy approximation for that algorithm on S. JL technique can be used in generating coresets
- Key advantage of JL is that it is *oblivious* to data

One more thing...

- JL Lemma extends to preserving vector distances in *entire subspaces* of $\mathbb{R}^n$!

One more thing...

- JL Lemma extends to preserving vector distances in *entire subspaces* of $\mathbb{R}^n$!
- Let E be a linear subspace of dimension d

One more thing...

- JL Lemma extends to preserving vector distances in *entire subspaces* of $\mathbb{R}^n$!
- Let E be a linear subspace of dimension d
- Can preserve distances between vectors in E with $k = \Omega\left(\frac{d \log(1/\delta)}{\epsilon^2}\right)$

One more thing...

- JL Lemma extends to preserving vector distances in *entire subspaces* of $\mathbb{R}^n$!
- Let E be a linear subspace of dimension d
- Can preserve distances between vectors in E with $k = \Omega\left(\frac{d \log(1/\delta)}{\epsilon^2}\right)$
- Works for all vectors in E, even though there are infinitely many!

One more thing...

- JL Lemma extends to preserving vector distances in *entire subspaces* of $\mathbb{R}^n$!
- Let E be a linear subspace of dimension d
- Can preserve distances between vectors in E with $k = \Omega\left(\frac{d \log(1/\delta)}{\epsilon^2}\right)$
- Works for all vectors in E, even though there are infinitely many!
- Poof: consider partitioning the d dimensional unit ball into small hypercubes with small side length. Show that preserving lengths of vectors to these hypercubes is sufficient to preserve lengths of all vectors.

$\sum g_i(F_i)$ $O(\sqrt{n}$ $O(pd \times \log)$

$0 < k \leq 2$