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 $\sum p_i = 1$

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 $X \sim D$.

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: $\text{Var}(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2$, a low variance indicates that most of the time, when we pick X it will be close to $\mathbb{E}[X]$
  - Note that for $c \in \mathbb{R}$, $\text{Var}(cX) = c^2\text{Var}(X)$

Even More Probability

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

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

- 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, $f(x_1, \dots, x_k) = f(x_1) \cdots f(x_k)$
- 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: $\mathrm{P}(|X - \mathbb{E}[X]| \geq k\sigma) \leq \frac{1}{k^2}$

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 $\delta \geq 1$. $\mathrm{P}(X > (1+\delta)\mathbb{E}[X]) \leq e^{-\delta^2 \mu/3}$

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

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 $\mathrm{Var}(Z) \leq g(\sigma)$
- How to reduce variance? Run the streaming algorithm h times in parallel, and let $Z^* = \frac{1}{h} \sum Z_i$

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?

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 $e_1, \dots \in \mathbb{Z}$ 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

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

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 $||f(\sigma)||_2^2$

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

```
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
```

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 z
```

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

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 $x \in \mathbb{R}^n$

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

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

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.