Streaming Algorithms and the JL Lemma

Ryan

April 27, 2023

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

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, $f(x_1, \ldots, x_k) = f(x_1) \cdots f(x_k)$

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

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

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

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

- By Chebyshev's inequality,

$$\mathrm{P}\left(|Z^* - g(\sigma)| > c\epsilon g(\sigma)\right) \leq \frac{1}{2c\epsilon^2}$$

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

The Median Trick

- Next goal: $|Z^* - g(\sigma)| > \epsilon g(\sigma)$ with some small probability δ
- 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 $O\left(\frac{\log\left(\frac{1}{\delta}\right)}{\epsilon^2}\right)$ independent copies of our algorithm!

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

AMS F2 Estimation

- Intuition: keep a single variable Z so that we can output Z^2 as our estimate of $||f(\sigma)||_2^2$
- Recall the definition of L_2 norm:

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

- 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
- 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
```

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
```

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

- 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
- 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

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 $\mathbb{R}^n$!
- 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.