CS 229r: Algorithms for Big Data Fall 2015

Lecture 1 — September 3, 2015

Prof. Jelani Nelson *Scribes: Zhengyu Wang*

1 Course Information

- Professor: Jelani Nelson
- TF: Jarosław Błasiok

2 Topic Overview

1. Sketching/Streaming
   - "*Sketch*" $C(X)$ with respect to some function f is a *compression* of data X. It allows us computing $f(X)$ (with approximation) given access only to $C(X)$.
   - Sometimes f has 2 arguments. For data X and Y, we want to compute $f(X, Y)$ given $C(X), C(Y)$.
   - Motivation: maybe you have some input data and I have some input data, and we want to compute some similarity measure of these two databases across the data items. One way is that I can just send you the database, and you can compute locally the similarity measure, and vise versa. But image these are really big data sets, and I don't want to send the entire data across the wire, rather what I will do is to compute the sketch of my data, and then send the sketch to you, which is something very small after compression. Now the sketch $C(X)$ is much smaller than X, and given the sketch you can compute the function.
   - Trivial example: image you have a batch of numbers, and I also have a batch of numbers. We want to compute their sum. The sketch I can do is just locally sum all my input data, and send you the sum.
   - *Streaming*: we want to maintain a sketch $C(X)$ on the fly as x is updated. In previous example, if numbers come on the fly, I can keep a running sum, which is a streaming algorithm. The streaming setting appears in a lot of places, for example, your router can monitor online traffic. You can sketch the number of traffic to find the traffic pattern.
2. Dimensionality Reduction
   - Input data is high-dimensional. Dimensionality reduction transforms high-dimensional data into lower-dimensional version, such that for the computational problem you are considering, once you solve the problem on the lower-dimensional transformed data, you can get approximate solution on original data. Since the data is in low dimension, your algorithm can run faster.
   - Application: *speed up* clustering, nearest neighbor, etc.

3. Large-scale Machine Learning

   - For example, regression problems: we collect data points $\{(z_i, b_i)|i = 1, \ldots, n\}$ such that $b_i = f(z_i) + \text{noise}$. We want to recover $\tilde{f}$ "close" to f.
   - Linear regression: $f(z) = \langle x, z \rangle$, where x is the parameter that we want to recover. If the noise is Gaussian, the popular (and optimal to some sense) estimator we use is Least Squares

   $$x^{LS} = \arg\min \|Zx - b\|_2^2 = (Z^T Z)^{-1} Zb, \tag{1}$$

   where $b = (b_1, \ldots, b_n)^T$ and $Z = (z_1, \ldots, z_n)^T$. If Z is big, matrix multiplication can be very expensive. In this course, we will study techniques that allow us to solve least squares much faster than just computing the closed form $(Z^T Z)^{-1} Zb$.
   - Other regression problems: PCA (Principal Component Analysis), matrix completion. For example, matrix completion for Netflix problem: you are given a big product-customer matrix of customer ratings of certain products. The matrix is very sparse because not every user is going to rate everything. Based on limited information, you want to guess the rest of the matrix to do product suggestions.

4. Compressed Sensing

   - Motivation: *compress* / cheaply *acquir*e high dimensional signal (using linear measurement)
   - For example, images are very high dimensional vectors. If the dimension of an image is thousands by thousands, it means that the image has millions of pixels. If we write the image in standard basis as pixels, it is likely that the pixels are not sparse (by sparse we mean almost zero), because just image that if we take a photo in a dark room, most of the pixels have some intensity. But there are some basis called *wavelet basis*, pictures are usually very sparse on that basis. Once something is sparse, you can compress it.
   - JPEG (image compression).
   - MRI (faster acquisition of the signal means less time in machine).

5. External Memory Model

   - Motivation: measure disk I/O's instead of number of instructions (because random seeks are very expensive).
   - Model: we have infinite *disk* divided into *blocks* of size b bits, and *memory* of size M divided into *pages* of size b bits. If the data we want to read or the location we want to write is in the memory, we can just simply do it for free; if the location we want to access is not in the memory, we cost 1 unit time to load the block from the disk into the memory, and vise versa. We want to minimize the time we go to the disk.
   - B trees are designed for this model.

6. Other Models (if time permitting)

   - For example, map reduce.

3 Approximate Counting Problem

In the following, we discuss the problem appearing in the first streaming paper [1].

Problem. There are a batch of events happen. We want to count the number of events while minimizing the *space* we use.

Note that we have a trivial solution - maintaining a counter - which takes $\log n$ bits where n is the number of events. On the other hand, by Pigeonhole Principle, we cannot beat $\log n$ bits if we want to count exactly.

For *approximate* counting problem, we want to output $\tilde{n}$ such that

$$\mathbb{P}(|\tilde{n} - n| > \varepsilon n) < \delta, \tag{2}$$

where let's say $\varepsilon = 1/3$ and $\delta = 1\%$.

First of all, we can say that if we want to design a deterministic algorithm for approximate counting problem, we cannot beat against $\log \log n$ bits, because similar to previous lower bound argument, there are $\log n$ different bands (of different powers of 2), and it takes $\log \log n$ bits to distinguish them. Therefore, we maybe hope for $O(\log \log n)$ bits algorithm. Actually, the following Morris Algorithm can give us the desired bound:

1. Initialize $X \leftarrow 0$.
2. For each event, increment X with probability $\frac{1}{2^X}$.
3. Output $\tilde{n} = 2^X - 1$.

Intuitively, we have $X \approx \lg n$ where $\lg x = \log_2(2+x)$. Before giving rigorous analysis (in Section 5) for the algorithm, we first give a probability review.

4 Probability Review

We are mainly discussing discrete random variables. Let random variable X takes values in S. Expectation of X is defined to be $\mathbb{E}\, X = \sum_{j \in S} j \cdot \mathbb{P}(X = j)$.

Lemma 1 (Linearity of expectation)**.**

$$\mathbb{E}(X + Y) = \mathbb{E}\, X + \mathbb{E}\, Y \tag{3}$$

Lemma 2 (Markov)**.**

$$X \textit{ is a non-negative random variable} \Rightarrow \forall \lambda > 0, \mathbb{P}(X > \lambda) < \frac{\mathbb{E}\, X}{\lambda} \tag{4}$$

Lemma 3 (Chebyshev)**.**

$$\forall \lambda > 0, \mathbb{P}(|X - \mathbb{E}\, X| > \lambda) < \frac{\mathbb{E}(X - \mathbb{E}\, X)^2}{\lambda^2} \tag{5}$$

Proof. $\mathbb{P}(|X - \mathbb{E}\,X| > \lambda) = \mathbb{P}((X - \mathbb{E}\,X)^2 > \lambda^2)$. It follows by Markov. □

Moreover, Chebyshev can be generalized to be:

$$\forall p > 0, \forall \lambda > 0, \mathbb{P}(|X - \mathbb{E}\,X| > \lambda) < \frac{\mathbb{E}(X - \mathbb{E}\,X)^p}{\lambda^p}. \tag{6}$$

Lemma 4 (Chernoff)**.** *$X_1, \ldots, X_n$ are independent random variables, where $X_i \in [0, 1]$. Let $X = \sum_i X_i$, $\lambda > 0$,*

$$\mathbb{P}(|X - \mathbb{E}\,X| > \lambda \cdot \mathbb{E}\,X) \leq 2 \cdot e^{-\lambda^2 \cdot \mathbb{E}\,X/3}. \tag{7}$$

Proof. Since it's quite standard, and the proof detail can be found in both previous scribe[1] (Lecture 1 in Fall 2013) and wiki[2], we only include a proof sketch here. We can prove that both $\mathbb{P}(X - \mathbb{E}\,X > \lambda \cdot \mathbb{E}\,X)$ and $\mathbb{P}(X - \mathbb{E}\,X < -\lambda \cdot \mathbb{E}\,X)$ are smaller than $e^{-\lambda^2 \cdot \mathbb{E}\,X/3}$, and then apply union bound to prove the lemma.

The proof for $\mathbb{P}(X - \mathbb{E}\,X < -\lambda \cdot \mathbb{E}\,X) < e^{-\lambda^2 \cdot \mathbb{E}\,X/3}$ is symmetric to $\mathbb{P}(X - \mathbb{E}\,X > \lambda \cdot \mathbb{E}\,X) < e^{-\lambda^2 \cdot \mathbb{E}\,X/3}$. So we can focus on how to prove $\mathbb{P}(X - \mathbb{E}\,X > \lambda \cdot \mathbb{E}\,X) < e^{-\lambda^2 \cdot \mathbb{E}\,X/3}$. Since $\mathbb{P}(X - \mathbb{E}\,X > \lambda\,\mathbb{E}\,X) = \mathbb{P}(e^{t(X - \mathbb{E}\,X)} > e^{t\,\mathbb{E}\,X}) < \frac{\mathbb{E}\,e^{t(X - \mathbb{E}\,t)}}{e^{t\,\mathbb{E}\,X}}$ for any $t > 0$, we can optimize t to get the desired bound. □

Lemma 5 (Bernstein)**.** *$X_1, \ldots, X_n$ are independent random variables, where $|X_i| \leq K$. Let $X = \sum_i X_i$ and $\sigma^2 = \sum_i \mathbb{E}(X_i - \mathbb{E}\,X_i)^2$. For $\forall t > 0$,*

$$\mathbb{P}(|X - \mathbb{E}\,X| > t) \lesssim e^{-ct^2/\sigma^2} + e^{-ct/K}, \tag{8}$$

where $\lesssim$ means $\leq$ up to a constant, and c is a constant.

Proof. First, we define p ($p \geq 1$) norm for random variable Z to be $\|Z\|_p = (\mathbb{E}\,|Z|^p)^{1/p}$. In the proof, we will also use Jensen Inequality: f is convex $\Rightarrow f(\mathbb{E}\,Z) \leq \mathbb{E}\,f(Z)$.

To prove Bernstein, it's equivalent to show (equivalence left to pset)

$$\forall p \geq 1, \|\sum_i X_i - \mathbb{E}\sum_i X_i\|_p \lesssim \sqrt{p} \cdot \sigma + p \cdot K. \tag{9}$$

Let Y_i be identically distributed as X_i, with $\{X_i | i = 1, \ldots, n\}, \{Y_i | i = 1, \ldots, n\}$ independent.

We have

[1] `http://people.seas.harvard.edu/~minilek/cs229r/fall13/lec/lec1.pdf`
[2] `https://en.wikipedia.org/wiki/Chernoff_bound`

$$\|\sum_i X_i - \mathbb{E}\sum_i X_i\|_p = \|\mathop{\mathbb{E}}_Y(\sum_i X_i - \sum_i Y_i)\|_p \tag{10}$$

$$\leq \|\sum_i (X_i - Y_i)\|_p \qquad \text{(Jensen Inequality)} \tag{11}$$

$$= \|\sum_i \alpha_i (X_i - Y_i)\|_p \qquad \text{(Add uniform random signs } \alpha_i = \pm 1\text{)} \tag{12}$$

$$\leq \|\sum_i \alpha_i X_i\|_p + \|\sum_i \alpha_i Y_i\|_p \quad \text{(Triangle Inequality)} \tag{13}$$

$$= 2\|\sum_i \alpha_i X_i\|_p \tag{14}$$

$$= 2 \cdot \sqrt{\frac{\pi}{2}} \cdot \|\mathop{\mathbb{E}}_g \sum_i \alpha_i |g_i| X_i\|_p \quad \text{(Let } g \text{ be vector of iid Gaussians)} \tag{15}$$

$$\lesssim \|\sum_i \alpha_i |g_i| X_i\|_p \qquad \text{(Jensen Inequality)} \tag{16}$$

$$= \|\sum_i g_i X_i\|_p \tag{17}$$

Note that $\sum_i \alpha_i |g_i| X_i$ is Gaussian with variance $\sum_i X_i^2$. The pth moment of Gaussian $Z \sim N(0,1)$:

$$\mathbb{E}\, Z^p = \begin{cases} 0, & p \text{ is odd.} \\ \frac{p!}{(p/2)! 2^{p/2}} \leq \sqrt{p}^p, & p \text{ is even.} \end{cases} \tag{18}$$

Therefore,

$$\|\sum_i g_i X_i\|_p \leq \sqrt{p} \cdot \|(\sum_i X_i^2)^{1/2}\|_p \tag{19}$$

$$= \sqrt{p} \cdot \|\sum_i X_i^2\|_{p/2}^{1/2} \tag{20}$$

$$\leq \sqrt{p} \cdot \|\sum_i X_i^2\|_p^{1/2} \qquad (\|Z\|_p \leq \|Z\|_q \text{ for } p < q) \tag{21}$$

$$= \sqrt{p}[\|\sum_i {X_i}^2 - \mathbb{E}\sum_i X_i^2 + \mathbb{E}\sum_i X_i^2\|_p^{\frac{1}{2}}] \tag{22}$$

$$\leq \sqrt{p}[\|\mathbb{E}\sum_i X_i^2\|_p^{1/2} + \|\sum_i X_i^2 - \mathbb{E}\sum_i X_i^2\|_p^{1/2}] \tag{23}$$

$$= \sigma\sqrt{p} + \sqrt{p} \cdot \|\sum_i X_i^2 - \mathbb{E}\sum_i X_i^2\|_p^{1/2} \tag{24}$$

$$\lesssim \sigma\sqrt{p} + \sqrt{p} \cdot \|\sum_i g_i X_i^2\|_p^{1/2} \qquad \text{(Apply the same trick (10)-(17))} \tag{25}$$

Note that $\sum_i g_i X_i^2$ is Gaussian with variance $\sum_i X_i^4 \leq K^2 \cdot \sum X_i^2$, and $\sum_i g_i X_i$ is Gaussian with variance $\sum_i X_i^2$,

$$\|\sum_i g_i X_i^2\|_p \leq K \cdot \|\sum_i g_i X_i\|_p. \tag{26}$$

Let $Q = \|\sum_i g_i X_i\|_p^{1/2}$, we have

$$Q^2 - C\sigma\sqrt{p} - C\sqrt{p}\sqrt{K}Q \leq 0, \tag{27}$$

where C is a constant.

Because it's a quadratic form, Q is upper bounded by the larger root of

$$Q^2 - C\sigma\sqrt{p} - C\sqrt{p}\sqrt{K}Q = 0. \tag{28}$$

By calculation, $Q^2 \leq C\sqrt{p}\sqrt{K}Q + C\sigma\sqrt{p} \lesssim \sqrt{p} \cdot \sigma + p \cdot K$.

□

5 Analysis

Let X_n denote X after n events in Morris Algorithm.

Claim 6.

$$\mathbb{E}\, 2^{X_n} = n + 1. \tag{29}$$

Proof. We prove by induction on n.

1. Base case. It's obviously true for $n = 0$.
2. Induction step.

$$\begin{aligned}
\mathbb{E}\, 2^{X_{n+1}} &= \sum_{j=0}^{\infty} \mathbb{P}(X_n = j) \cdot \mathbb{E}(2^{X_{n+1}} | X_n = j) \\
&= \sum_{j=0}^{\infty} \mathbb{P}(X_n = j) \cdot (2^j(1 - \frac{1}{2^j}) + \frac{1}{2^j} \cdot 2^{j+1}) \\
&= \sum_{j=0}^{\infty} \mathbb{P}(X_n = j) 2^j + \sum_j \mathbb{P}(X_n = j) \\
&= \mathbb{E}\, 2^{X_n} + 1 \\
&= (n+1) + 1
\end{aligned} \tag{30}$$

□

By Chebyshev,

$$\mathbb{P}(|\tilde{n} - n| > \varepsilon n) < \frac{1}{\varepsilon^2 n^2} \cdot \mathbb{E}(\tilde{n} - n)^2 = \frac{1}{\varepsilon^2 n^2} \mathbb{E}(2^X - 1 - n)^2. \tag{31}$$

Furthermore, we can prove the following claim by induction.

Claim 7.

$$\mathbb{E}(2^{2X_n}) = \frac{3}{2}n^2 + \frac{3}{2}n + 1. \tag{32}$$

Therefore,

$$\mathbb{P}(|\tilde{n} - n| > \varepsilon n) < \frac{1}{\varepsilon^2 n^2} \cdot \frac{n^2}{2} = \frac{1}{2\varepsilon^2}. \tag{33}$$

5.1 Morris+

We instantiate s independent copies of Morris and average their outputs. Then the right hand side of (33) becomes $\frac{1}{2s\varepsilon^2} < \frac{1}{3}$ for $s > \frac{3}{2\varepsilon^2} = \Theta(\frac{1}{\varepsilon^2})$. (or $< \delta$ for $s > \frac{1}{2\varepsilon^2\delta}$)

5.2 Morris++

Run t instantiations of Morris+ with failure probability $\frac{1}{3}$. So $s = \Theta(\frac{1}{\varepsilon^2})$. Output median estimate from the s Morris+'s. It works for $t = \Theta(\lg \frac{1}{\delta})$, because if the median fails, then more than 1/2 of Morris+ fails.

Let

$$Y_i = \begin{cases} 1, & \text{if } i\text{th Morris+ fails.} \\ 0, & \text{otherwise.} \end{cases} \tag{34}$$

By Chernoff bound,

$$\mathbb{P}(\sum_i Y_i > \frac{t}{2}) \leq \mathbb{P}(|\sum_i Y_i - \mathbb{E}\sum_i Y_i| > \frac{t}{6}) \leq e^{-ct} < \delta \tag{35}$$

References

[1] Robert Morris. Counting Large Numbers of Events in Small Registers. *Commun. ACM*, 21(10): 840-842, 1978.

CS 229r: Algorithms for Big Data Fall 2015

Lecture 2 — Sept. 8, 2015

Prof. Jelani Nelson *Scribe: Jeffrey Ling*

1 Probability Recap

Chebyshev: $P(|X - \mathbb{E}X| > \lambda) < \frac{Var[X]}{\lambda^2}$

Chernoff: For $X_1, \ldots, X_n$ independent in $[0, 1]$, $\forall 0 < \epsilon < 1$, and $\mu = \mathbb{E} \sum_i X_i$,

$$P(|\sum_i X_i - \mu| > \epsilon\mu) < 2e^{-\epsilon^2\mu/3}$$

2 Today

- Distinct elements
- Norm estimation (if there's time)

3 Distinct elements (F_0)

Problem: Given a stream of integers $i_1, \ldots, i_m \in [n]$ where $[n] := \{1, 2, \ldots, n\}$, we want to output the number of distinct elements seen.

3.1 Straightforward algorithms

1. Keep a bit array of length n. Flip bit if a number is seen.
2. Store the whole stream. Takes $m \lg n$ bits.

We can solve with $O(\min(n, m \lg n))$ bits.

3.2 Randomized approximation

We can settle for outputting $\widetilde{t}$ s.t. $P(|t - \widetilde{t}| > \epsilon t) < \delta$. The original solution was by Flajolet and Martin [2].

3.3 Idealized algorithm

1. Pick random function $h : [n] \to [0, 1]$ (idealized, since we can't actually nicely store this)
2. Maintain counter $X = \min_{i \in stream} h(i)$
3. Output $1/X - 1$

Intuition. X is a random variable that's the minimum of t i.i.d $Unif(0, 1)$ r.v.s.

Claim 1. $\mathbb{E}X = \frac{1}{t+1}$.

Proof.

$$\begin{aligned}
\mathbb{E}X &= \int_0^\infty P(X > \lambda) d\lambda \\
&= \int_0^\infty P(\forall i \in str, h(i) > \lambda) d\lambda \\
&= \int_0^\infty \prod_{r=1}^{t} P(h(i_r) > \lambda) d\lambda \\
&= \int_0^1 (1 - \lambda)^t d\lambda \\
&= \frac{1}{t+1}
\end{aligned}$$

□

Claim 2. $\mathbb{E}X^2 = \frac{2}{(t+1)(t+2)}$

Proof.

$$\begin{aligned}
\mathbb{E}X^2 &= \int_0^1 P(X^2 > \lambda) d\lambda \\
&= \int_0^1 P(X > \sqrt{\lambda}) d\lambda \\
&= \int_0^1 (1 - \sqrt{\lambda})^t d\lambda && u = 1 - \sqrt{\lambda} \\
&= 2 \int_0^1 u^t (1 - u) du \\
&= \frac{2}{(t+1)(t+2)}
\end{aligned}$$

□

This gives $Var[X] = \mathbb{E}X^2 - (\mathbb{E}X)^2 = \frac{t}{(t+1)^2(t+2)}$, and furthermore $Var[X] < \frac{1}{(t+1)^2} = (\mathbb{E}X)^2$.

4 FM+

We average together multiple estimates from the idealized algorithm FM.

1. Instantiate $q = 1/\epsilon^2\eta$ FMs independently
2. Let X_i come from FM_i.
3. Output $1/Z - 1$, where $Z = \frac{1}{q}\sum_i X_i$.

We have that $\mathbb{E}(Z) = \frac{1}{t+1}$, and $Var(Z) = \frac{1}{q}\frac{t}{(t+1)^2(t+2)} < \frac{1}{q(t+1)^2}$.

Claim 3. $P(|Z - \frac{1}{t+1}| > \frac{\epsilon}{t+1}) < \eta$

Proof. Chebyshev.

$$P(|Z - \frac{1}{t+1}| > \frac{\epsilon}{t+1}) < \frac{(t+1)^2}{\epsilon^2}\frac{1}{q(t+1)^2} = \eta$$

□

Claim 4. $P(|(\frac{1}{Z} - 1) - t| > O(\epsilon)t) < \eta$

Proof. By the previous claim, with probability $1 - \eta$ we have

$$\frac{1}{(1 \pm \epsilon)\frac{1}{t+1}} - 1 = (1 \pm O(\epsilon))(t+1) - 1 = (1 \pm O(\epsilon))t \pm O(\epsilon)$$

□

5 FM++

We take the median of multiple estimates from FM+.

1. Instantiate $s = \lceil 36\ln(2/\delta)\rceil$ independent copies of FM+ with $\eta = 1/3$.
2. Output the median $\widehat{t}$ of $\{1/Z_j - 1\}_{j=1}^{s}$ where Z_j is from the jth copy of FM+.

Claim 5. $P(|\widehat{t} - t| > \epsilon t) < \delta$

Proof. Let

$$Y_j = \begin{cases} 1 & \text{if } |(1/Z_j - 1) - t| > \epsilon t \\ 0 & \text{else} \end{cases}$$

We have $\mathbb{E}Y_j = P(Y_j = 1) < 1/3$ from the choice of η. The probability we seek to bound is equivalent to the probability that the median fails, i.e. at least half of the FM+ estimates have $Y_j = 1$. In other words,

$$\sum_{j=1}^{s} Y_j > s/2$$

We then get that

$$P(\sum Y_j > s/2) = P(\sum Y_j - s/3 > s/6) \tag{1}$$

Make the simplifying assumption that $\mathbb{E}Y_j = 1/3$ (this turns out to be stronger than $\mathbb{E}Y_j < 1/3$. Then equation 1 becomes

$$P(\sum Y_j - \mathbb{E}\sum Y_j > \frac{1}{2}\mathbb{E}\sum Y_j)$$

using Chernoff,

$$< e^{-\frac{(\frac{1}{2})^2 s/3}{3}} < \delta$$

as desired. □

The final space required, ignoring h, is $O(\frac{\lg(1/\delta)}{\epsilon^2})$ for $O(\lg(1/\delta))$ copies of FM+ that require $O(1/\epsilon^2)$ space each.

6 k-wise independent functions

Definition 6. *A family $\mathcal{H}$ of functions mapping $[a]$ to $[b]$ is k-wise independent if $\forall j_1, \ldots, j_k \in [b]$ and $\forall$ distinct $i_1, \ldots, i_k \in [a]$,*

$$P_{h\in\mathcal{H}}(h(i_1) = j_1 \wedge \ldots \wedge h(i_k) = j_k) = 1/b^k$$

Example. The set $\mathcal{H}$ of all functions $[a] \to [b]$ is k-wise independent for every k. $|\mathcal{H}| = b^a$ so h is representable in $a \lg b$ bits.

Example. Let $a = b = q$ for $q = p^r$ a prime power, then $\mathcal{H}_{poly}$, the set of degree $\leq k-1$ polynomials with coefficients in $\mathbb{F}_q$, the finite field of order q. $|\mathcal{H}_{poly}| = q^k$ so h is representable in $k \lg p$ bits.

Claim 7. *$\mathcal{H}_{poly}$ is k-wise independent.*

Proof. Interpolation. □

7 Non-idealized FM

First, we get an $O(1)$-approximation in $O(\lg n)$ bits, i.e. our estimate $\widetilde{t}$ satisfies $t/C \leq \widetilde{t} \leq Ct$ for some constant C.

1. Pick h from 2-wise family $[n] \to [n]$, for n a power of 2 (round up if necessary)
2. Maintain $X = \max_{i\in str} lsb(h(i))$ where lsb is the least significant bit of a number
3. Output 2^X

For fixed j, let Z_j be the number of i in stream with $lsb(h(i)) = j$. Let $Z_{>j}$ be the number of i with $lsb(h(i)) > j$.

Let

$$Y_i = \begin{cases} 1 & lsb(h(i)) = j \\ 0 & \text{else} \end{cases}$$

Then $Z_j = \sum_{i \in str} Y_i$. We can compute $\mathbb{E}Z_j = t/2^{j+1}$ and similarly

$$\mathbb{E}Z_{>j} = t(\frac{1}{2^{j+2}} + \frac{1}{2^{j+3}} + \ldots) < t/2^{j+1}$$

and also

$$Var[Z_j] = Var[\sum Y_i] = \mathbb{E}(\sum Y_i)^2 - (\mathbb{E}\sum Y_i)^2 = \sum_{i_1, i_2} \mathbb{E}(Y_{i_1} Y_{i_2})$$

Since h is from a 2-wise family, Y_i are pairwise independent, so $\mathbb{E}(Y_{i_1} Y_{i_2}) = \mathbb{E}(Y_{i_1})\mathbb{E}(Y_{i_2})$. We can then show

$$Var[Z_j] < t/2^{j+1}$$

Now for $j^* = \lceil \lg t - 5 \rceil$, we have

$$16 \leq \mathbb{E}Z_{j^*} \leq 32$$

$$P(Z_{j^*} = 0) \leq P(|Z_{j^*} - \mathbb{E}Z_{j^*}| \geq 16) < 1/5$$

by Chebyshev.

For $j = \lceil \lg t + 5 \rceil$

$$\mathbb{E}Z_{>j} \leq 1/16$$

$$P(Z_{>j} \geq 1) < 1/16$$

by Markov.

This means with good probability the max lsb will be above j^* but below j, in a constant range. This gives us a 32-approximation, i.e. constant approximation.

8 Refine to $1 + \epsilon$

Trivial solution. Algorithm TS stores first C/ϵ^2 distinct elements. This is correct if $t \leq C/\epsilon^2$.

Algorithm.

1. Instantiate $\text{TS}_0, \ldots, \text{TS}_{\lg n}$
2. Pick $g : [n] \to [n]$ from 2-wise family
3. Feed i to $\text{TS}_{lsb(g(i))}$
4. Output 2^{j+1} out where $t/2^{j+1} \approx 1/\epsilon^2$.

Let B_j be the number of distinct elements hashed by g to TS_j. Then $\mathbb{E}B_j = t/2^{j+1} = Q_j$. By Chebyshev $B_j = Q_j \pm O(\sqrt{Q_j})$ with good probability. This equals $(1 \pm O(\epsilon))Q_j$ if $Q_j \geq 1/\epsilon^2$.

Final space: $\frac{C}{\epsilon^2}(\lg n)^2 = O(\frac{1}{\epsilon^2}\lg^2 n)$ bits.

It is known that space $O(1/\epsilon^2 + \log n)$ is achievable [4], and furthermore this is optimal [1, 5] (also see [3]).

References

[1] Noga Alon, Yossi Matias, Mario Szegedy The Space Complexity of Approximating the Frequency Moments. *J. Comput. Syst. Sci.* 58(1): 137–147, 1999.

[2] Philippe Flajolet, G. Nigel Martin Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.*, 31(2):182–209, 1985.

[3] T. S. Jayram, Ravi Kumar, D. Sivakumar: The One-Way Communication Complexity of Hamming Distance. *Theory of Computing* 4(1): 129–135, 2008.

[4] Daniel M. Kane, Jelani Nelson, David P. Woodruff An optimal algorithm for the distinct elements problem. In *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 41–52, 2010.

[5] David P. Woodruff. Optimal space lower bounds for all frequency moments. In *SODA*, pages 167–175, 2004.

CS 229r: Algorithms for Big Data Fall 2015

Lecture 03, September 11th

Prof. Jelani Nelson *Scribe: Vasileios Nakos*

1 Overview

In the last lecture we looked at distinct elements, k-wise independence, geometric subsampling of streams.

In this lecture we will see lower bounds on exact and approximate deterministic algorithms on data streams, as well as the AMS sketch and Indyk's algorithm for the F_p moments, when $0 \leq p \leq 2$.

2 Main Section

2.1 Space lower bounds for streaming algorithms

We begin by an impossibility result. We consider the distinct elements problems, where you want to find the number of distinct elements in a stream, where queries and additions are permitted. We will denote by s the space of the algorithm, n the size of the universe from which the elements come from, and m the length of the stream. We have the following result [1].

Theorem 1. *There is no deterministic exact algorithm for computing number of distinct elements in $O(minn, m)$ space.*

Proof. We are going to make an information-theoretic argument. Using a streaming algorithm with space s for the current problem we are going to show how to encode $\{0,1\}^n$ using only s bits. In other words, we are going to construct an injective mapping from $\{0,1\}^n$ to $\{0,1\}^s$. So, this implies that s must be at least n and we are done. We look for procedures Dec,Enc such that $\forall x Dec(Enc(x)) = x$ and $Enc(x)$ is a function from $\{0,1\}^n$ to $\{0,1\}^s$.
For the encoding procedure, given a string x, create a stream containing and append i at the end of the stream if $x_i = 1$. Then $Enc(x)$ is the memory content of the algorithm on that stream.
For the decoding procedure, we are going to look at each i and append it at the end of the stream(feed it to the streaming algorithm) and query then the number of distinct elements. If the number of distinct elements increases this implies that $x_i = 0$, otherwise it implies that $x_i = 1$. So we can recover x completely and this finishes the proof.

□

We move on by showing that even approximate algorithms are hopeless for this problem.

Theorem 2. *Any deterministic F_0 algorithm that provides* 1.1 *approximation requires $\Omega(n)$ space*

Proof. Suppose we had a colection C satisfying:

- $|C| \geq 2^{cn}$, for some constant $c < 1$.
- $\forall S \in C, |S| = \frac{n}{100}$
- $\forall S \neq T \in C, |S \cap T| \leq \frac{n}{2000} \leq \frac{1}{20}|S|$

We leave the proof of existence of such a set later and we are moving to our lower bound. We are going to use the algorithm to encode vectors $x_S \forall S \in C$, where x_S is the indicator vector of set S. The lower bound follows as before since we must have $s \geq cn$ The encoding procedure is going to be the same as before.
For the decoding procedure, we are going to iterate over all sets and test for each set S if it corresponds to our initial encoded set(remember we are just doing an information-theoretic argument and we do not care about the running running- we only care that this map is an injection). For that, we keep at each time the memory contents of M of the streaming algorithm after having inserted our initial string. Then for each S, we initialize our algorithm with memory contents M and then feed element i if $i \in S$. It is easy to see that if S equals the initial encoded set, the number of distinct elements does not increase by much, whereas if it is not it almost doubles. Taking also into account the approximation guarantee of the algorithm we see for example that if S is not our initial set then the number of distinct elements grows by $\frac{3}{2}$(we can tune the parametersif needed).

We now only need to prove the existence of such a family of sets C. We are going to argue via probabilistic method. We partition n into $\frac{n}{100}$ intervals of length 100 each in the obvious way. To form a set S we pick one number from each interval uniformly at random. Obviously, such a set has size exactly $\frac{n}{100}$. For two sets S, T chosen uniformly at random as before let U_i be the random variable that equals 1 if they have the same number chosen from interval i. Obviously $P[U_i = 1] = \frac{1}{100}$. So the expected size of the intersection is just $E \sum_{i=1}^{\frac{n}{100}} = \frac{n}{100} \cdot \frac{1}{100}$. The probability that this interesction is bigger than five times its mean is smaller than $e^{-c'n}$ for some constant c', by a standard Chernoff bound. We can then apply a union bound over all possible intersections and get the desired result.

□

2.2 Linear Sketches and upper bounds

2.2.1 What is a Linear Sketch?

We introduce the turnstile model in streaming algorithms. In this model we have a vector $x \in \mathbb{R}^n$ that starts as the all zero vector and then a sequence of updates comes. Each update is of the form (i, Δ), where $\Delta \in \mathbb{R}$ and $i \in \{1, .., n\}$. This corresponds to the operation $x_i \leftarrow x_i + \Delta$.
Given a function f, we want to compute or approximate $f(x)$.For example in the problem of distinct elements Δ is always 1 and $f(x) = |i : x_i \neq 0$.
The common/only technique for designing turnstile algorithms is **linear sketching**. The idea is to maintain in memory $y = \Pi x$, where $\Pi \in \mathbb{R}^{m \times n}$, a matrix that is short and fat. We care that $m < n$, usually much smaller. We can see that y is m-dimensional so we can store it efficiently but what about Π? If we need to store the whole Π in memory this will not lead to a better algorithm in terms of space. So, there are two common ways in constructing and storing Π. The one is that Π is deterministic and so we can easily compute Π_{ij} without keeping the whole matrix in memory.

The other way is that Π is defined by k-wise independent hash functions for some small k, so we can afford storing the hash functions and computing Π_{ij}.

Let's see now how updates happen when we have a linear sketch. Let Π^i be the i-th column of the matrix Π. Then $\Pi_x = \sum_{i=1}^n \Pi^i x_i$. So by storing $y = \Pi x$ when the update (i, Δ) comes we have that the new y equals $\Pi(x + \Delta e_i) = \Pi x + \Delta \Pi^i$. Observe that the first summand is the old y and the second summand is just some multiple of the i-th column of Π. This means that if I can tell which is the i-th column of Π in small space I can also perform updates in small space.

2.2.2 Moment Estimation for $p = 2$

This problem was investigated by Alon,Matis, Szegedy at StoC 96 [1]. Let $F_p = \|x\|_p^p = \sum_{i=1}^p |x_i|^p$. We want to estimate the space needed to solve the moment estimation problem as p changes. There is a transition point in complexity of F_p.

- $0 \leq p \leq 2, poly(\frac{logn}{\epsilon})$ space is achievable for $(1+\epsilon)$ approximation with $\frac{2}{3}$ success probability [1, 3].
- For $p > 2$ then we need exactly $\Theta(n^{1-\frac{2}{p}} poly(\frac{logn}{\epsilon}))$ bits of space for $(1+\epsilon)$ space with $\frac{2}{3}$ success probability [2, 4].

We not look at the case $p = 2$ which is the AMS sketch. Let $\sigma_i \in \{-1, 1\}^n$ be random signs coming from a 4-wise independent family. We need $O(logn)$ bits to represent such a family. Then set $y_i = \sum_{j=1}^n \sigma_{ij} x_j$. This means that our matrix Π we were referring before has rows the vectors σ_i. Observe that we can get columns because we just need to evaluate hash functions. Our algorithm is going to output $\frac{1}{m}\|y\|_2^2$ as estimator of $\|x\|_2^2$.

It holds that $\mathbb{E}y_i^2 = \mathbb{E}(\sum_{j=1}^n \sigma_{ij} x_j)^2 = \|x\|_2^2 + \mathbb{E}\sum_{j \neq j'} \sigma_{ij}\sigma_{ij'} x_j x_{j'}$. Moreover we need to estimate the variance and apply Chebyshev's inequality. Observe that $\mathbb{E}y_i^4 = \mathbb{E}(\sum_j \sigma_{ij} x_j)^4 = \mathbb{E}\sum_{j_1,j_2,j_3,j_4 \in [n]^4} \sigma_{ij_1}\sigma_{ij_2}\sigma_{ij_3}\sigma_{ij_4} x_{ij_1} x_{ij_2} x_{ij_3} x_{ij_4} = \sum_{j=1}^n x_j^4 + (1/2)\binom{4}{2} \sum_{j \neq j'} x_j^2 x_{j'}^2$, because when we take the expectation if a term in a summand appears odd number of times its expectation is going to be zero and cancel the whole summand, so we are left only with the summands in which all terms appear an even number of times. Calculating the variance and walking through the calculations we get the desired result.

Facts: We can also use gaussians instead of random signs. The analysis would be similar, but we would have to discretize the gaussians in order to avoid the need of infinite precision. We can also have only one hash function $\sigma : [m] \times [n] \to \{-1, 1\}$, as long as it is 4-wise independent. The difference now is that we need to compute expectation and variance of the whole estimator $\frac{1}{m}\sum_{i=1}^m y_i^2$ instead of each term individually.

2.2.3 Moment Estimation for $p \leq 2$

We are going to describe an idealized algorithm with infinite precision, given by Indyk [3]. Call a distribution $\mathbb{D}$ over $\mathbb{R}$ $p-stable$ if for $z_1, .., z_n$ iid from this distribution and for all $x \in \mathbb{R}^n$ we have that $\sum_{i=1}^n z_i x_i$ is a random variable with distribution $\|x\|_p \mathbb{D}$. An example of such a distribution are the gaussians for $p = 2$ and for $p = 1$ the Cauchy distribution, which has probability density function $pdf(x) = \frac{1}{\pi(x+1)^2}$.

Note: Observe that by the Central Limit Theorem an average of d samples from a distribution approaches a gaussian as d goes to infinity. But, if we pick a p-stable distribution for $p < 2$, such as Cauchy distribution, why does this not violate the Central Limit Theorem? Actually, for CLT to hold we also need bounded moments of the distribution. But for example second moment of Cauchy is unbounded so the theorem does not apply. Intuitively, one can think that p-stable distributions cannot exist for $p > 2$ because they violate Central Limit Theorem. Indeed, this is the case as was proved by Lévy in the 20's [5]. In fact, he proved that p-stable distributions exist if and only if $0 \leq p \leq 2$.

We get back to Indyk's algorithm. In fact the algorithm is very simple. Let the i-th row of Π be z_i, as before, where z_i comes from a p-stable distribution. Then consider $y_i = \sum_{j=1}^n z_{ij} x_j$. When a query comes, output the median of all the y_i. We can assume that without loss of generality a p-stable distribution has median equal to 1, which in fact means that for z from this distribution $\mathbb{P}(-1 \leq z \leq 1) \leq \frac{1}{2}$.

2.3 Next time

In the next lecture we are going to analyze Indyk's algorithm.

References

[1] Noga Alon, Yossi Matias, Mario Szegedy. The Space Complexity of Approximating the Frequency Moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.

[2] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar. An information statistics approach to data stream and communication complexity. *J. Comput. Syst. Sci.* 68(4):702–732, 2004.

[3] Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. ACM* 53(3):307–323, 2006.

[4] Piotr Indyk, David P. Woodruff. Optimal approximations of the frequency moments of data streams. *STOC*, pages 202–208, 2005.

[5] Paul Lévy. Calcul des probabilités. Gauthier-Villars, Paris, 1925.

CS 229r: Algorithms for Big Data Fall 2015

Lecture 4 — September 15, 2015

Prof. Jelani Nelson *Scribe: Hyunghoon Cho*

1 Recap

Recall the following definition of *p-stable distributions.*

Definition 1. *D_p is a p-stable distribution on $\mathbf{R}$ if, for any n independent samples $z_1, \ldots, z_n$ from D_p, the following holds: $\forall x \in \mathbf{R}^n, \sum_{i=1}^n x_i z_i \sim ||x||_p D_p$.*

In other words, any linear combination of a set of samples from a p-stable distribution must itself be a sample from the same distribution, scaled by the p-norm of the weight vector.

Theorem 1. *A p-stable distribution exists iff $0 < p \leq 2$.*

We know that the standard Gaussian distribution is 2-stable and the standard Cauchy distribution is 1-stable. Although in most cases a simple closed-form expression for p-stable distributions is not known, there is an efficient way to generate samples for any $p \in (0, 2]$. If we let $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ and $r \in [0, 1]$ be uniformly random samples, then

$$\frac{\sin(p\theta)}{\cos^{1/p}(\theta)} \left(\frac{\cos(\theta(1-p))}{\ln(1/r)} \right)^{\frac{1-p}{p}}$$

is a sample from a p-stable distribution [CMS76].

2 Indyk's Algorithm

2.1 Overview

Let $\Pi = \{\pi_{ij}\}$ be an $m \times n$ matrix where every element π_{ij} is sampled from a p-stable distribution, D_p. Given $x \in \mathbf{R}^n$, Indyk's algorithm [Indyk06] estimates the p-norm of x as

$$||x||_p \approx \text{median}_{i=1,\ldots,m} |y_i|,$$

where $y = \Pi x$. Recall from last lecture that in a turnstile streaming model, each element in the stream reflects an update to an entry in x. While a naive algorithm would maintain x in memory and calculate $||x||_p$ at the end, thus requiring $\Theta(n)$ space, Indyk's algorithm stores y and Π. Combined with a space-efficient way to produce Π (described later in the lecture) we achieve better space complexity.

2.2 Analysis

For simplicity of analysis, we will assume Π is generated with D_p such that if $Z \sim D_p$ then median$(|Z|) = 1$. In other words, we assume the probability mass of D_p assigned to interval $[-1, 1]$ is $1/2$. In addition, let $I_{[a,b]}(x)$ be an indicator function defined as

$$I_{[a,b]}(x) = \begin{cases} 1 & x \in [a, b], \\ 0 & \text{otherwise.} \end{cases}$$

Let Z_i be the i-th row of Π. We have

$$y_i = \sum_{j=1}^{n} Z_{ij} x_j \sim ||x||_p D_p, \tag{1}$$

which follows from the definition of p-stable distributions and noting that Z_{ij}'s are sampled from D_p. This implies

$$\mathbf{E}\left[I_{[-1,1]}\left(\frac{y_i}{||x||_p} \right) \right] = \frac{1}{2}, \tag{2}$$

since $y_i/||x||_p \sim D_p$.

Moreover, it can be shown that

$$\mathbf{E}\left[I_{[-1-\varepsilon,1+\varepsilon]}\left(\frac{y_i}{||x||_p} \right) \right] = \frac{1}{2} + \Theta(\varepsilon), \tag{3}$$

$$\mathbf{E}\left[I_{[-1+\varepsilon,1-\varepsilon]}\left(\frac{y_i}{||x||_p} \right) \right] = \frac{1}{2} - \Theta(\varepsilon). \tag{4}$$

Next, consider the following quantities:

$$C_1 = \frac{1}{m} \sum_{i}^{m} I_{[-1-\varepsilon,1+\varepsilon]}\left(\frac{y_i}{||x||_p} \right), \tag{5}$$

$$C_2 = \frac{1}{m} \sum_{i}^{m} I_{[-1+\varepsilon,1-\varepsilon]}\left(\frac{y_i}{||x||_p} \right). \tag{6}$$

C_1 represents the fraction of y_i's that satisfy $|y_i| \leq (1+\varepsilon)||x||_p$, and similarly, C_2 represents the fraction of y_i's that satisfy $|y_i| \leq (1-\varepsilon)||x||_p$. By linearity of expectation, we have $\mathbf{E}[C_1] = 1/2 + \Theta(\varepsilon)$ and $\mathbf{E}[C_2] = 1/2 - \Theta(\varepsilon)$. Therefore, in expectation, the median of $|y_i|$ lies in

$$[(1-\varepsilon)||x||_p, (1+\varepsilon)||x||_p]$$

as desired.

Now we analyze the variance of C_1 and C_2. We have

$$\mathbf{Var}\left(C_1\right) = \frac{1}{m^2} \times m \times (\text{variance of the indicator variable}). \tag{7}$$

Since variance of any indicator variable is at most 1, $\mathbf{Var}(C_1) \leq \frac{1}{m}$. Similarly, $\mathbf{Var}(C_2) \leq \frac{1}{m}$. With an appropriate choice of m now we can ensure that the median of $|y_i|$ is in the desired ε-range of $||x||_p$ with high probability.

2.3 Derandomizing Π

We have shown that Indyk's algorithm work, but independently generating and storing all mn elements of Π is expensive. Can we get by with a smaller degree of randomness? To invoke the definition of p-stable distributions for Equation 1, we need the entries in each row to be independent from one another. And the rows need to be pairwise independent for our calculation of variance to hold. As a side note, generating pairwise independent Gaussian or Cauchy random variables can be achieved by discretizing the space and treating them as discrete random variables. One can make a claim that, with fine enough discretization, the algorithm still succeeds with high probability.

Now suppose $w_i = \sum_{j=1}^{n} Q_{ij}x_j$ where Q_{ij}'s are k-wise independent p-stable distribution samples. What we want is an argument of the form

$$\mathbf{E}\left[I_{[a,b]}\left(\frac{w_i}{||x||_p}\right)\right] \approx_\varepsilon \mathbf{E}\left[I_{[a,b]}\left(\frac{y_i}{||x||_p}\right)\right]. \tag{8}$$

If we can make such claim, then we can use k-wise independent samples in each row in lieu of fully independent samples to invoke the same arguments in the analysis above. This has been shown for $k = \Omega(1/\varepsilon^p)$ [KNW10], but we are not going to cover this in class. With this technique, we can specify Π using only $O(k \lg n)$ bits; across rows, we only need to use 2-wise independent hash function that maps a row index to a $O(k \lg n)$ bit seed for the k-wise independent hash function. Indyk's approach to derandomizing Π, while the results are not as strong, employs a useful technique, so we will cover that here instead.

2.3.1 Branching Programs

A *branching program* can be described with a grid of nodes representing different states of the program—we can think of it as a DFA. The program starts at an initial node which is not part of the grid. At each step, the program reads S bits of input, reflecting the fact that space is bounded by S, and makes a decision about which node in the subsequent column of the grid to jump to. After R steps (number of columns in the grid), the final node visited by the program represents the outcome. The entire input, which can be represented as a length-RS bit string, induces a distribution over the final states. Our goal is to generate the input string using fewer ($\ll RS$) random bits such that the original distribution over final states is well preserved. The following theorem addresses this goal.

Theorem 2. *([Nisan92]) There exists $h : \{0,1\}^t \to \{0,1\}^{RS}$ for $t = O(S \lg R)$ such that*

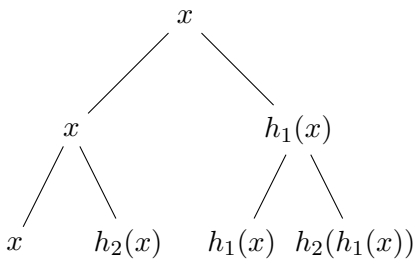
$$\left|P_{x\sim U(\{0,1\}^{RS})}\{f(B(x)) = 1\} - P_{y\sim U(\{0,1\}^t)}\{f(B(h(y))) = 1\}\right| \leq \frac{1}{2^S}. \tag{9}$$

for any branching program B and any function $f : \{0,1\}^S \to \{0,1\}$.

In other words, such function h can simulate the input to the branching program with only t random bits such that it is almost impossible to discriminate the outcome of the simulated program from that of the original program.

2.3.2 Nisan's Pseudorandom Generator (PRG)

What does such a function look like? We start by taking a random sample x from $\{0,1\}^S$. Then we place x at the root and repeat the following procedure to create a complete binary tree. At each node, create two children and copy the string over to the left child. For the right child, use a random 2-wise independent hash function $h_j : [2^S] \to [2^S]$ chosen for the corresponding level of the tree and record the result of the hash. Once we reach R levels, output the concatenation of all leaves, which is a length-RS bit string. To illustrate, the first few levels of our tree would look like:



Since each hash function requires S random bits and there are $\lg R$ levels in the tree, this function uses $O(S \lg R)$ bits total. We will not discuss why this function satisfies the condition in Theorem 2.

2.3.3 Back to Indyk's Algorithm

Now we will use Nisan's PRG to derandomize Π in Indyk's algorithm. Consider the following program:

1. Initialize $c_1 \leftarrow 0$, $c_2 \leftarrow 0$
2. For $i = 1, \ldots, m$:
   (a) Initialize $y \leftarrow 0$
   (b) For $j = 1, \ldots, n$:
      i. Update $y \leftarrow y + \pi_{ij} x_j$
   (c) If $y \leq (1+\varepsilon)||x||_p$, then increment c_1
   (d) If $y \leq (1-\varepsilon)||x||_p$, then increment c_2

Note that this program only uses $O(\lg n)$ bits and is a branching program that mimics the proof of correctness for Indyk's algorithm. More specifically, Indyk's algorithm succeeded iff at the end of this program $c_1 > \frac{m}{2}$ and $c_2 < \frac{m}{2}$. The only source of randomness in this program are the π_{ij}'s. We will use Nisan's PRG to produce these random numbers. We invoke Theorem 2 with the above program as B and an indicator function checking whether the algorithm succeeded or not as f. Note that the space bound is $S = O(\lg n)$ and the number of steps taken by the program is $R = O(mn)$, or $O(n^2)$ since $m \leq n$. This means we can "fool" the proof of correctness of Indyk's algorithm only using $O(\lg^2 n)$ random bits to generate Π.

3 The $p > 2$ Case

Indyk's algorithm uses p-stable distributions which only exist for $p \in (0, 2]$. What do we do when $p > 2$?

Theorem 3. *$n^{1-2/p} poly(\frac{\lg n}{\varepsilon})$ space is necessary and sufficient.*

A nearly optimal lower bound was given by [BarYossef04] and first achieved by [IW05]. In this lecture we will discuss the algorithm of [Andoni12], which is based on [AKO11] and [JST11]. We will focus on $\varepsilon = \Theta(1)$. Refining this result may be included in the next homework.

In this algorithm, we let $\Pi = PD$. P is a $m \times n$ matrix, where each column has a single non-zero element that is either 1 or -1. D is a $n \times n$ diagonal matrix with $d_{ii} = u_i^{-1/p}$, where $u_i \sim \text{Exp}(1)$. In other words,

$$P\{u_i > t\} = \begin{cases} 1 & t \leq 0, \\ e^{-t} & t > 0. \end{cases}$$

Similar to the $0 < p \leq 2$ case, we will keep $y = \Pi x$, but here we estimate $||x||_p$ with

$$||x||_p \approx ||y||_\infty = \max_i |y_i|. \tag{10}$$

Theorem 4. *$P\left\{\frac{1}{4}||x||_p \leq ||y||_\infty \leq 4||x||_p\right\} \geq \frac{11}{20}$ for $m = \Theta(n^{1-2/p} \lg n)$.*

Let $z = Dx$, which means $y = Pz$. To prove Theorem 4, we will first show that $||z||_\infty$ provides a good estimate and then show that applying P to z preserves this. The latter step is deferred to next class due to time constraints.

Claim 1. *$P\left\{\frac{1}{2}||x||_p \leq ||z||_\infty \leq 2||x||_p\right\} \geq \frac{3}{4}$*

Proof. Let $q = \min\left\{\frac{u_1}{|x_1|^p}, \ldots, \frac{u_n}{|x_n|^p}\right\}$. We have

$$\begin{aligned} P\{q > t\} &= P\left\{\forall i, u_i > t|x_i|^p\right\} && (11) \\ &= \prod_{i=1}^{n} e^{-t|x_i|^p} && (12) \\ &= e^{-t||x||_p^p}, && (13) \end{aligned}$$

which implies $q \sim \frac{Exp(1)}{||x||_p^p}$. Thus,

$$\begin{aligned} P\left\{\frac{1}{2}||x||_p \leq ||z||_\infty \leq 2||x||_p\right\} &= P\left\{\frac{1}{2^p}||x||_p^{-p} \leq q \leq 2^p||x||_p^{-p}\right\} && (14) \\ &= e^{-\frac{1}{2^p}} - e^{-2^p} && (15) \\ &\geq \frac{3}{4}, && (16) \end{aligned}$$

for $p > 2$.

□

References

[CMS76] J. M. Chambers, C. L. Mallows, and B. W. Stuck. A method for simulating stable random variables. *Journal of the american statistical association*, 71.354 (1976): 340-344.

[KNW10] Daniel Kane, Jelani Nelson, David Woodruff. On the exact space complexity of sketching and streaming small norms. *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms (SODA)*, 2010.

[BarYossef04] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar An information statistics approach to data stream and communication complexity. *Journal of Computer and System Sciences*, 68 (2004): 702-732.

[Andoni12] Alexandr Andoni. High frequency moments via max-stability. Manuscript, 2012. `http://web.mit.edu/andoni/www/papers/fkStable.pdf`

[AKO11] Alexandr Andoni, Robert Krauthgamer, Krzysztof Onak. Streaming Algorithms via Precision Sampling. *FOCS*, pgs. 363–372, 2011.

[JST11] Hossein Jowhari, Mert Saglam, Gábor Tardos. Tight bounds for L_p samplers, finding duplicates in streams, and related problems. *PODS*, pgs. 49–58, 2011.

[Indyk06] Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. ACM* 53(3): 307–323, 2006.

[IW05] Piotr Indyk, David P. Woodruff. Optimal approximations of the frequency moments of data streams. *STOC*, pgs. 202–208, 2005.

[Nisan92] Noam Nisan. Pseudorandom Generators for Space-Bounded Computation. *Combinatorica*, 12(4):449-461, 1992.

CS 229r: Algorithms for Big Data Fall 2015

Lecture 5 — September 17, 2015

Prof. Jelani Nelson *Scribe: Yakir Reshef*

1 Recap and overview

Last time we discussed the problem of norm estimation for p-norms with $p > 2$. We had described an algorithm by [Andoni12] that, given $x \in \mathbb{R}^n$ updated under a turnstile model, approximates $\|x\|_p$ with constant multiplicative error. The algorithm generates two random matrices $P \in \mathbb{R}^{m \times n}$ (with $m \ll n$) and $D \in \mathbb{R}^{n \times n}$. P is sampled so that each of its columns contains all zeros except for one entry, which contains a random sign. D is a diagonal matrix whose i-th diagonal entry is $u_i^{-1/p}$ where the u_i are i.i.d. exponential random variables. The algorithm then maintains $y = PDx$, and its output is $\|y\|_\infty = \max_i |y_i|$.

In this lecture we will complete the proof of correctness of this algorithm and then move on from p-norm estimation to other problems related to linear sketching.

2 Completing the proof of correctness

From last time we have the following claim.

Claim 1. *Let $Z = DX$. Then*

$$P\left(\frac{1}{2}\|x\|_p \leq \|Z\|_\infty \leq 2\|x\|_p\right) \geq 3/4$$

This claim establishes that if we could maintain Z instead of y then we would have a good solution to our problem. Remember though that we can't store Z in memory because it's n-dimensional and $n \gg m$. That's why we need to analyze $PZ \in \mathbb{R}^m$.

2.1 Overview of analysis of $y = PZ$

The idea behind our analysis of $y = PZ$ is as follows: each entry in y is a sort of counter. And the matrix P takes each entry in Z, hashes it to a perfectly random counter, and adds that entry of Z times a random sign to that counter. Since $n > m$ and there are only m counters, there will be collisions, and these will cause different Z_i to potentially cancel each other out or add together in a way that one might expect to cause problems. We'll get around this by showing that there are very few large Z_i's, so few relative to m that with high probability none of them will collide with each other.

We still need to worry, because small Z_i's and big Z_i's might collide with each other. But remember that when we add the small Z_i's, we multiply them with a random sign. So the expectation of the aggregate contributions of the small Z_i's to each bucket is 0. We'll bound their variance as well,

which will show that if they collide with big Z_i's then with high probability this won't substantially change the relevant counter. All of this together will show that the maximal counter value (i.e., $\|y\|_\infty$) is close to the maximal Z_i – and therefore to $\|x\|_p$ – with high probability

2.2 Analysis of $y = PZ$

We make the following definitions.

- Let $T = \|x\|_p$.
- Define the "heavy indices" as $H = \{j : |Z_j| \geq T/(v \lg(n))\}$. Think of c as big. We'll set it later.
- Define the "light indices" as $L = [n] \backslash H$.

2.2.1 Analyzing the heavy indices

We begin by showing that there will not be many heavy indices.

Claim 2. *For any $\ell > 0$, we have*

$$E\left(\left|\left\{i \in [n] : |Z_i| > \frac{T}{\ell}\right\}\right|\right) < \ell^p$$

Before we prove this claim, let's reflect: if $\ell = v \lg(n)$ then we get polylog(n) heavy indices, which is miniscule compared to the $m = O(n^{1-2/p} \ln(n))$ counters. Birthday paradox-type reasoning will then translate this bound into the idea that with high probability there will not be collisions between big Z_j.

Proof. Let

$$Q_i = \begin{cases} 1 & |z_i| > T/\ell \\ 0 & \text{else} \end{cases}$$

so that the number of indices with $|Z_i| > T/\ell$ equals $\sum Q_i$. We then have

$$\begin{aligned}
E\left(\sum_i Q_i\right) &= \sum_i E(Q_i) \\
&= \sum_i P\left(|x_i/u_i^{1/p}| > T/\ell\right) \\
&= \sum_i P\left(u_i < |x_i|^p \ell^p / T^p\right) \\
&= \sum_i (1 - e^{-|x_i|^p \ell^p / T_p}) && (u_i \text{ exponentially distributed}) \\
&\leq \sum_i \ell^p |x_i|^p / T^p && (1 + x \leq e^x \text{ for } x \in \mathbb{R}) \\
&= \ell^p && \textstyle\sum_i |x_i|^p = \|x\|_p^p = T^p
\end{aligned}$$

which completes the proof. □

2.2.2 Recalling Bernstein's inequality

To analyze the light indices, we'll need to recall *Bernstein's inequality.*

Theorem 1 (Bernstein's inequality)**.** *Suppose $R_1, \ldots, R_n$ are independent, and for all i, $|R_i| \leq K$, and var$(\sum_i R_i) = \sigma^2$. Then for all $t > 0$*

$$P\left(\left|\sum_i R_i - E\left(\sum_i R_i\right)\right| > t\right) \lesssim e^{-ct^2/\sigma^2} + e^{-ct/K}$$

2.2.3 Analyzing the light indices

We now establish that the light indices together will not distort any of the heavy indices by too much. Before we write down our specific claim, let's parametrize P as follows. We have a function $h : [n] \to [m]$ as well as a function $\sigma : [n] \to \{-1, 1\}$ that were both chosen at random. (One can show that these can be chosen to be k-wise independent hash functions, but we won't do so in this lecture.) We then write

$$P_{ij} = \begin{cases} \sigma(j) & \text{if } h(j) = i \\ 0 & \text{else.} \end{cases}$$

So essentially, h tells us which element of the column to make non-zero, and σ tells us which sign to use for column j.

We can now write our claim about the light indices.

Claim 3. *It holds with constant probability that for all $j \in [m]$,*

$$\left|\sum_{j \in L : h(j) = i} \sigma(j) Z_j\right| < T/10.$$

Let us see how this claim completes our argument. It means that

- If y_i didn't get any heavy indices then the magnitude of y_i is much less than T, so it won't interfere with our estimate.
- If y_i got assigned the maximal Z_j, then by our previous claim that is the only heavy index assigned to y_i. In that case, this claim means that all the light indices assigned to y_i won't change it by more than $T/10$, and since Z_j is within a factor of 2 of T, y_i will still be within a constant multiplicative factor of T.
- If y_i got assigned some other heavy index, then the corresponding Z_j is by definition is less than $2T$ since it's less than the maximal Z_j. In that case, this claim again tells us that y_i will be at most $2.1T$.

To put this more formally:

$$\begin{aligned} y_i &= \sum_{j : h(j) = i} \sigma(j) Z_j \\ &= \sum_{j \in L : h(j) = i} \sigma_(j) Z_j + \sigma(j_{heavy}) Z_{j_{heavy}} \end{aligned}$$

where the second term is added only if y_i got some heavy index, in which case we can assume it received at most one. The triangle inequality then implies that

$$\begin{aligned} |y_i| &\in Z_{j_{heavy}} \pm \left| \sum_{j \in L : h(j) = i} \sigma_(j) Z_j \right| \\ &= Z_{j_{heavy}} \pm T/10 \end{aligned}$$

Applying this to the bucket that got the maximal z_i then gives that that bucket of y should contain at least $0.4T$. And applying this to all other buckets gives that they should contain at most $2.1T$.

Let us now prove the claim.

Proof of Claim 3. Fix $i \in [m]$. We use Bernstein on the sum in question. For $j \in L$, define

$$\delta_j = \begin{cases} 1 & \text{if } h(j) = i \\ 0 & \text{else.} \end{cases}$$

Then the sum we seek to bound equals

$$\sum_{j \in L} \delta_j \sigma(j) Z_j$$

We will call the j-th term of the summand R_j and then use Bernstein's inequality. The brunt of the proof will be computing the relevant quantities to see what the inequality gives us. First, the easy ones:

1. We have $E(\sum R_j) = 0$, since the $\sigma(j)$ represent random signs.
2. We also have $K = T/(v \lg(n))$ since $|\delta_j| \leq 1$, $|\sigma(j)| \leq 1$, and we only iterate over light indices so $|Z_j| \leq T/(v \lg(n))$.

It remains only to compute $\sigma^2 = \text{var}(\sum_j R_j)$. If we condition on Z, then a problem from problem set 1 implies that

$$\text{var}\left(\sum_j R_j | Z \right) \leq \frac{\|Z\|_2^2}{m}$$

This isn't enough of course: we need to get something that takes the randomness of Z into account. However, instead of computing the unconditional variance of our sum, we will prove that σ^2 is small with high probability over the choice of Z. We'll do this by computing the unconditional expectation of σ^2 and then using Markov. We write

$$E\left(\|Z\|_2^2 \right) = \sum_j x_j^2 E\left(\frac{1}{u_j^{2/p}} \right)$$

and

$$\begin{aligned}
E\left(\frac{1}{u_j^{2/p}}\right) &= \int_0^\infty e^{-x}(x^{-2/p})dx \\
&= \int_0^1 e^{-x}(x^{-2/p})dx + \int_1^\infty e^{-x}\cdot(x^{-2/p})dx \\
&= \int_0^1 x^{-2/p}dx + \int_1^\infty e^{-x}dx. && \text{(trivial bounds on } e^{-x} \text{ and } x^{-2/p})
\end{aligned}$$

The second integral trivially converges, and the former one converges because $p > 2$. This gives that

$$E(\|Z\|^2) = O(\|x\|_2^2)$$

which gives that with high probability we will have $\sigma^2 \leq O(\|x\|_2^2)/m$.

To use Bernstein's inequality, we'll want to relate this bound on σ^2, which is currently stated in terms of $\|x\|_2$, to a bound in terms of $\|x\|_p$. We will do this using a standard argument based on Hölder's inequality, which we re-state without proof below.

Theorem 2 (Hölder's inequality)**.** *Let $f, g \in \mathbb{R}^n$. Then*

$$\sum_i f_i g_i \leq \|f\|_a \|g\|_b$$

for any $1 \leq a, b \leq \infty$ satisfying $1/a + 1/b = 1$.

Setting $f_i = x_i^2$, $g_i = 1$, $a = p/2$, $b = 1/(1-a)$ then gives

$$\begin{aligned}
\|x_i\|^2 &= \sum_i f_i g_i \\
&\leq \left(\sum_i (x_i^2)^{p/2}\right)^{2/p} \left(\sum_i 1^{1/(1-2/p)}\right)^{1-2/p} && \text{(Hölder)} \\
&\leq \|x\|_p^2 \cdot n^{1-2/p}
\end{aligned}$$

Using the fact that we chose m to $\Theta(n^{1-2/p}\lg(n))$, we can then obtain the following bound on σ^2 with high probability.

$$\begin{aligned}
\sigma^2 &\leq O\left(\frac{\|x\|_2^2}{m}\right) \\
&\leq O\left(\frac{T^2 n^{1-2/p}}{m}\right) && \text{(Hölder trick)} \\
&\leq O\left(\frac{T^2 n^{1-2/p}}{n^{1-2/p}\lg n}\right) && \text{(choice of } m) \\
&\leq O\left(\frac{T^2}{\lg(n)}\right)
\end{aligned}$$

We now need to apply Bernstein's inequality and show that it gives us the desired result. Initially, the inequality gives us the following guarantee.

$$\begin{aligned} P\left(\left|\sum R_i\right| > T/10\right) &\lesssim e^{-cT^2/100 \cdot O(\lg(n)/T^2)} + e^{-cT/10 \cdot (v \lg(n)/T)} \\ &\leq e^{-C \lg(n)} && \text{(for some new constant } C\text{)} \\ &= n^C \end{aligned}$$

So the probability that the noise at most $T/10$ can be made poly n. But there are at most n buckets, which means that a union bound gives us that with at least constant probability all of the light index contributions are are at most $T/10$. □

3 Wrap-up

Thus far we presented algorithms for p-norm estimation for $p \leq 2$, $p = 2$, and $p > 2$ separately. (Of course, the $p \leq 2$ can be used for $p = 2$ as well.) We noticed that at $p = 2$ there seems to be a critical point above which we appeared to need a different algorithm. Later in the course we'll see that there are space lower-bounds that say that once $p > 2$ we really do need as much space as the algorithm we presented for $p > 2$ required.

We conclude our current treatment of norm estimation and approximate counting by briefly noting some motivating applications for these problems. For example, distinct elements is used in SQL to efficiently count distinct entries in some column of a data table. It's also used in network anomaly detection to, say, track the rate at which a worm is spreading: you run distinct elements on a router to count how many distinct entities are sending packets with the worm signature through your router. Another example is: how many distinct people visited a website? For more general moment estimation, there are other motivating examples as well. Imagine x_i is the number of packets sent to IP address i. Estimating $\|x\|_\infty$ would give an approximation to the highest load experienced by any server. Of course, as we just mentioned, $\|x\|_\infty$ is difficult to approximate in small space, so in practice people settle for the closest possible norm to the ∞-norm, which is the 2-norm. And they do in fact use the 2-norm algorithm developed in the problem set for this task.

4 Some setup for next time

Next time we'll talk about two new, related problems that sites like Google trends solve. They are called the heavy hitters problem and the point query problem.

In Point Query, we're given some $x \in \mathbb{R}^n$ updated in a turnstile model, with n large. (You might imagine, for instance, that x has a coordinate for each string your search engine could see and x_i is the number of times you've seen string i.) We seek a function query(i) that, for $i \in [n]$, returns a value in $x_i \pm \varepsilon \cdot \|x\|_1$.

In Heavy Hitters, we have the same x but we seek to compute a set $L \subset [n]$ such that

1. $|x_i| \geq \varepsilon \|x\|_1 \Rightarrow i \in L$
2. $|x_i| < \frac{\varepsilon}{2} \|x\|_1 \Rightarrow i \notin L$

As an observation: if we can solve Point Query with bounded space then we can solve Heavy Hitters with bounded space as well (though not necessarily efficient run-time). To do this, we just run Point Query with $\varepsilon/10$ on each $i \in [n]$ and output the set of indices i for which we had large estimates of x_i.

4.1 Deterministic solution to Point Query

Let us begin a more detailed discussion of Point Query. We begin by defining an *incoherent matrix*.

Definition 1. $\Pi \in \mathbb{R}^{m \times n}$ *is* ε-incoherent *if*

1. *For all* i, $\|\Pi_i\|_2 = 1$
2. *For all* $i \neq j$, $|\langle \Pi_i, \Pi_j \rangle| \leq \varepsilon$.

We also define a related object: a *code*.

Definition 2. *An* (ε, t, q, N)*-code is a set* $\mathcal{C} = \{C_1, \ldots, C_N\} \subseteq [q]^t$ *such that for all* $i \neq j$, $\Delta(C_i, C_j) \geq (1 - \varepsilon)t$*, where* Δ *indicates Hamming distance.*

The key property of a code can be summarized verbally: any two distinct words in the code agree in at most εt entries.

There is a relationship between incoherent matrices and codes.

Claim 4. *Existence of an* (ε, t, q, n)*-code implies existence of an* ε*-incoherent* Π *with* $m = qt$.

Proof. We construct Π from $\mathcal{C}$. We have a column of Π for each $C_i \in \mathcal{C}$, and we break each column vector into t blocks, each of size q. Then, the j-th block contains binary string of length q whose a-th bit is 1 if the j-th element of C_i is a and 0 otherwise. Scaling the whole matrix by $1/\sqrt{t}$ gives the desired result. □

We'll start next time by showing the following two claims.

Claim 5 (to be shown next time)**.** *Given an* ε*-incoherent matrix, we can create a linear sketch to solve Point Query.*

Claim 6 (shown next time)**.** *A random code with* $q = O(1/\varepsilon)$ *and* $t = O(\frac{1}{\varepsilon} \log N)$ *is an* (ε, t, q, N)*-code.*

References

[Andoni12] Alexandr Andoni. High frequency moments via max-stability. Manuscript, 2012. `http://web.mit.edu/andoni/www/papers/fkStable.pdf`

CS 229r: Algorithms for Big Data Fall 2015

Lecture 6 — September 22, 2015

Prof. Jelani Nelson *Scribe: Brabeeba Wang*

1 Overview

In the last lecture we

1. l_1 point query: $query(i) = x_i \pm \varepsilon\|x\|_1$
2. l_1 heavy hitters: $query()$ return $L \in [n]$ such that
   (1) $|x_i| > \varepsilon\|x\|_1 \to i \in L$
   (2) $|x_i| < \varepsilon\|x\|_1/2 \to i \notin L$

In this lecture we are going to cover few algorithms on point query and heavy hitters

Definition 1. $\Pi \in \mathbb{R}^{m\times n}$ *is* ε*-coherent if*

1. *For any* i, $\|\Pi_i\|_2 = 1$
2. *for any* $i \neq j$, $|\langle \Pi_i, \Pi_j\rangle| < \varepsilon$

Claim 2. $\Pi \in \mathbb{R}^{m\times n}$ *is* ε*-coherent* $\to$ m *dimensional sketch for* l_1 *point query [2]*

Proof.
1. sketch is $y = \Pi x$
2. estimate x_i as $(\Pi^Y y)_i = (\Pi^Y \Pi x)_i$
3. $(\Pi^Y \Pi x)_i = e_i^T \Pi^T \Pi x =< \Pi, \sum_j \Pi_j x_j >= x_i + \sum_{j\neq i} z_j \langle \Pi_i, \Pi_j\rangle \leq x_i + \sum \varepsilon |x_j|$

□

Given Πx, recover $x' = \Pi^t \Pi x$ such that $\|x - x'\|_\infty \leq \varepsilon\|x\|_1$ is called l_∞/l_1 guarantee

2 Incoherent Π (Construction)

Using (ε, t, q, n) code, we get $\Pi \in \mathbb{R}^{m\times n}$, m = qt ε-incoherent. We have $q = 2/\varepsilon$, $t = \Theta(\log n/\varepsilon)$ random such code works why?

Proof. Two deterministic codes are following.

1. Look at two codewords C, D. We have $\mathbb{E}(\#$ indices where C, D agree$) = t/q$. By Chernoff, we have the probability of $\leq 2t/q = \varepsilon t$ w.p $\geq 1 - e^{-\Omega(t/q)}$

2. **Real-Soloman codes**
   $q = t = \Theta(\varepsilon^{-1} \lg n / \lg \lg n + \lg(1/\varepsilon)) \to m = \Theta(\varepsilon^{-2}(\log n / \log \log n + \lg(1/\varepsilon))^2)$. Better than 1 when $\varepsilon << 2^{-c \lg n^{1/2}}$

□

Fact 3. *$m \geq 1/\varepsilon^2 \cdot \lg n / \lg(1/\varepsilon)$ for any ε-incoherent Π*

3 Randomized Point Query

Definition 4. *CountMin (CM) sketch [1]*

1. *Hashing $h_1, ...h_L : [n] \to [t]$ (will come from 2-wise family)*
2. *counters $C_{a,b}$ for $a \in [L]$, $b \in [t]$*
3. *$C_{a,b} = \sum_{i \in [n], h_a(i) = b} x_i$*
4. *for ε-point query with failure probability δ, set $t = 2/\varepsilon, L = \lg(1/\delta)$.*
   And let query(i) output $\min_{i \leq r \leq L} C_{r,h_r(i)}$ (assuming "strict turnstile", for any i, $x_i \geq 0$).

Claim 5. *query$(i) = x_i \pm \varepsilon \|x\|_1$ w.p $\geq 1 - \delta$. $m = O(\varepsilon^{-1} \lg(1/\delta))$*

Proof. CM sketch

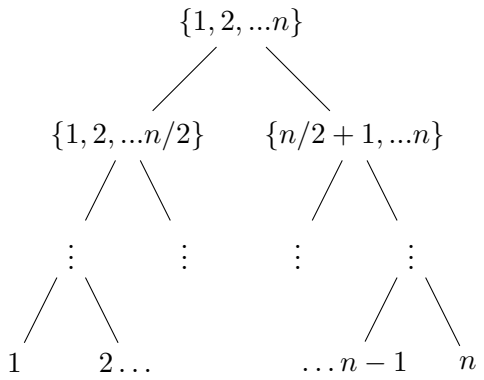
1. Fix i, let $Z_j = 1$ if $h_r(j) = h_r(i)$, $Z_j = 0$ otherwise. $C_{r,h_r(i)} = x_i + \sum_{j \neq i} x_j Z_j$ error E.
2. We have $\mathbb{E}(E) = \sum_{j \neq i} |x_j| \mathbb{E} Z_j = \sum_{j \neq i} |x_j| / t \leq \varepsilon/2 \cdot \|x\|_1$
3. $\mathbb{P}(E > \varepsilon \|x\|_1) < 1/2$
4. $\mathbb{P}(\min_r C_{r,h_r(i)} > x_i + \varepsilon \|x\|_1) < 1/2^L = \delta$

□

Theorem 6. *There is an α-Heavy Hitter (strict turnstile) w.p $1 - \eta$*

Proof. Naively, we can do point query repeatedly with $\varepsilon = \alpha/4, \delta = \eta/n \to m = O(1/\alpha \log(n/\eta))$ with query time $O(n \cdot \log(n/\eta))$.

But we have a quicker way, consider a perfect binary tree using our n vector elements as the leaves.



There are $\lg n$ levels and the weight of each node is the sum of elements. Now for each levels consider a $CountMin$ algorithm.

Now our algorithm is:

- Run CountMin from the roots downward with error $\epsilon = \alpha/4$ and $\delta = \eta\alpha/4 \log n$
- Move down the tree starting from the root. For each node, run CountMin for each of its two children. If a child is a heavy hitter, i.e. CountMin returns $\geq 3\alpha/4\|x\|_1$, continue moving down that branch of the tree.
- Add to L any leaf of the tree that you point query and that has $CM(i) \geq 3\alpha/4\|x\|_1$.

Correctness:

- Notice that l_1 norm will be the same at every level since the weight of the parents node is exactly the sum of children nodes.
- Also notice that node u contains heavy hitter amongst leaves in its subtree $\rightarrow$ u is hit at its level.
- Notice that there is at most $2/\alpha$ nodes at any given level which are $\alpha/2$-heavy hitter at that level.
- This implies that if all point queries correct, we only touch at most $(2/\alpha)\lg n$ vertices during BFS
- For each CM_j, we have $\epsilon = \alpha/4, \delta = \eta\alpha/4 \log n \rightarrow space(CM_j) = O(1/\alpha \cdot \log(\log n/\alpha\eta)) \rightarrow totalSpace = O(1/\alpha \cdot \log n \cdot \log(\log n/\alpha\eta))$

□

We know heavy hitter is l_∞/l_1 guarantee. We will see later something called compressed sensing that gets l_1/l_1. To be precise $\|x - x'\|_1 \leq (1+\varepsilon)\|x_{tail(k)}\|_1$. The question is can you get to l_1/l_1 for HH. CM sketch can give this with $\|x'\|_0 \leq k$

Definition 7. *$x_{tail(k)}$ is x but with the heaviest k coordinates in magnitude zero'd out.*

Claim 8. *If CM has* $t \geq \Theta(k/\varepsilon)$, $L = \Theta(\lg(1/\delta))$ *then w.p.* $1-\delta$, $x'_i = x_i \pm \varepsilon/k\|x_{tail(k)}\|_1$

Given x' from CM output ($x'_i = query(i)$). Let $T \subset [n]$ correspond to largest k entries of x' in magnitude. Now consider $y = x'_T$.

Claim 9. $\|x-y\|_1 \leq (1+3\varepsilon)\|x_{tail(k)}\|_1$

Proof. Let S denote $head(x) \subset [n]$ and T denote $head(x') \subset [n]$. We have

$$\begin{aligned}
\|x-y\|_1 &= \|x\|_1 - \|x_T\|_1 + \|x_T - y_T\|_1 \\
&\leq \|x\|_1 + \|x_T - y_T + y_T\|_1 + \|x_T - y_T\|_1 \\
&\leq \|x\|_1 - \|y_T\|_1 + 2\|x_T - y_T\|_1 \\
&\leq \|x\| - \|y_S\| + 2\|x_T - y_T\|_1 \\
&\leq \|x\| - \|x_S\| + \|x_S - y_S\|_1 + 2\|x_T - y_T\|_1 \\
&\leq \|x_{tail(k)}\|_1 + 3\varepsilon\|x_{tail(k)}\|_1
\end{aligned}$$

□

References

[1] Graham Cormode, S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75 , 2005.

[2] Jelani Nelson, Huy L. Nguyen, David P. Woodruff. On Deterministic Sketching and Streaming for Sparse Recovery and Norm Estimation. *Linear Algebra and its Applications, Special Issue on Sparse Approximate Solution of Linear Systems*, 441: 152–167, January 15, 2014.

CS 229r: Algorithms for Big Data Fall 2015

Lecture 7 — September 24, 2015

Prof. Jelani Nelson *Scribe: David Mende*

1 Recap

In the last lecture we looked at ℓ_1 point query and heavy hitters. For point query we have:

$$\text{query}(i) = x_i \pm \varepsilon \|x\|_1 .$$

What about other norms?

1.1 Count Sketch

See [1] for more information. We are given $\sigma_1, \ldots, \sigma_L$ where $\sigma_j : [n] \to \{-1, 1\}$. We have

$$C_{r,h_r(i)} = \sigma_r(i)x_i + \underbrace{\left(\sum_{\substack{h_r(j)=h_r(i) \\ j \neq i}} \sigma_r(j)x_j \right)}_{\beta} .$$

For the error term β, we have that

$$\mathbb{E}[\beta^2] \leq \frac{1}{t} \|x\|_2^2 \implies |\beta| \leq \frac{\sqrt{3}}{\sqrt{t}} \|x\|_2$$

with probability $\geq \frac{2}{3}$.

We want to find $\tilde{x}$ such that

$$\|x - \tilde{x}\|_2^2 \leq (1 + \varepsilon) \left\|x_{\text{tail}(k)}\right\|_2^2 .$$

Gilbert et al. [2] showed that

$$M \lesssim \frac{k}{\varepsilon} \lg\left(\frac{n}{k}\right) .$$

We can recover $\tilde{x}$ in time $(k/\varepsilon) \log^{O(1)} n$ and process updates in $\lg^{O(1)} n$.

1.2 New topics

- ℓ_0-sampling
- Graph algorithms
  - Connectivity
  - k-connectivity

2 ℓ_p-sampling

See [3] for more information. We have that:

- x is an updated turnstile,
- Want to draw $i \in [n]$ with probability $i = j$ being $\frac{|x_j|^p}{\|x_j\|_p^p}$.
- The norm "$\|x\|_0^0$"$= |\text{supp}(x)| = |\{i : x_i \neq 0\}|$.
- The norm $\|x\|_p^p = \sum_{i=1}^n |x_i|^p$.

2.1 ℓ_0-sampling

See [4] for more information.

$$\mathbb{P}(i = j) = \begin{cases} 0, & \text{if } j \notin \text{supp}(x), \\ \frac{1}{\text{supp}(x)} \pm \delta, & \text{if } j \in \text{supp}(x). \end{cases}$$

Ingredients of Algorithm

(a) Show if y is 1-sparse, we can recover y with probability 1 using $O(\lg \frac{1}{\delta})$ rows. (This algorithm is deterministic).

(b) Also, if $|\text{supp}(y)| \neq 1$, output "Fail" with probability $\geq 1 - \delta$ using $O(\lg \frac{1}{\delta})$ rows.

(c) Geometric sampling.

Analysis

(a) $A = \sum_{i=1}^n y_i$, $B = \sum_{i=1}^n i \cdot y_i$,

$\|y\|_0 = 1 \implies \text{supp}(y) = \left\{ \frac{B}{A} \right\}, y_i = A.$

(b) Detect $|\text{supp}(y)| = 0$ (i.e., $y = 0$) by AMS sketch ($y = 0 \iff \|y\|_2 = 0$).

(c) Detect $|\text{supp}(y)| > 1$.

(1) Let y' be returned from (a). Test whether $\Pi(y' - y) = 0$.

(2) $h : [n] \to ?$
if $|\text{supp}(y)| > 2$ (contains $i_1, i_2, \ldots$).

Combine (a) and (b) using geometric sampling to get ℓ_0-sample. Create $\lg n$ virtual streams with vectors $y^{(0)}, \ldots, y^{(\lg n)}$. For

$$h : [n] \to \{0, \ldots, \lg n\}, \qquad \mathbb{P}(h(i) = j) = \frac{1}{2^{j+1}}.$$

Include index i in $y^{(h(i))}$. $y^{(j)} = x_{\{i:h(i)=j\}}$. We have that $1 \leq |\operatorname{supp}(x)| \leq n$ and

$$\mathbb{E}\left[\left|\operatorname{supp}(y^{(j)})\right|\right] = \frac{|\operatorname{supp}(x)|}{2^{j+1}}, 0 \leq j \leq \lg n.$$

This implies that

$$\exists\, j^* \text{ such that } 1 \leq \mathbb{E}\left[\left|\operatorname{supp}(y^{(j)})\right|\right] \leq 2,$$

Now we want to show that for this j^*,

$$\mathbb{P}\left(\left|\operatorname{supp}(y^{(j^*)})\right| = 1\right)$$

is large (i.e. $\Omega(1)$). Let $T = |\operatorname{supp}(x)|$. So $T/2 \leq 2^{j^*+1} \leq T$. Suppose T items are each kept independently with probability $2^{-(j^*-1)} \approx \frac{1}{T}$. What is the probability that *exactly* one item is kept? We have that

$$\begin{aligned}
\mathbb{P}(\text{exactly one survives}) &= \sum_{i=1}^{T} \mathbb{P}\left(\text{item } i \text{ survives and no one else dies}\right) \\
&= \left(\sum_{i=1}^{T} \frac{1}{T}\right) \cdot \left(1 - \frac{1}{T}\right)^{T-1} \\
&= \left(1 - \frac{1}{T}\right)^{T-1} \\
&\approx \frac{1}{e}\left(\frac{1}{1 - \frac{1}{T}}\right) \\
&\approx \frac{1}{e} \\
&= \Theta(1).
\end{aligned}$$

Final Space We have $\lg n \cdot \lg^2 \frac{1}{\delta}$ counters. After Nisan, we need $\lg^2 n \cdot \lg^2 \frac{1}{\delta}$ counters. It is known that we can achieve $O(\lg n \cdot \lg \frac{1}{\delta})$ words [5]. Instead of using 1-sparse y, Jowhari et al. used s-sparse y where $s = \Theta(\lg \frac{1}{\delta})$. You can recover y with probability 1 using $2s$ rows (for example, using Prony's method).

3 Graphs

Let $G = (V, E)$, where we see edges $e \in E$ in stream. Let $|V| = n$ and $|E| = m$.

3.1 Connectivity

Define

$$\text{query}(i,j) = \begin{cases} 1, & \text{if } i,j \text{ in same connected component,} \\ 0, & \text{else.} \end{cases}$$

Insertion Only

Straightforward: $O(m)$ space by storing all edges.

Straightforward−−: $O(n)$ space. Store a spanning forest.

Claim 1. *Any deterministic algorithm needs $\Omega(n)$ space.*

Proof. Suppose we have $x \in \{0,1\}^{n-1}$. As before, we will perform an encoding argument. We create a graph with n vertices $0, 1, \ldots, n-1$. The only edges that exist are as follows: for each i such that $x_i = 1$, we create an edge from vertex 0 to vertex i. The encoding of x is then the space contents of the connectivity streaming algorithm run on the edges of this graph. Then in decoding, by querying connectivity between 0 and i for each i, we can determine whether x_i is 1 or 0. Thus the space of the algorithm must be at least $n-1$, the minimum encoding length for compressing $\{0,1\}^{n-1}$. □

As an exercise, try to extend the $\Omega(n)$ lower bound above to randomized algorithms with constant failure probability (try using an approach similar to that for problem set 1, problem 3(b)).

For many interesting graph problems, it turns out that $\Omega(n)$ space is required. This motivated the "Semi-streaming" model for graphs [6], where the goal is to achieve want $O(n \lg^c n)$ space.

Question: Can we solve connectivity in turnstile model?

3.1.1 Graph Sketching

Now we will investigate the case of the turnstile model, where edges can be inserted *and* deleted in a stream. For example, users (vertices) on Facebook can friend each other (i.e. create an edge between them) then later one can defriend the other (delete the edge). We will show a sketching approach to handle turnstile streaming for graphs due to [7].

First, consider the following non-streaming algorithm.

Algorithm: ConnComp($G = (V, E)$).

- $S \leftarrow \{\{v_1\}, \{v_2\}, \ldots, \{v_n\}\}$
- For $i = 1$ to $\lg n$:
  - For each $s \in S$ (in parallel)

    - Pick an edge $e = (s, w)$ for some $w \in S$.
    - contract(e)
- Return S.

Turnstile streaming implementation:

- Each $v \in V$ will store $f(v) \in \mathbb{R}^{\binom{n}{2}}$.

$$(f(v))_{(a,b)} = \begin{cases} 0, & \text{if } v \notin \{a, b\} \text{ or } (a, b) \notin E, \\ +1, & \text{if } v = \min(a, b), \\ -1, & \text{if } v = \max(a, b). \end{cases}$$

- For $s \in S$, $f(s) = \sum_{v \in S} f(v)$.
- supp($f(s)$) is the set of edges with one endpoint in s and the other outside s.
- Use $\lg n$ different ℓ_0-sampler sketches $A_1(f(v)), \ldots, A_{\lg n}(f(v))$.

For each iteration i through the main loop in ConnComp, for each $s \in S$we use the ℓ_0-sampler $A_i(f(\cdot))$ to sample an edge e_s leaving s. Then, for each edge e obtained, we contract e between some supernodes s, t in the ConnComp algorithm. We perform this contraction by adding the sketches $A_j(f(s))$ and $A_j(f(t))$ for all $j > i$.

References

[1] Moses Charikar, Kevin C. Chen, Martin Farach-Colton. Finding Frequent Items in Data Streams. ICALP 2002: 693–703.

[2] Anna C. Gilbert, Yi Li, Ely Porat, Martin J. Strauss. Approximate Sparse Recovery: Optimizing Time and Measurements. *SIAM J. Comput.*, 41(2): 436–453 (2012).

[3] Morteza Monemizadeh, David P. Woodruff. 1-Pass Relative-Error L_p-Sampling with Applications. SODA 2010: 1143–1160.

[4] Graham Cormode, Donatella Firmani. A unifying framework for ℓ_0 sampling algorithms. *Distributed and Parallel Databases* 32(3): 315–335 (2014).

[5] Hossein Jowhari, Mert Saglam, Gábor Tardos. Tight bounds for L_p samplers, finding duplicates in streams, and related problems. PODS 2011: 49–58.

[6] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, Jian Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci*, 348(2–3): 207–216 (2005).

[7] Kook Jin Ahn, Sudipto Guha, Andrew McGregor. Analyzing graph structure via linear measurements. SODA 2012: 459–467.

CS 229r: Algorithms for Big Data Fall 2015

Lecture 8 — September 29, 2015

Prof. Jelani Nelson *Scribe: Johnny Ho*

1 Overview

In this lecture we introduce the longest increasing subsequence (LIS) and distance to monotonicity (DTM) problems along with various recent bounds and algorithms. We discuss and prove one particular randomized approximate algorithm for DTM.

2 Distance to Monotonicity

We introduce the idea of dynamic programming (DP) on streams. In particular, the longest increasing subsequence (LIS) problem is: Given an input sequence $x_1, x_2, ...x_n \in [m]$, find the longest possible increasing subsequence $x_{i_1} \leq x_{i_2} \leq x_{i_3} \leq ...x_{i_k}$.

The distance to monotonicity (DTM) problem is similar, except removing the minimal number of elements to obtain such an increasing subsequence. Then clearly $DTM = n - |LIS|$.

2.1 Distinction

These two problems are actually distinct in terms of approximation. If we have some estimate $\widetilde{LIS}$ of the LIS, then we can imagine estimating $\widetilde{DTM} = n - \widetilde{LIS} = n - (1 \pm \varepsilon)LIS = DTM \pm \varepsilon LIS$. This is, however, not a multiplicative error but rather an additive error in terms of n.

2.2 Recent results

In 2007, Gopalan, Jayram, Krauthgamer, Kumar [3] demonstrated a deterministic approximate algorithm for LIS in $O(\sqrt{n/\varepsilon})$ space. In 2008, Ergun, Jowhari [1] and in 2007, Gal, Gopalon [3], both demonstrate that this is a lower bound in terms of space required. It is open whether a poly-log algorithm exists for approximating the LIS problem.

The DTM problem has such an algorithm, published by Saks, Seshadhri in 2013 [5], using $O(1/\varepsilon \log(n/\varepsilon) \log(n))$ words of space. Subsequently, Naumovitz, Saks in 2015 [4] published lower bounds of $\Omega(1/\varepsilon \log^2 n/\log\log n)$ and $\Omega(1/\varepsilon \log^2 n)$ for randomized and determinstic algorithms, respectively. They also showed a deterministic algorithm using space bounded by $O(1/\varepsilon^2 \log^5 n)$.

3 Algorithm

We will present and prove Saks, Sehadri in this lecture, targeting a success probability of $1-\delta$ and the above space bound.

3.1 Presentation

Imagine $w(i)$ as the weights of the items i.e. the cost of removal, with $w(i) = 1$ normally.

We construct a DP table $s(i) = DTM(x_0, ..., x_i)$ such that we are forced to include (not remove) item i. As in dynamic programming, this will be iteratively constructed for each i based on previous i. Note that this index starts at zero because we will add sentinels at indices 0 and $n+1$.

Another auxillary DP table will be $W(t) = \sum_{i=1}^{t} w(i)$, i.e. the prefix sums of our weights.

We will also keep a set R.

Steps:

1. Give each item weight $w(i) = 1$
2. Prepend/postpend sentinels $-\infty/+\infty$ with weight ∞
3. Initialize $R = \{0\}$, $W(0) = 0$, $s(0) = 0$
4. For t = 1 to n+1:
   a. $W(t) = W(t-1) + w(t)$
   b. $s(t) = \min_{i \in R,\, x_i \leq x_t} s(i) + W(t-1) - W(i)$
   c. $R = R \cup \{t\}$
5. Return $s(n+1)$

This is deterministic so far, and uses linear space. For the probabilistic algorithm, replace the symbol r instead of s, and also add another step:
d. For each item $i \in R$, remove i with probability $1 - p(i,t)$.

Note that if indices are being removed from R, then step b. will be taking the minimum of a subset of the indices, generating an overestimate of the true answer.

Denote R^t as R at after t-th iteration. Define a function q for the probability of being kept at t:

$$q(i,t) = \mathbb{P}(i \in R^t) = \min\left\{1, \frac{1+\varepsilon}{\varepsilon} \ln\left(\frac{4t^3}{\delta}\right) \frac{w(i)}{W(i,t)}\right\}$$

The intuition here is that the probability of remembering is inversely proportional to distance, since the algorithm will only fail if we forget the most recent members of the DP table.

Then if we define p in step d. as:

$$p(i,t) = \begin{cases} q(i,t)/q(i,t-1) & \text{if } i < t \\ 1 & \text{if } i = t \end{cases},$$

then $q(i,t) = \mathbb{P}(i \in R^t)$ will be as desired, since the probability of being kept at t is the product of the probabilities of being kept so far.

3.2 Proof

We now wish to prove:

(A) With probability $1 - \delta/2$, $r(n+1) \leq (1+e)s(n+1)$, since it can only be an overestimate.

(B) With probability $1 - \delta/2$, $\forall t$, $|R^t|$ is small i.e. poly-log in size.

If R^t ever exceeds a certain size, the program can just terminate, throwing an error value.

3.2.1 Proving (A)

Fix $C \subseteq [n]$ to be a particular optimal LIS.

Define $i \in [n]$ to be unsafe at time t if $(C \cap [i,t]) \cap R^t = \emptyset$. This is to say that we have removed everything in between $[i,t]$ from R. Further, let $U^t = \{i,\ i \text{ unsafe at time } t\}$, and $U = \bigcup_{t=1}^{n+1} U^t$ be the union of all such unsafe sets. This is the set of all elements that have ever been unsafe.

Safeness is not a monotonic property over i, i.e. i may be safe, then get removed from R, making it unsafe, but then another element of the LIS might be inserted back into R, making it safe again. However, given a fixed t, it is true that safeness is monotonic as i is decreasing, i.e. they can only be unsafe and then safe.

Lemma 1. $r(n+1) \leq W(\overline{C} \cup U)$

Proof. Here $\overline{C}$ is the complement of C. Define $\overline{C}_{\leq t} = \overline{C} \cap [t]$

By induction on $t \in C$, we will show that $r(t) \leq W(\overline{C}_{\leq t-1} \cup \bigcup_{k=1}^{t-1} U^k)$. As the base case, clearly $r(0) = 0$.

The inductive step has two cases:

- $C_{\leq t-1} \setminus U^{t-1} = \emptyset$

  We forgot everything in $C_{\leq t-1}$, so $W(\overline{C}_{t-1} \cup U^{t-1}) = W(1, t-1)$, and $r(t) \leq W(1, t-1)$ clearly.

- $C_{\leq t-1} \setminus U^{t-1} \neq \emptyset$

  Let W apply to intervals and sets as the sum of the interval/sets. Pick the largest safe $j \in C_{\leq t-1} \setminus U^{t-1}$. Then we have that $r(t) \leq r(j) + W(j+1, t-1) \leq W(\overline{C}_{\leq j-1} \cup \bigcup_{k=1}^{j-1} U^k) + W(j+1, t-1) = W(\overline{C}_{\leq t-1} \cup \bigcup_{k=1}^{t-1} U^k)$.

  Equality holds because everything in the second term of the sum must be either unsafe at time t or not in the LIS C at all.

□

Define interval $I \subseteq [n]$ dangerous if $|I \cap C| \leq \frac{\varepsilon}{1+\varepsilon}|I|$. Define any $i \in C$ dangerous if i is the left endpoint of some dangerous interval.

We greedily form a dangerous collection $I_1, I_2, ...$ as follows: First, let D be the set of dangerous $i \in C$. Take the leftmost element of D and extend to the right as far as possible, and then repeat, taking leftmost elements not already included. Note that the elements not in these intervals should be inside C. Let $B = \cup_j I_j$ be the union of these intervals.

We claim:

a. $\overline{C} \subseteq D \subseteq B$

b. $W(B) \leq (1+\varepsilon)W(\overline{C})$

c. $\mathbb{P}(U \subseteq B) \geq 1 - \delta/2$

To prove (A) from these claims, note that $r(n+1) \leq W(\overline{C} \cup U)$, and that $\leq W(\overline{C} \cup B)$ with probability $1 - \delta/2$, and this has weight equal to $W(B)$ since $\overline{C} \subseteq B$. $W(B) \leq (1+\varepsilon)W(\overline{C}) = (1+\varepsilon)s(n+1)$ and thus $r(n+1) \leq (1+\varepsilon)s(n+1)$, as desired.

Proof. Proof of claims:
a. First, $\overline{C} \subseteq D$ since $i \in \overline{C} \implies [i,t]$ dangerous $\implies i \in D$. Then $D \subseteq B$ by construction.

b.

$$\forall j,\, W(I_j \cap C) \leq \frac{\varepsilon}{1+\varepsilon} W(I_j) \implies \forall j,\, W(I_j \cap \overline{C}) \geq \frac{1}{1+\varepsilon} W(I_j)$$

Then, $W(B \cap \overline{C}) \geq 1/(1+e)W(B)$, and since $\overline{C} \subseteq B$, $(1+e)W(\overline{C}) \geq W(B)$ as desired.

c.

Lemma 2.

$$\forall t \in [n],\, \forall i \in \overline{B} \cap t,\, \mathbb{P}(i \in U^t) \leq \frac{\delta}{4t^3},$$

If this lemma were true, then by union bound,

$$\begin{aligned}
\mathbb{P}(U \subseteq B) &= 1 - \mathbb{P}(\overline{B} \cap U \neq \emptyset) \\
&= 1 - \mathbb{P}(\exists t,\, \exists i,\, i \in U^t) \\
&\geq 1 - \sum_t \sum_{i \in \overline{B} \cap [t]} P(i \in U^t) \\
&\geq 1 - \delta/4 \sum_t 1/t^2 \geq 1 - \delta/2,
\end{aligned}$$

as desired for Lemma 1. □

Proof. Proving Lemma 2: We know that i is not dangerous, and that $[i,t]$ is not dangerous, so thus $W(C \cap [i,t]) \geq \frac{\varepsilon}{1+\varepsilon} W(i,t)$. By definition $i \in U^t$ iff everything in $C \cap [i,t]$ has been forgotten at time t. Thus, substituting the first expression into q:

$$\mathbb{P}(i \in U^t) = \prod_{j \in C \cap [i,t]} (1 - q(j,t)) \leq \prod_{j \in C \cap [i,t]} \left(1 - \ln\left(\frac{4t^3}{\delta}\right) \frac{w(j)}{W(C \cap [j,t])}\right)$$

Substituting $1 - x \leq \mathrm{e}^{-x}$, we obtain the bound

$$\mathbb{P}(i \in U^t) \leq \prod_{j \in C \cap [i,t]} \mathrm{e}^{-\ln(\frac{4t^3}{\delta})\frac{w(j)}{W(C\cap[j,t])}} = \mathrm{e}^{-\ln(\frac{4t^3}{\delta})\left(\sum_{j\in C\cap[i,t]}\frac{w(j)}{W(C\cap[j,t])}\right)} = \frac{\delta}{4t^3}$$

□

3.2.2 Proving (B)

This is not as complicated. The space used during the algorithm is proportional to $\max_{t\in[n]}|R^t|$. Fixing t, we need to show that the probability that $|R^t|$ is large can be bounded by $\frac{\delta}{4t^2}$, so by union bound $\mathbb{P}(\exists t \text{ s.t. } |R^t| \text{ large}) < \frac{\delta}{2}$. Let Z_i^t be the indicator random variable for the event that $i \in R^t$, which is 1 with probability $q(i,t)$. Thus

$$|R^t| = \sum_{i \leq t} Z_i^t,$$

which implies

$$\mathbb{E}\,|R^t| = \mu_t = \Theta(\varepsilon^{-1}\log(t/\delta)\log t)$$

by our choice of $q(i,t)$. This is because the sum of $w(i)/W(i,t)$ for $1 \leq i \leq t$ in our definition of $q(i,t)$ is a Harmonic series $\sum_{k=1}^{t} 1/k$, and is thus $\Theta(\log t)$ (recall $w(i) = 1$ except for the sentinels, and thus $W(i,t) = t - i + 1$).

Using Chernoff bounds, we have

$$\mathbb{P}(|R^t| > 2\mu_t) < e^{-\Omega(\mu_t)} < \frac{\delta}{4t^2}.$$

Thus

$$\mathbb{P}(\exists t : |R^t| > 2\mu_t) < \frac{\delta}{4} \cdot \sum_{t=1}^{n} \frac{1}{t^2} < \frac{\delta}{2}$$

as desired.

References

[1] Funda Ergün, Hossein Jowhari. On distance to monotonicity and longest increasing subsequence of a data stream. *SODA*, 730-736, 2008.

[2] Anna Gál, Parikshit Gopalan. Lower Bounds on Streaming Algorithms for Approximating the Length of the Longest Increasing Subsequence. *SIAM J. Comput.*, 39(8):3463-3479, 2010.

[3] Parikshit Gopalan, T.S. Jayram, Robert Krauthgamer, Ravi Kumar. Estimating the sortedness of a data stream. *SODA*, 318-327, 2007.

[4] Timothy Naumovitz, Michael Saks. A polylogarithmic space deterministic streaming algorithm for approximating distance to monotonicity. *SODA*, 1252-1262, 2015.

[5] Michael Saks, C. Seshadri. Space efficient streaming algorithms for the distance to monotonicity and asymmetric edit distance. *SODA*, 1698-1709, 2013.

CS 229r: Algorithms for Big Data Fall 2015

Lecture Lecture 9 — October 1, 2015

Prof. Jelani Nelson *Scribe: Rachit Singh*

1 Overview

In the last lecture we covered the distance to monotonicity (DTM) and longest increasing subsequence (LIS) problems.

In this lecture we will talk about how to prove space lower bounds for a variety of problems using communication complexity.

2 Space lower bounds

We're going to see some sophisticated techniques to prove space lower bounds. These are all proved via something called **communication complexity**. The problems we're going to look at today are F_0 (distinct elements) - specifically any algorithm that solves F_0 within a factor of ϵ must use $\Omega(1/\epsilon^2 + \log n)$ bits. We're also going to discuss **median**, or randomized exact median, which requires $\Omega(n)$ space. Finally, we'll talk about F_p or $\|x\|_p$, which requires $\Omega(n^{1-2/p})$ space for a 2-approximation.

2.1 2 player communication complexity

Suppose we have Alice and Bob, and a function $f : X \times Y \to \{0, 1\}$. Alice gets $x \in X$, and Bob gets $y \in Y$. They want to compute $f(x, y)$. Suppose that Alice starts the conversation. Suppose she sends a message m_1 to Bob. Then Bob replies with m_2, and so on. After k iterations, someone can say that $f(x, y)$ is determined. The goal for us is to minimize the total amount of communcation, or $\sum_{i=1}^{k} |m_i|$, where the absolute value here refers to the length of the binary string.

A **communicaton protocol** is a way of conversing agreed upon ahead of time, where Alice and Bob both know f. There's obvious the two obvious protocols, where Alice sends $\log X$ bits to send x, or where Bob sends y via $\log Y$ bits to Alice. The goal is to either beat these trivial protocols or prove that none exists.

There's a natural connection between communicaton complexity and space lower bounds as follows: a communication complexity lower bound can yield a streaming lower bound. We'll restrict our attention to 1-way protocols, where Alice just sends messages to Bob. Suppose that we had a lower bound for a communication problem - Alice has $x \in X$, and Bob has $y \in Y$ and we know that the lower bound (LB) on the optimal communcation complexity is $\overrightarrow{D}(f)$. The D here refers to the fact that the communication protocol is deterministic. If there's a streaming problem, then Alice can run her streaming algorithm on x, the first half of the stream, and send the memory contents

across to Bob, who can then load it and pass y, the second half of the stream, and calculate $f(x, y)$, the final answer. So the minimal amount of space necessary is $\overrightarrow{D}(f)$.

2.2 F_0

Exact and deterministic F_0 requires $\Omega(n)$ space (we saw this in class via the compression argument, but we want to rephrase in the communication complexity argument). We'll use a reduction - if comm. complexity is hard, then the F_0 problem must also be hard, because otherwise we could use the above argument. We use the *equality problem* (EQ), which is where $f(x, y) = x == y$. We claim $D(EQ) = \omega(n)$. This is pretty simple to prove in the one-way protocol, by using the pigeonhole principle, as before.

We're going to reduce EQ to F_0. Suppose that there exists a streaming algorithm A for F_0 that uses S bits of space. Alice is going to run A on her stream x, and then send the memory contents to Bob. Bob then queries F_0, and then for each $i \in y$, he can append and query as before, and solve the equality problem. However, this solves EQ, which requires $\Omega(n)$ space, so S must be $\Omega(n)$. This is just a rephrasing of the earlier argument in terms of communication complexity.

Now, a few definitions:

- $D(f)$ is the optimal cost of a deterministic protocol
- $R_\delta^{\text{pub}}(f)$ is the optimal cost of the random protocol with failure probability δ such that there is a shared random string (written in the sky or something).
- $R_\delta^{\text{pri}}(f)$ is the same as above, but each of Alice/Bob have private random strings.
- $D_{\mu,s}(f)$ is the optimal cost of a deterministic protocol with failure probability δ where $(x, y) \sim \mu$.

Claim 1. $D(f) \geq R_\delta^{\text{pri}}(f) \geq R_\delta^{\text{pub}}(f) \geq D_{\mu,s}(f)$

Proof. The first inequality is clear, since we can just simulate the problem. The second inequality follows from the following scheme: Alice just uses the odd bits, and Bob just uses the even bits in the sky. The final inequality follows from an indexing argument: suppose that P is a public random protocol with a random string s, $\forall (x, y)\, \mathbb{P}(P_s\text{correct}) \geq 1 - \delta$. Then there exists an s^* such that the probability of P_s succeeding is large. Note that s^* depends on μ. □

If we want to do a lower bound on deterministic algorithms, we want to lower bound $D(f)$. If we want to do the lower bound of a randomized algorithm, we want to lower bound $R_\delta^{\text{pri}}(f)$. We need Alice to communicate the random bits over to Bob so that he can continue running the algorithm, and we need to *include* these bits in the cost since we store the bits in memory. So, to lower bound randomized algorithms, we lower bound $D_{\mu,s}(f)$.

If you want to learn more, you can read a book called *Communication Complexity* by Kushilevitz and Nisan [Kus06]. Fun fact: you can solve EQ using public randomness with constant number of bits. If you want to solve it using private randomness for EQ, you need $\log n$ bits. Alice picks

a random prime, and she sends $x \mod p$ and sends across $x \mod p$ and the prime. Neumann's theorem says that you can reverse the middle inequality in the above at a cost of $\log n$ (i.e. the LHS is smaller than $\log n$ times the RHS).

We're going to show that *INDEX*, the problem of finding the jth element of a streamed vector, is hard. Then, we'll show that this reduces to *GAPHAM*, or Gap Hamming which'll reduce to F_0. Also, INDEX reduces to *MEDIAN*. Finally, $DISJ_t$ reduces (with $t = (2n)^{1/p}$) to F_p, $p > 2$.

2.3 Index

INDEX is a two-player problem. Alice gets $x \in \{0,1\}^n$, and Bob gets $j \in [n]$, and $INDEX(x,j) = x_j$.

Claim 2. $R_\delta^{\text{pub}\rightarrow}(INDEX) \geq (1 - \mathfrak{H}_2(\delta))n$, *where* $\mathfrak{H}_2(\delta) = \delta\log(\delta) + (1-\delta)\log(1-\delta)$, *the entropy function. If* $\delta \approx 1/3$.

In fact, it's true that the distributional complexity has the same lower bound. The reason this is hard is because it's one way - Alice doesn't know which bit to send to Bob.

2.4 Information Theory crash course

Mostly definitions, but you can check out *Essentials of Information Theory* by Cover and Thomas for more details.

Definitions: if we have a random variable X, then

- $H(X) = \sum_x p_x \log(p_x)$ (*entropy*)
- $H(X,Y) = \sum_{(x,y)} p_{x,y} \log p_{x,y}$ (*joint entropy*)
- $H(X|Y) = \mathbb{E}_y(H(X|Y=y))$ (*conditional entropy*)
- $I(X,Y) = H(X) - H(X|Y)$ (*mutual information*) (note that this is a symmetric quantity)

The *entropy* is the amount of information or bits we need to send to communicate $x \in X$ in expectation. This can be achieved via Huffman coding (in the limit). The mutual information is how much of X we get by communicating Y.

Here are some basic lemmas involving these equalities

Lemma 3.

- *Chain rule:* $H(X,Y) = H(X) + H(Y|X)$
- *Chain rule for mutual information:* $I(X,Y|Z) = I(X,Z) + I(Y,Z|X)$
- *Subadditivity:* $H(X,Y) \leq H(X) + H(Y)$
- *Chain rule + subadditivity:* $H(X|Y) \leq H(X)$.

- *Basic* $H(X) \leq \log |\operatorname{supp}(X)|$.
- $H(f(X)) \leq H(X) \quad \forall f$ *(no free lunch)*

Theorem 4. ***Fano's Inequality***

Formally, if there exist two random variables X, Y *and a predictor* g *such that* $\mathbb{P}(g(Y) \neq X) \leq \delta$*, then* $H(X|Y) \leq H_2(\delta) + \delta \cdot \log_2(|\operatorname{supp}(X)| - 1)$.

Note that if X is a binary random variable then the second term vanishes. Intuitively, if all you have is Y, and based on Y you make a guess of X. Then if you're able to guess well, then they must be correlated in some way. Note that for small δ, $H_2(\delta) \approx \delta$. Now we'll go back to INDEX, and our earlier claim.

2.5 INDEX revisited

Let Π be the transcript of the optimal communication protocol. It's a one-way protocol here, so it's just what Alice said. So, we know that $R_\delta^{\text{pub}}(INDEX) \geq H(\Pi) \geq I(\Pi, \text{input}) = I(\Pi, \text{input})$

We know that for all x and for all j, $\mathbb{P}_s(\text{Bob is correct}) \geq 1 - \delta$, which implies that for all j, $\mathbb{P}_{X \sim \text{Unif}}\, \mathbb{P}_s(\text{Bob is correct}) \geq 1 - \delta$, which them implies that by Fano,

$$H(X_j|\Pi) \geq H_2(\delta)$$

Note that Π is a random variable because of the random string in the sky, and also because it is dependent on X.

Note that we have

$$\begin{aligned}
|\Pi| &\geq I(X; \Pi) \\
&= \sum_{i=1}^{n} I(X_i; \Pi | X_1, \dots X_{i-1}) \text{ (chain rule n times)} \\
&= \sum_i H(X_i|X^{<i}) - H(X_i|\Pi, X^{<i}) \\
&\geq \sum_i 1 - H_2(\delta) = n(1 - H_2(\delta))
\end{aligned}$$

Now that we have INDEX, let's use it to prove another lower bound, namely MEDIAN. We want a randomized, exact median of $x_1, \dots x_n$ with probability $1 - \delta$. We'll use a reduction (see [GM09]).

Claim: INDEX on $\{0,1\}^n$ reduces to MEDIAN with $m = 2n + 2$, with string length $2n - 1$. To solve INDEX, Alice inserts $2 + x_1, 4 + x_2, 6 + x_3 \dots$ into the stream, and Bob inserts $n - j$ copies of 0, and another $j - 1$ copies of $2n + 2$.

Suppose that $n = 3$ and $x = 101_2$. Then Alice will choose $3, 4, 7$ out of $2, 3, 4, 5, 6, 7$. Bob cares about a particular index, suppose the first index. Bob is going to make this stream length 5, such that the median of the stream is exactly the index he wants. Basically, we can insert 0 or $2n + 2$ exactly where we want, moving around the j index to be the middle, which then we can then output.

2.6 INDEX $\to$ GAPHAM $\to$ F_0

GAPHAM (Gap Hamming): Alice gets $x \in \{0,1\}^n$ and Bob gets $y \in \{0,1\}^n$. They're promised that the Hamming distance $\Delta(x,y) > n/2 + c\sqrt{n}$ or $\Delta(x,y) < n/2 - c\sqrt{n}$ for some constant c, and we need to decide which.

The reduction INDEX $\to$ GAPHAM was shown by [JKS08], implying an $\Omega(n)$ lower bound for GAPHAM. The lower bound $R_{1/3}^{\text{pub}\to}(\text{GAPHAM}) = \Omega(n)$ was also shown earlier, without this reduction, by [IW03, Woo04]. It was later shown that even if you allow yourself an arbitrary number of rounds, you still need $\Omega(n)$ communication [CR12].

An F_0 algorithm that fails with probability $1/3$ and gives a $(1+\epsilon)$ approximation requires $\Omega(1/\epsilon^2)$ space (assume that $1/\epsilon^2 < n$).

Proof: Reduce from GAPHAM. Alice and Bob get t bit vectors, where $t = \Theta(1/\epsilon^2)$. Note that $c\sqrt{t} \leq \epsilon t/3$. Now, note that $2F_0 = |\operatorname{supp}(x)| + |\operatorname{supp}(y)| + \Delta(x,y)$. Alice sends the streaming memory and $|\operatorname{supp}(x)|$ which is $S + \log t$ bits (where S is the space complexity of the streaming algorithm for F_0 approximation). Bob knows $2(1 \pm \epsilon)F_0 = 2F_0 \pm \epsilon t/2$. Then he can estimate $\tilde{\Delta} = 2F_0 \pm \epsilon t/2 - |\operatorname{supp}(x)| + |\operatorname{supp}(y)| = \Delta \pm \epsilon t/2$ and can decide GAPHAM. Thus $S + \log t = \Omega(t)$, implying $S = \Omega(t)$.

References

[CR12] Amit Chakrabarti and Oded Regev. An optimal lower bound on the communication complexity of Gap-Hamming-Distance. *SIAM J. Comput.*, 41(5):1299–1317, 2012.

[GM09] Sudipto Guha and Andrew McGregor. Stream order and order statistics: Quantile estimation in random-order streams. *SIAM J. Comput.*, 38(5):2044–2059, 2009.

[IW03] Piotr Indyk and David P. Woodruff. Tight lower bounds for the distinct elements problem. In *Proceedings of the 44th Symposium on Foundations of Computer Science (FOCS)*, pages 283–288, 2003.

[JKS08] TS Jayram, Ravi Kumar, and D Sivakumar. The one-way communication complexity of hamming distance. *Theory of Computing*, 4(1):129–135, 2008.

[Kus06] Eyal Kushilevitz. *Communication complexity*. Cambridge University Press, Cambridge, 2006.

[Woo04] David P. Woodruff. Optimal space lower bounds for all frequency moments. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 167–175, 2004.

CS 229r: Algorithms for Big Data Fall 2015

Lecture 10 — October 6, 2015

Prof. Jelani Nelson *Scribe: Morris Yau*

1 Overview

In the last lecture we talked about communication complexity. We found that deterministic exact F_0 required $\Omega(n)$ space. We also proved the inequality $D(f) \geq R^{pri}_{\delta}(f) \geq R^{pub}_{\delta}(f) \geq D_{\mu,s}(f)$ that we use to provide lower bounds on randomized and approximate algorithms. Our main tool was reductions from INDEX which has a communication complexity of roughly $\Omega(n)$.

In this lecture we use INDEX to give a lower bound on the space usage of randomized exact F_0. We then present a lower bound for randomized approximate F_0, something that we have thus far been unable to do. We then provide a lower bound on F_p via the disjointness problem. Then we move on to dimensionality reduction, distortion, and distributional Johnson-Lindenstrauss and the fact that it implies Johnson-Lindenstrauss.

2 Randomized Exact Bound F_0

We prove that randomized exact F_0 requires $\Omega(n)$ space with failure probability $\frac{1}{3}$.

Proof. We perform the following reduction from INDEX. Let Alice receive $x \in \{0,1\}^n$ and Bob receive $j \in [n]$. It is then Bob's job to find the j'th index of x. They proceed in the following manner. Alice runs our F_0 algorithm on x and sends both the memory contents of the algorithm and the support of x to Bob. Bob then appends j to the stream and queries F_0 from the the memory contents of the algorithm. If F_0 increases, Bob outputs 0, else he outputs 1. We conclude that for S equal to the space usage of the algorithm

$$S + \log n \geq c * n \implies S = \Omega(n)$$

Where $\log n$ factor comes from sending the support of x. □

3 Randomized Approximate Bound F_0

To prove randomized approximate F_0 has space lower bound $\Omega(\log n)$ we first state this theorem that can be found in Kushilevitz and Nisan. Roughly speaking, it lower bounds the private communication bound by the log of the deterministic communication bound.

Theorem 1. $\forall f : \{0,1\}^n x \{0,1\}^n \to \{0,1\}$, *where f is a communication problem, than*

$$R^{pri}_{\frac{1}{3}} \geq \Omega(\log(D(f)))$$

Proof. If we view f as a two player game between Alice and Bob on a binary tree of height s and total leaves 2^s, than Alice and Bob could deterministically simulate the private randomized procedure on this tree. For instance, for any path from root to leaf, Alice can compute the probability she would stay on the path given that Bob does as well. She can then send these probabilities to Bob for every single leaf. Bob can then compute the probabilities he stays on the same paths and can output the final result accordingly. □

Now we prove that randomized approximate F_0 has space lower bound $\Omega(\log n)$

Proof. Let C be a subset of $\{0,1\}^n$ such that $\forall c \in C$ the support of c is $\frac{n}{100}$. Also $\forall c \neq c' \in C$, we have $|c \cap c'| \leq \frac{n}{2000}$. Finally $|C| \geq 2^{\Omega(n)}$. We have constructed this set in previous lectures. In essence, it is a collection of subsets that are largely disjoint but very numerous. We know deterministic equality, EQ, on C requires $\Omega(n)$ communication. Then using Kushilevitz Nisan we have

$$\implies R^{pri}_{\frac{1}{3}}(EQ_C) \geq \Omega(\log n)$$

Now we notice there is a natural reduction from EQ_C to randomized approximate F_0. Namely, Alice runs F_0 on her set c and sends the memory contents to Bob. Bob then runs c' on the memory contents and determines whether the output for F_0 has roughly doubled. If it has, then $c \neq c'$, if not, than $c = c'$. □

4 Disjointness Problem

We now move on to the t-player disjointness problem, useful for proving lower bounds for the F_p. We have t-players $p_1, p_2, .., p_t$. We assign an n bit string $x_i \in \{0,1\}^n$ to player p_i. We are then promised that either of the following conditions hold.

1. $\forall i \neq j$ we have $x_i \cap x_j \neq \emptyset$
2. $\exists k \in [n]$ such that $\forall i \neq j$ we have $x_i \cap x_j = \{k\}$

The problem is then to find k with the least communication possible where communication occurs from player 1 passing on to player 2 and so on and so forth until player player t gives the final result.

Jelani mentions this theorem but does not prove it because it takes too much time. The proof also uses an information theoretic approach, known as *information complexity* [4]. The idea is the following chain of inequalities, where Π is the optimal δ-error communication protocol for some function f: $R^{pub}_{\delta}(f) = |\Pi| \geq H(\Pi(\mathbf{X})) \geq I(\mathbf{X}; \Pi(X))$, where $\mathbf{X}$ is the set of inputs given to the t players, and $\Pi(\mathbf{X})$ is the transcript of the communication protocol (or the "communication log") when the input is $\mathbf{X}$ (note that it is a random variable since Π uses randomness). Then we define the *information complexity* $IC_{\mu,\delta}(f)$ as the minimum value $I(\mathbf{X}, \Pi(X))$ achievable by any δ-error protocol Π when $\mathbf{X}$ is drawn from distribution μ. Then we have that $R^{pub}_{\delta}(f) \geq IC_{\mu,\delta}(f)$ for all μ. A variant of this approach was used by [2] to obtain lower bounds for t-player disjointness, with improvements in [3]. The sharp bound was shown in [5], with a later work showing how the arguments in [2] could be strengthened to also get the sharp bound [6].

Theorem 2. $R^{pub}_{\frac{1}{3}}(DISJ_t) = \Omega(\frac{n}{t})$

Remark: Although we do not prove the theorem we know that it implies some player sends $\Omega(\frac{n}{t^2})$ bits which is what we'll need to prove the following claim.

Claim 3. *For $p > 2$ the randomized 1.1 approximation to F_p requires $\Omega(n^{1-\frac{2}{p}})$ bits of space.*

Proof. Set $t = \lceil (2n)^{\frac{1}{p}} \rceil$ for the disjoint players problem. Each player creates a virtual stream containing j if and only if $j \in x_i$. We then compute F_p on these virtual streams. If all x_i are disjoint then $F_p \leq n$. Otherwise, $F_p \geq t^p \geq 2n$ because some element k must appear at least t times. Then since our F_p algorithm is a 1.1 approximation, we can discern between the two cases. This implies the space usage of our algorithm, S satisfies

$$S \geq \frac{n}{t^2} = \Omega(n^{1-\frac{2}{p}})$$

as desired. □

5 Dimensionality Reduction

Dimensionality reduction is useful for solving problems involving high dimensional vectors as input. Typically we are asked to preserve certain structures such as norms and angles. Some of the problems include

1. nearest neighbor search
2. large scale regression problems
3. minimum enclosing ball
4. numerical linear algebra on large matrices
5. various clustering applications

Our goals run in the same vein as streaming. That is to say fast runtime, low storage, and low communication. Certain geometric properties that we would like to preserve upon lowering the dimension of the input data include

1. distances
2. angles
3. volumes of subsets of inputs
4. optimal solution to geometric optimization problem

First and foremost, we would like to preserve distances, and to do so we must first define distortion.

5.1 Distortion

Definition 4. *Suppose we have two metric spaces, (X, d_X), and (Y, d_Y), and a function $f : X \to Y$. Then f has distortion D_f if $\forall x, x' \in X$, $C_1 \cdot d_X(x, x') \leq d_Y(f(x), f(x')) \leq C_2 \cdot d_X(x, x')$, where $\frac{C_2}{C_1} = D_f$.*

We will focus on spaces in which $d_X(x, x') = \|x - x'\|_X$ (ie. normed spaces).

5.2 Limitations of Dimensionality Reduction

If $\|\cdot\|_X$ is the l_1 norm, then $D_f \leq C \implies$ in worst case, target dimension is $n^{\Omega(\frac{1}{C^2})}$. That is, there exists a set of n points X, such that for all functions $f : (X, l_1) \to (X', l_1^m)$, with distortion $\leq C$, then m must be at least $n^{\Omega(\frac{1}{C^2})}$[10].

More recently in 2010, we have the following theorem by Johnson and Naor [12]

Theorem 5. *Suppose $(X, ||\cdot||_x)$ is a complete normed vector space or "Banach Space" such that for any N point subset of X, we can map to $O(\log n)$ dimension subspace of X with $O(1)$ distortion, then every n-dimensional linear subspace of X embeds into l_2 with distortion $\leq 2^{2^{O(\log^* n)}}$*

5.3 Johnson Lindenstrauss

Theorem 6. *The Johnson-Lindenstrauss (JL) lemma [11] states that for all $\epsilon \in (0, \frac{1}{2})$, $\forall x_1, ..., x_n \in l_2$, there exists $\Pi \in R^{m \times n}$, $m = O(\frac{1}{\epsilon^2} \log(n))$ such that for all i, j , $(1-\epsilon)\|x_i - x_j\|_2 \leq \|\Pi x_i - \Pi x_j\|_2 \leq (1 + \epsilon)\|x_i - x_j\|_2$*

$f : (x, l_2) \to (x, l_2^m)$, $f(x) = \Pi x$

5.4 Distributional Johnson Lindenstrauss

Theorem 7. *for all $0 < \epsilon, \delta < \frac{1}{2}$, there exists a distribution $D_{\epsilon,\delta}$ on matrices $\Pi \in \mathbb{R}^{m \times n}$, $m = O(\frac{1}{\epsilon^2} \log(\frac{1}{\delta}))$ such that for all $x \in \mathbb{R}^n$, and Π drawn from the distribution $D_{\epsilon,\delta}$,*

$$\mathbb{P}(\|\Pi x\|_2 \notin [(1 - \epsilon)\|x\|_2, (1 + \epsilon)\|x\|_2] < \delta$$

Now we prove that the distributional Johnson Lindenstrauss proves Johnson Lindenstrauss.

Claim 8. $DJL \implies JL$

Proof. Set $\delta < \frac{1}{\binom{N}{2}}$ and look at $T = \frac{x_i - x_j}{\|x_i - x_j\|_2}$ for $i < j$. Also note that $|T| = \binom{N}{2}$. Then

$$P(\Pi \text{ doesn't have distortion } (1 + \epsilon) \text{ on X }) = P(\exists z \in T \text{ such that } \left|||\Pi z||_2^2 - 1\right| \geq \epsilon)$$

and so by union bound this probability is $\leq |T| * \delta < 1$ □

Bibliography.

References

[1] Noga Alon, Yossi Matias, Mario Szegedy. The Space Complexity of Approximating the Frequency Moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.

[2] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar. An information statistics approach to data stream and communication complexity. *J. Comput. Syst. Sci.*, 68(4): 702–732, 2004.

[3] Amit Chakrabarti, Subhash Khot, Xiaodong Sun. Near-Optimal Lower Bounds on the Multi-Party Communication Complexity of Set Disjointness. *IEEE Conference on Computational Complexity*, pgs. 107–17, 2003.

[4] Amit Chakrabarti, Yaoyun Shi, Anthony Wirth, Andrew Chi-Chih Yao. Informational Complexity and the Direct Sum Problem for Simultaneous Message Complexity. *FOCS*, pgs. 270–278, 2001.

[5] Andre Gronemeier. Asymptotically Optimal Lower Bounds on the NIH-Multi-Party Information Complexity of the AND-Function and Disjointness. *STACS*, pgs. 505–516, 2009.

[6] T. S. Jayram. Hellinger Strikes Back: A Note on the Multi-party Information Complexity of AND. *APPROX-RANDOM*, pgs. 562–573, 2009.

[7] T. S. Jayram, Ravi Kumar, D. Sivakumar. The One-Way Communication Complexity of Hamming Distance. *Theory of Computing*, 4(1): 129–135, 2008.

[8] Eyal Kushilevitz, Noam Nisan Communication Complexity *Cambridge University Press*, 1997

[9] David P. Woodruff. Optimal space lower bounds for all frequency moments. *SODA*, pgs. 167–175, 2004.

[10] B. Brinkman and M. Charikar, On the impossibility of dimension reduction in l_1, J. ACM, vol. 52, no. 5, pp. 766–788, 2005.

[11] William B. Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

[12] William B. Johnson and Assaf Naor, The Johnson-Lindenstrauss Lemma Almost Characterizes Hilbert Space, But Not Quite. *Discrete and Computational Geometry*, vol. 43, no. 3, pp. 542–553, 2010.

[13] David Lee Hanson and Farroll Tim Wright. A bound on tail probabilities for quadratic forms in independent random variables. *Ann. Math. Statist.*, 42(3):1079–1083, 1971.

[14] Gilles Pisier. Probabilistic methods in the geometry of Banach spaces. *Probability and Analysis. Lecture Notes in Mathematics.* Vol. 1206, 167–241, 1986.

CS 229r: Algorithms for Big Data Fall 2015

Lecture 10 — October 8, 2015

Prof. Jelani Nelson

Today we will prove the distributional JL lemma from last lecture. First we collect some notation and basic lemmas we will use.

Throughout, for a random variable X, $\|X\|_p$ denotes $(\mathbb{E}\,|X|^p)^{1/p}$. It is known that $\|\cdot\|_p$ is a norm for any $p \geq 1$ (Minkowski's inequality). It is also known $\|X\|_p \leq \|X\|_q$ whenever $p \leq q$. Henceforth, whenever we discuss $\|\cdot\|_p$, we will assume $p \geq 1$.

We often use Jensen's inequality below, especially for $F(x) = |x|^p$ ($p \geq 1$).

Lemma 1 (Jensen's inequality)**.** *For F convex, $F(\mathbb{E}\,X) \leq \mathbb{E}\,F(X)$.*

Lemma 2. *For $1 \leq p < q < \infty$, $\|X\|_p \leq \|X\|_q$.*

Proof. Define $f(x) = |x|^{q/p}$. Then f is convex. Thus by Jensen's inequality,

$$(\mathbb{E}\,|X|^p)^{q/p} \leq \mathbb{E}\,|X|^q.$$

Now raise both sides of the inequality to the $1/p$. □

Definition 3. *The gaussian distribution $\mathcal{N}(0,\sigma^2)$ has density function $f(x) = (2\pi\sigma^2)^{-1/2}e^{-x^2/(2\sigma^2)}$.*

Fact 4. *If $g \sim \mathcal{N}(0,\sigma^2)$ then $\mathbb{E}\,g^p$ for integer p is 0 for p odd and is $\sigma^p(p-1)\cdot(p-3)\cdots 1 < (\sigma\sqrt{p})^p$ for p even.*

Lemma 5 (Khintchine inequality)**.** *For any $p \geq 1$, $x \in \mathbb{R}^n$, and (σ_i) independent Rademachers,*

$$\|\sum_i \sigma_i x_i\|_p \lesssim \sqrt{p}\cdot\|x\|_2$$

Proof. Without loss of generality we can assume p is an even integer. (If not, let q be the smallest even integer larger than p then it suffices to have $\|\sum_i \sigma_i x_i\|_q \lesssim \sqrt{p}\|x\|_2$ since $\|\cdot\|_p \leq \|\cdot\|_q$ by Lemma 2.) Consider (g_i) independent gaussians of mean zero and variance 1. Expand $\mathbb{E}(\sum_i \sigma_i x_i)^p$ into a sum of monomials.

$$\begin{aligned}\mathbb{E}(\sum_i \sigma_i x_i)^p &= \sum_{t=1}^{\min\{p,n\}} \sum_{i_1<i_2<\ldots<i_t} \sum_{\substack{d_1,\ldots,d_t\geq 1\\ d_1+\ldots+d_t=p}} \binom{p}{d_1,\ldots,d_t}\left(\prod_{j=1}^t x_{i_j}^{d_j}\right)\left(\mathbb{E}_\sigma \prod_{j=1}^t \sigma_{i_j}^{d_j}\right)\\ &= \sum_{t=1}^{\min\{p,n\}} \sum_{i_1<i_2<\ldots<i_t} \sum_{\substack{d_1,\ldots,d_t\geq 1\\ d_1+\ldots+d_t=p}} \binom{p}{d_1,\ldots,d_t}\left(\prod_{j=1}^t x_{i_j}^{d_j}\right)\left(\prod_{j=1}^t \mathbb{E}_{\sigma_{i_j}} \sigma_{i_j}^{d_j}\right)\end{aligned}$$

Any monomial with odd exponents (i.e. odd d_j) vanishes, as in the gaussian case. Meanwhile, monomials with all d_j being even have $\prod_{j=1}^t x_{i_j}^{d_j}$ nonnegative and $\prod_{j=1}^t \mathbb{E}_{\sigma_{i_j}} \sigma_{i_j}^{d_j} = 1$. Meanwhile if

the σ_{i_j} are replaced by gaussians g_{i_j}, then $\prod_{j=1}^{t} \mathbb{E}_{\sigma_{i_j}} g_{i_j}^{d_j} \geq 1$. Thus the Rademacher pth moment is term-by-term dominated by the gaussian case and thus $\|\sum_i \sigma_i x_i\|_p \leq \|\sum_i g_i x_i\|_p$. But $\sum_i g_i x_i$ is a gaussian with mean zero and variance $\|x\|_2^2$, and we apply Fact 4.

The point of the above argument was to show that the Rademacher case is bounded by the gaussian case, after which point we concluded. Another way to show this is as follows. Now we will not require p to be an even integer.

$$\begin{aligned}
\|\sum_i \sigma_i x_i\|_p &= \sqrt{\frac{\pi}{2}} \cdot \|\mathop{\mathbb{E}}_g \sum_i \sigma_i |g_i| x_i\|_p \text{ (since } \mathbb{E}\,|g| = \sqrt{2/\pi}) \\
&\leq \sqrt{\frac{\pi}{2}} \cdot \|\sum_i \sigma_i |g_i| x_i\|_p \text{ (Jensen)} \\
&= \sqrt{\frac{\pi}{2}} \cdot \|\sum_i g_i x_i\|_p \text{ ($\sigma_i |g_i|$ is distributed as g_i)}
\end{aligned}$$

□

We now prove a decoupling inequality which will be useful for our proof of Hanson-Wright, which we will use to prove distributional JL. We use $\|\cdot\|_{L^p(X)}$ to denote $(\mathbb{E}_X |\cdot|^p)^{1/p}$ when we want to make it clear which random variable we are taking the expectation over.

Lemma 6 (Decoupling [dlPnG99])**.** *Let $x_1, \ldots, x_n$ be independent and mean zero, and $x'_1, \ldots, x'_n$ identically distributed as the x_i and independent of them. Then for any $(a_{i,j})$ and for all $p \geq 1$*

$$\|\sum_{i \neq j} a_{i,j} x_i x_j\|_p \leq 4 \|\sum_{i,j} a_{i,j} x_i x'_j\|_p$$

Proof. Let $\eta_1, \ldots, \eta_n$ be independent Bernoulli random variables each of expectation $1/2$. Then

$$\begin{aligned}
\|\sum_{i \neq j} a_{i,j} x_i x_j\|_{L^p(x)} &= 4 \cdot \|\mathop{\mathbb{E}}_\eta \sum_{i \neq j} a_{i,j} x_i x_j |\eta_i||1 - \eta_j|\|_{L^p(x)} \\
&\leq 4 \cdot \|\sum_{i \neq j} a_{i,j} x_i x_j \eta_i (1 - \eta_j)\|_{L^p(x,\eta)} \text{ (Jensen)} \qquad (1)
\end{aligned}$$

Hence there must be some fixed vector $\eta' \in \{0,1\}^n$ which achieves

$$\|\sum_{i \neq j} a_{i,j} x_i x_j \eta_i (1 - \eta_j)\|_{L^p(x,\eta)} \leq \|\sum_{i \in S} \sum_{j \notin S} a_{i,j} x_i x_j\|_{L^p(\eta)}$$

where $S = \{i : \eta'_i = 1\}$. Let x_S denote the $|S|$-dimensional vector corresponding to the x_i for $i \in S$. Then

$$\begin{aligned}
\|\sum_{i \in S} \sum_{j \notin S} a_{i,j} x_i x_j\|_{L^p(x)} &= \|\sum_{i \in S} \sum_{j \notin S} a_{i,j} x_i x'_j\|_{L^p(x_S, x'_{\bar{S}})} \\
&= \|\mathop{\mathbb{E}}_{x_S} \mathop{\mathbb{E}}_{x'_{\bar{S}}} \sum_{i,j} a_{i,j} x_i x'_j\|_{L^p(x_S, x'_{\bar{S}})} \text{ ($\mathbb{E}\, x_i = \mathbb{E}\, x'_j = 0$)} \\
&\leq \|\sum_{i,j} a_{i,j} x_i x'_j\|_{L^p(x,x')} \text{ (Jensen)}
\end{aligned}$$

□

The following proof of the Hanson-Wright was shared to me by Sjoerd Dirksen (personal communication). See also a recent proof in [RV13].

Recall that by problem set 1, problem 1, the statement of the Hanson-Wright inequality below is equivalent to the statement that there exists a constant $C > 0$ such that for all $\lambda > 0$

$$\mathop{\mathbb{P}}_{\sigma}\left(|\sigma^T A\sigma - \mathbb{E}\,\sigma^T A\sigma| > \lambda\right) \lesssim e^{-C\lambda^2/\|A\|_F^2} + e^{-C\lambda/\|A\|}. \tag{2}$$

Theorem 7 (Hanson-Wright inequality [HW71])**.** *For $\sigma_1, \ldots, \sigma_n$ independent Rademachers and $A \in \mathbb{R}^{n\times n}$ real and symmetric, for all $p \geq 1$*

$$\|\sigma^T A\sigma - \mathbb{E}\,\sigma^T A\sigma\|_p \lesssim \sqrt{p}\cdot\|A\|_F + p\cdot\|A\|.$$

Proof. Without loss of generality we assume in this proof that $p \geq 2$ (so that $p/2 \geq 1$). Then

$$\begin{aligned}
\|\sigma^T A\sigma - \mathbb{E}\,\sigma^T A\sigma\|_p &\lesssim \|\sigma^T A\sigma'\|_p \text{ (decoupling)} &(3)\\
&\lesssim \sqrt{p}\cdot\|\|Ax\|_2\|_p \text{ (Khintchine)} &(4)\\
&= \sqrt{p}\cdot\|\|Ax\|_2^2\|_{p/2}^{1/2} &(5)\\
&\leq \sqrt{p}\cdot\|\|Ax\|_2^2\|_p^{1/2}\\
&\leq \sqrt{p}\cdot(\|A\|_F^2 + \|\|Ax\|_2^2 - \mathbb{E}\,\|Ax\|_2^2\|_p)^{1/2} \text{ (triangle inequality)}\\
&\leq \sqrt{p}\cdot\|A\|_F + \sqrt{p}\cdot\|\|Ax\|_2^2 - \mathbb{E}\,\|Ax\|_2^2\|_p^{1/2}\\
&\lesssim \sqrt{p}\cdot\|A\|_F + \sqrt{p}\cdot\|x^T A^T Ax'\|_p^{1/2} \text{ (decoupling)}\\
&\lesssim \sqrt{p}\cdot\|A\|_F + p^{3/4}\cdot\|\|A^T Ax\|_2\|_p^{1/2} \text{ (Khintchine)}\\
&\lesssim \sqrt{p}\cdot\|A\|_F + p^{3/4}\cdot\|A\|^{1/2}\cdot\|\|Ax\|_2\|_p^{1/2} &(6)
\end{aligned}$$

Writing $E = \|\|Ax\|_2\|_p^{1/2}$ and comparing (4) and (6), we see that for some constant $C > 0$,

$$E^2 - Cp^{1/4}\|A\|^{1/2}E - C\|A\|_F \leq 0.$$

Thus E must be smaller than the larger root of the above quadratic equation, implying our desired upper bound on E^2. □

Remark 1. The "square root trick" in the proof of the Hanson-Wright inequality above is quite handy and can be used to prove several moment inequalities (for example, you will see how to prove the Bernstein inequality with it in tomorrow's lecture). As far as I am aware, the trick was first used in a work of Rudelson [Rud99].

Remark 2. We could have upper bounded Eq. (5) by

$$\sqrt{p}\cdot\|A\|_F + \sqrt{p}\cdot\|\|Ax\|_2^2 - \mathbb{E}\,\|Ax\|_2^2\|_{p/2}^{1/2}$$

by the triangle inequality. Now notice we have bounded the pth central moment of a symmetric quadratic form (3) by the $p/2$th moment also of a symmetric quadratic form. Writing $p = 2^k$, this observation leads to a proof by induction on k, which was the approach used in [DKN10].

Distributional Johnson-Lindenstrauss (DJL) lemma Now we finally prove the *Distributional JL Lemma (DJL)* stated in last lecture (which implies the JL lemma itself).

Lemma 8. *DJL Lemma For any integer $n > 1$ and $\varepsilon, \delta \in (0, 1/2)$, there exists a distribution $\mathcal{D}_{\varepsilon,\delta}$ over $\mathbb{R}^{m \times n}$ for $m \lesssim \varepsilon^{-2} \log(1/\delta))$ such that for any $x \in \mathbb{R}^n$ of unit Euclidean norm,*

$$\mathbb{P}_{\Pi \sim \mathcal{D}_{\varepsilon,\delta}} (|\|\Pi x\|_2^2 - 1| > \varepsilon) < \delta$$

Proof. Write $\Pi_{i,j} = \sigma_{i,j}/\sqrt{m}$, where the $\sigma_{i,j}$ are independent Rademachers. Also overload σ to mean these Rademachers arranged as a vector of length mn, by concatenating rows of Π. Note

$$\Pi x = A_x \sigma, \text{ implying } \|\Pi x\|_2^2 = \|A_x \sigma\|_2^2$$

where

$$A_x = \frac{1}{\sqrt{m}} \cdot \begin{bmatrix} -x^T- & 0 & \cdots & 0 \\ 0 & -x^T- & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & -x^T- \end{bmatrix}. \tag{7}$$

Thus

$$\mathbb{P}(|\|\Pi x\|_2^2 - 1| > \varepsilon) = \mathbb{P}(|\|A_x \sigma\|_2^2 - \mathbb{E}\,\|A_x \sigma\|_2^2| > \varepsilon),$$

where we see that the right-hand side is readily handled by the Hanson-Wright inequality with $A = A_x^T A_x$ (using (2)). Observe A is a block-diagonal matrix with each block equaling $(1/m)xx^T$, and thus $\|A\| = \|x\|_2^2/m = 1/m$. We also have $\|A\|_F^2 = 1/m$. Thus Hanson-Wright yields

$$\mathbb{P}(|\|\Pi x\|_2^2 - 1| > \varepsilon) \lesssim e^{-C\varepsilon^2 m} + e^{-C\varepsilon m},$$

which for $\varepsilon < 1$ is at most δ for $m \gtrsim \varepsilon^{-2} \log(1/\delta)$. □

References

[DKN10] Ilias Diakonikolas, Daniel M. Kane, and Jelani Nelson. Bounded independence fools degree-2 threshold functions. In *51th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 11–20, 2010.

[dlPnG99] Victor de la Peña and Evarist Giné. *Decoupling: From dependence to independence.* Probability and its Applications. Springer-Verlag, New York, 1999.

[HW71] David Lee Hanson and Farroll Tim Wright. A bound on tail probabilities for quadratic forms in independent random variables. *Ann. Math. Statist.*, 42:1079–1083, 1971.

[Rud99] Mark Rudelson. Random vectors in the isotropic position. *J. Functional Analysis*, 164(1):60–72, 1999.

[RV13] Mark Rudelson and Roman Vershynin. Hanson-Wright inequality and sub-gaussian concentration. *Electronic Communications in Probability*, 18:1–9, 2013.

CS 229r: Algorithms for Big Data Fall 2015

Lecture October 13th

Prof. Jelani Nelson *Scribe: Vasileios Nakos*

1 Overview

In the last lecture we looked at (ϵ, δ)-DJL: There exists a distribution $\mathbb{D}_{\epsilon,\delta}$ over $\mathbb{R}^{m\times n}$ with $m = O(\varepsilon^{-2}\log(1/\delta))$ such that $\forall x, \|x\| = 1$ we have that $\mathbb{P}_{\Pi\sim\mathbb{D}_{\epsilon,\delta}}[|\|\Pi x\|_2^2 - 1| < max\{\epsilon, \epsilon^2\}] < \delta$. Moreover, we have seen metric JL which tells us that for all $X \subset l_2^n$ there exist a matrix $\Pi \in \mathbb{R}^{m\times n}$ with $m = \Theta(\epsilon^{-2}\log|X|)$ such that for all $x \in T$ we have that $|\|\Pi x\|_2^2 - 1| < \epsilon$, where $T = \{\frac{x-y}{\|x-y\|}, x, y \in X\}$.

2 Today

Today we are going to look at three main things:

- Johnson-Lindenstrauss lower bound
- Beyond worst-case analysis and more specifically what is called Gordon's Theorem
- Bounding supremum of gaussian processes.

3 Main Section

3.1 Lower bounds on Dimensionality Reduction

3.1.1 Lower bounds on Distributional Johnson Lindenstrauss Lemma

The first lower bound was proved by Jayram and Woodruff [1] and then by Kane,Meka,Nelson [2]. The lower bound tells that any (ϵ, δ)-DJL for $\epsilon, \delta \in (0, 1/2)$ must have $m = \Omega(min\{n, \epsilon^{-2}\log(\frac{1}{\delta})\})$. The second proof builds on the following idea: Since for all x we have the probabilistic guarantee $\mathbb{P}_{\Pi\sim\mathbb{D}_{\epsilon,\delta}}[|\|\Pi x\|_2^2 - 1| < max\{\epsilon, \epsilon^2\}] < \delta$, then it is true also for any distribution over x. We are going to pick x according to the uniform distribution over the sphere. Then this implies that there exists a matrix Π such that $\mathbb{P}_x[|\|\Pi x\|_2^2 - 1| < \epsilon] < \delta$. It is shown that this cannot happen for any fixed matrix $\Pi \in \mathbb{R}^{m\times n}$ unless m satisfies the lower bound.

3.1.2 Johnson-Lindenstrauss lower bound for the linear mapping case

If we insist on linear maps, then a lower bound of Kasper and Nelson [3] tells us that m must be at least $\Omega(min\{n, \epsilon^{-2}\log|X|\})$. The hard set is shown to exist via the probabilistic method,

and is constructed by taking the union of $\{0, e_1, \ldots, e_n\}$ in $\mathbb{R}^n$ together with plus sufficiently many random vectors. We just need to take a fine finite net that approximates all possible linear maps that work for the simplex (i.e. $\{e_1, \ldots, e_n\}$) and then argue that for each linear map in the net then with some good probability there exists a point from the random ones such that its length is not preserved by the map. So, approximating well enough the set of linear maps and adjusting the parameteres then we can take a union bound over all matrices in the net.

3.1.3 Johnson-Lindenstrauss lower bound for the general case

This lower bound has been found by Alon [4]. It tells that m must be at least $\Omega(min\{n, \epsilon^{-2}\frac{\log n}{\log(\frac{1}{\epsilon})}$ to preserve distances between the set of points $X = \{0, e_1, \ldots, e_n\}$. The hard set is the simplex $X = \{0, e_1, .., e_n\}$. Let f be the mapping. Translate the embedding so that $f(0) = 0$ (i.e. translate the embedding so that each point $x \in X$ is actually mapped to $f(x) - f(0)$; this does not affect any distances). Now write $f(e_i) = v_i$. Then we have that $\|v_i\| = 1 \pm \epsilon$ since f preserves the distance from e_i to 0. We also have for $i \neq j$ that $\|v_i - v_j\| = \sqrt{2}(1 \pm \epsilon)$. This implies since $\|v_i - v_j\| = \|v_i\|^2 + \|v_j\|^2 - 2\langle v_i, v_j\rangle$ we get that $\langle v_i, v_j\rangle = O(\epsilon)$. Setting $w_i = \frac{v_i}{\|v_i\|}$ we have that $\|w_i\| = 1$ and $\langle w_i, w_j\rangle = O(\epsilon)$. Let Π be the matrix that has w_i as columns. Then observe that $A = \Pi^T\Pi$ is a matrix with 1 on the diagonal and elements with absolute value at most ϵ everywhere. We have that $rank(A) = rank(\Pi^T\Pi) = rank(\Pi) \leq m$. We are going to use the following lemma which solves the problem for $\epsilon = \frac{1}{\sqrt{n}}$ and then bootstrap it to work for all values of ϵ.

Lemma 1. *Any real symmetric matrix that is $\frac{1}{\sqrt{n}}$-near to the identity matrix, i.e. its diagonals are 1 and off-diagonals are in $[-1/\sqrt{n}, 1/\sqrt{n}]$, must have $rank(A) \geq \Omega(n)$.*

Proof. Let $\lambda_1, ..\lambda_r$ be the non-zero eigenvalues of A, where $r = rank(A)$. By Cauchy-Schwarz, we have $r \geq \frac{(\sum_{i=1}^r \lambda_i)^2}{\sum_{i=1}^r \lambda_i^2}$. By standard linear algebra the numerator is the trace of A squared and the denominator is just the Frobenius norm of A squared. We have that $tr(A) = n$ and $\|A\|_F^2 \leq n + n(n-1)\epsilon^2$. Plugging everything into the inequality along with the fact that $\epsilon = \frac{1}{\sqrt{n}}$ we get the desired result.

□

Theorem 2. *Any real symmetric matrix that is ϵ-near to the identity matrix must have $rank(A) \leq min\{n, \epsilon^{-2}\frac{\log n}{\log \frac{1}{\epsilon}}\}$.*

Proof. Define the matrix $A^{(k)}$ such that $(A^{(k)})_{ij} = a_{ij}^k$. We will build our proof on the following claim: It holds that $rank(A^{(k)} \leq \binom{r+k-1}{k}$ where $r = rank(A)$. Assumign that the claim is true we pick k to be the closest integet to $\log n_{\epsilon^{-1}}\sqrt{n}$. Thus $\epsilon^k \leq \frac{1}{\sqrt{n}}$, so we have that $\Omega(n) \leq r(A^{(k)}) \leq \binom{r+k-1}{k}$. Using the fact that $\binom{n}{k} \leq (enk^{-1})^k$ and walking thought the calculations we can get the desired result.
What remains is to prove the claim. Let $t_1, ..t_r$ be the row-space of A. This means that $\forall i \exists \beta \in mathbbR^r$ such that $a_i = \sum_{q=1}^r \beta_q t_q$. Then observe that $(A^{(k)})_{ij} = a_{ij}^k = (\sum_{q=1}^r \beta e_q(t_q)_j)^k = \sum_{q1,\ldots,q_k} \Pi_{z=1}^k \beta_{q_z} \Pi_{z=1}^k t_{q_z}$. It is easy to see that each vector of this form is a linear combination of

vectors of the form $(\Pi_{z=1}^{y}(t_{q_z})_1^{d_z^1}, \Pi_{z=1}^{y}(t_{q_z})_2^{d_z^2}, ..)$. where $\sum d_z^i = k$. This is a standard combinatorics problem of putting r balls into bins k bins with repetition, so the answer is $\binom{r+k-1}{k}$.

□

3.2 Beyond Worst Case Analysis

Given a subset T of the unit sphere- for example $\{T = \frac{x-y}{\|x-y\|}, x, y \in X\}$- ideally we would like that $\forall x \in T, |\|\Pi x\|^2 - 1| < \epsilon$. We want that $\mathbb{P}_\Pi(sup_{x\in T}|\|\Pi x\|^2 - 1| > \epsilon) < \delta$.

We are moving with Gordon's Theorem [10] which was several times [5] [7] [9]. First of all we are going to define the gaussian mean widht of the set.

Definition 3. *The Gaussian mean width of a set T is defined as $g(T) = \mathbb{E}_g sup_{x\in T} \langle g, x\rangle$.*

Back to Gordon's Theorem. Suppose that $\Pi_{ij} = \frac{\pm 1}{\sqrt{m}}$ for random signs, with $m \geq \Omega(\epsilon^{-2}(g^2(T) + \log \frac{1}{\delta})$. Then we have that $\mathbb{P}_\Pi(sup_{x\in T}|\|\Pi x\|^2 - 1| > \epsilon) < \delta$. Actually, we just need a distribution that decays as fast as a gaussian, has variance one and zero mean.

Let us give a simple example of the gaussian mean width. For example, if T is the simplex then we have that $g(T) = \|g\|_\infty$ which is roughly equal to $\log n$ by standard computations on gaussians. Actually, what Gordon's theorem tells us is that if the vectors of T have a nice geometry then one can improve upon Johnson-Lindenstrauss: the more well-clustered the vectors are, the lower dimension you can achieve.

We continue with the following claim: $\forall T$ which is a subset of the unit sphere, $g(T) \leq O(\sqrt{\log N})$, where $N = |T|$.

Proof. Define $Z_i = |\langle g_i, x_i\rangle|$, where $T = \{x_1, .., x_N\}$. Then

$$g(T) \leq \mathbb{E}_g max\{Z_1, ..Z_N\} = \int_0^\infty \mathbb{P}_g(max\{Z_1, .., Z_N\} > u)du =$$

$$= \int_0^{2\sqrt{\log n}} \mathbb{P}_g(max\{Z_1, .., Z_N\} > u)du + \int_0^{2\sqrt{\log n}} \mathbb{P}_g(max\{Z_1, .., Z_N\} > u)du \leq$$

$$\leq 2\sqrt{\log n} + \int_{2\sqrt{\log n}}^\infty \mathbb{P}_g(\exists Z_i \geq u)du \leq 2\sqrt{\log n} + \int_{2\sqrt{\log n}}^\infty \sum_i \mathbb{P}_g(Z_i \geq u)du \leq$$

$$2\sqrt{\log N} + \int_{2\sqrt{\log n}}^\infty Ne^{-u^2/2} \leq 2\sqrt{\log n} + O(1)$$

□

3.2.1 How to bound g(T)

- $g(T) \leq \sqrt{\log |T|}$, as we just showed. In fact if every vector in T has norm at most α, then one gets $g(T) \lesssim \alpha\sqrt{\log |T|}$.
- Let $T' \subset T$ be such that $\forall x \in T$, $\exists x' \in T'$ such that $\|x - x'\| \leq \epsilon$. That is, T' is an ϵ-net of T. This implies that $\langle g, x\rangle = \langle g, x'\rangle + \langle g, x - x'\rangle \leq \|g\|_2\epsilon$, which implies that $g(T) \leq g(T') + \epsilon\mathbb{E}\|g\|_2^2 \leq g(T') + e(\mathbb{E}\|g\|_2^2)^{\frac{1}{2}} \leq g(T') + \epsilon\sqrt{n} \leq O(\sqrt{\log |T'|}) + \epsilon\sqrt{n}$. Thus if T is covered well by a small net, one can get a better bound.
- Let $T_{\subset}T_1 \subset T_2 \subset ... \subset T$, such that T_r is a (2^{-r})-net of T (we are assuming every vector in T has at most unit norm). Then $x = x^{(0)} + (x^{(1)} - x^{(0)}) + (x^{(2)} - x^{(1)}) + ...$ Then we have

$$\begin{aligned}
\mathbb{E}\, sup_{x\in T}\langle g, x\rangle &\leq \mathbb{E}\sup_{x\in T}\left\langle g, x^{(0)}\right\rangle + \sum_{r=1}^{\infty}\mathbb{E}\sup_{x\in T}\left\langle g, x^{(r)} - x^{(r-1)}\right\rangle \\
&\lesssim \log^{1/2}|T_0| + \sum_{r=1}^{\infty}(\sup_{x\in T}\|x^{(r)} - x^{(r-1)}\|)\cdot\log^{1/2}(|T_r|\cdot|T_{r-1}|) \\
&\lesssim \log^{1/2}|T_0| + \sum_{r=1}^{\infty}\frac{1}{2^r}\cdot\log^{1/2}|T_r|.
\end{aligned}$$

  The llast inequality holds since by the triangle inequality, $\|x^{(r)} - x^{(r-1)}\| \leq \|x^{(r)} - x\| + \|x^{(r-1)} - x\| \leq 3/2^{r-1}$. Furthermore, $|T_{r-1}| \leq |T_r|$, so $\log(|T_r|\cdot|T_{r-1}|) \leq \log(|T_r|^2) = 2\log|T_r|$. Thus $g(T) \leq \sum_{r=0}^{\infty} 2^{-r}\log^{\frac{1}{2}}|T_r| = \sum_{r=0}^{\infty} 2^{-r}N(T_2, \|\dot{\|}_2, 2^{-r})$, where $N(T, d, \epsilon)$ is the size of the best ϵ-net of T under metric d. Bounding this sum by an integral, we have that $g(T)$ is at most a constant factor times $\int_0^{\infty}\log^{\frac{1}{2}}N(T, \|\cdot\|_2, u)du$. This inequality is called Dudley's inequality.
- Write $S_0 \subset S_1 \subset ... \subset T$, such that $|S_0| = 1$ and $\|S_s| \leq 2^{2^s}$. One can show that the Dudley bound is in fact

$$\inf_{\{S_s\}_{s=0}^{\infty}}\sum_{s=0}^{\infty}2^{s/2}\sup_{x\in T}d_{\|\cdot\|_2}(x, S_s).$$

  Write

$$\gamma_2(T) = \inf_{\{S_s\}_{s=0}^{\infty}}\sup_{x\in T}\sum_{s=0}^{\infty}2^{s/2}d_{\|\cdot\|_2}(x, S_s).$$

  It was shown by Fernique [8] that $g(T) \lesssim \gamma_2(T)$ for all T. Talagrand later showed that in [11] the lower bound is also true, and hence $g(T) = \Theta(\gamma_2(T))$; this is known as the "majorizing measures" theorem, which we will not prove in this class.

3.2.2 Johnson Lindenstrauss implies Gordon's Theorem

What we already saw is that Gordon's theorem implies the Johnson Lindenstrauss lemma. In fact this summer by Oymak, Recht and Soltanolkotabi [6] it was proved that with right parameteres the Distributional Johnson Lindenstrauss lemma implies Gordon's theorem. Basically take a DJL $\epsilon' = \frac{\epsilon}{\gamma_2^2(T)}$ then for $m \geq \epsilon^{-2}(\gamma_2^2(T)\log\frac{1}{\delta})($ where we hide constants in the inequalities) we take

the guarantee for Gordon's Theorem. Actually, their proof works by preserving the sets S_s (plus differences and sums of vectors in these sets) at different scales. The result is not exactly optimal because it is known $m = O(\epsilon^{-2}(\gamma_2^2(T) + \log(1/\delta)))$ suffices (see for example [9, 7]), but it provides a nice reduction from Gordon's theorem to DJL.

References

[1] T. S. Jayram, David P. Woodruff. Optimal Bounds for Johnson-Lindenstrauss Transforms and Streaming Problems with Subconstant Error. *ACM Transactions on Algorithms* 9(3), 26, 2013.

[2] Daniel M. Kane, Raghu Meka, Jelani Nelson. Almost optimal explicit Johnson-Lindenstrauss transformations. In *Proceedings of the 15th International Workshop on Randomization and Computation (RANDOM)*, pages 628–639, 2011.

[3] Kasper Green Larsen and Jelani Nelson. The Johnson-Lindenstrauss lemma is optimal for linear dimensionality reduction. *CoRR* abs/1411.2404, 2014.

[4] Noga Alon. Problems and results in extremal combinatorics–I. *Discrete Mathematics* 273(1-3), pages 31–53, 2003.

[5] Bo'az Klartag, Shahar Mendelson. Empirical processes and random projections. *J. Functional Analysis*, 225(1), pages 229–245, 2005.

[6] Samet Oymak, Benjamin Recht, Mahdi Soltanolkotabi. Isometric sketching of any set via the Restricted Isometry Property. *CoRR* abs/1506.03521, 2015.

[7] Sjoerd Dirksen. Tail bounds via generic chaining. *Electron. J. Probab.*, 20(53):1–29, 2015.

[8] Xavier Fernique. Regularité des trajectoires des fonctions aléatoires gaussiennes. *Ecole dEté de Probabilités de Saint-Flour IV*, *Lecture Notes in Math.*, vol. 480, pages 1–96, 1975.

[9] Shahar Mendelson, Alain Pajor, Nicole Tomczak-Jaegermann. Reconstruction and subgaussian operators in asymptotic geometric analysis. *Geometric and Functional Analysis*, vol. 1, pages 1248–1282, 2007.

[10] Yehoram Gordon. On Milman's inequality and random subspaces which escape through a mesh in $\mathbb{R}^n$. *Geometric Aspects of Functional Analysis*, vol. 1317, pages 84–106, 1986–87.

[11] Michel Talagrand. Majorizing measures: the generic chaining. *Ann. Probab.*, 24(3), pages 1049–1103, 1996.

CS 229r: Algorithms for Big Data Fall 2015

Lecture 13 — October 15, 2015

Prof. Jelani Nelson *Scribe: Yakir Reshef*

1 Recap and overview

Last time we started by talking about lower bounds for JL that were worst-case. That is, we stated a lower bound on the reduced dimension m such that for any Π mapping to $\mathbb{R}^m$ there would exist a "bad" set T of x's whose distances would not be preserved by Π. But this left open the question of how well we could do on some particular set T. This was where Gordon's theorem came in. It said

Theorem 1 (Gordon)**.** *Suppose $T \subset S^{n-1}$. If $\Pi \in \mathbb{R}^{m\times n}$ has $\Pi_{ij} = g_{ij}/\sqrt{m}$, where the g_{ij} are iid standard normals, and $m \gtrsim \frac{g^2(T)+1}{\varepsilon^2}$, then*

$$P_\Pi\left(\exists x \in T \,:\, |\|\Pi x\| - 1| > \varepsilon\right) < \frac{1}{10}.$$

where $g(T) = E_g \sup_{x\in T}\langle g, x\rangle$ is the mean width of T, with the expectation taken over a gaussian with mean zero and identity covariance.

Recall that we also mentioned the following result last time concerning $g(T)$, with one direction of the inequality showed by Fernique and the other by Talagrand:

Theorem 2. *Let $T \subset \mathbb{R}^n$ bounded, and let $T_0 \subset T_1 \subset \cdots T$ be such that $|T_0| = 1$ and $|T_r| \le 2^{2^r}$. Define*

$$\gamma_2(T, d) := \inf_{\{T_r\}_{r=0}^\infty} \sup_{x\in T} \sum_{r=0}^{\infty} 2^{r/2} d(x, T_r).$$

Then $\gamma_2(T, \ell_2) \simeq g(T)$.

Henceforth, when we say $\gamma_2(T)$ without specifying a metric, ℓ_2 is implied.

Gordon's theorem implies DJL, because in general $g(T)$ is at most $\sqrt{\log|T|}$ in it can be much smaller for some T. Today we'll show that the converse is also true, i.e., that DJL implies Gordon's theorem.

2 Today: DJL ⇒ Gordon's theorem

2.1 Statement of result

The main result is summarized in the following theorem.

Theorem 3 ([1]). *Define* $L = \lceil \log n \rceil$, $\tilde{\varepsilon} = \varepsilon/(c\gamma_2(T))$, $\tilde{\varepsilon}_r = \max\{2^{r/2}\tilde{\varepsilon}, 2^{r/2}\tilde{\varepsilon}^2\}$, $\delta_r = \frac{\delta}{C2^r 8^{2^r}}$. *Let* $T \subset S^{n-1}$. *Then if* D *satisfies* $(\varepsilon_r, \delta_r)$*-DJL for* $r = 0, \ldots, L$, *then*

$$P_{\Pi \sim D}\left(\sum_{x \in T} \left| \|\Pi x\|_2^2 - 1 \right| > \varepsilon\right) < \delta$$

To see why this implies Gordon's theorem, we consider the random sign matrix, e.g., $\Pi_{ij} = \frac{\sigma_{ij}}{\sqrt{m}}$. We know that this matrix satisfies $(\tilde{\varepsilon}, \tilde{\delta})$-DJL for $m \gtrsim \frac{\log(1/\tilde{\delta})}{\tilde{\varepsilon}^2}$, which equals $\frac{2^r \log(1/\tilde{\delta})}{(2^{r/2}\tilde{\varepsilon})^2} \geq \frac{\log(1/\delta_r)}{\varepsilon_r^2}$ for all r. The theorem therefore applies and so we see that we get an (ε, δ) guarantee with $m \gtrsim \log(1/\delta)/\tilde{\varepsilon}^2 \asymp \frac{\gamma_2^2(T)}{\varepsilon^2}\log(1/\delta)$. And since $\gamma_2(T) \simeq g(T)$, this is approximately $\frac{g^2(T)\log(1/\delta)}{\varepsilon^2}$, which gives Gordon's theorem. As mentioned last time, different proofs yield that $\frac{g^2(T)+\log(1/\delta)}{\varepsilon^2}$ actually suffices.

2.2 Proof of result

To prove the above theorem, the lemma below suffices.

Lemma 1. *For a given set* T, *let* T_r *be the sequence that achieves the infimum in the definition of* γ_2. *To achieve* $\sup_{x \in T} \left| \|\Pi x\|_2^2 - 1 \right| < \varepsilon$, *it suffices that for all* $r = 0, \ldots, L$, *the following hold simultaneously for all* $r \in [L]$.

- *For all* $v \in T_{r-1} \cup T_r \cup (T_{r-1} - T_r)$,

$$\|\Pi v\| \leq (1 + 2^{r/2}\tilde{\varepsilon})\|v\| \tag{1}$$

- *For all* $v \in T_{r-1} \cup T_r \cup (T_{r-1} - T_r)$,

$$|\|\Pi v\|^2 - \|v\|^2| \leq \max\{2^{r/2}\tilde{\varepsilon}, 2^r\tilde{\varepsilon}^2\} \cdot \|v\|^2 \tag{2}$$

- *For all* $u \in T_{r-1}$ *and* $v \in T_r - \{u\}$,

$$|\langle \Pi u, \Pi v \rangle - \langle u, v \rangle| \leq \max\{2^{r/2}\tilde{\varepsilon}, 2^r\tilde{\varepsilon}^2\} \cdot \|u\| \cdot \|v\| \tag{3}$$

- *We also have*

$$\|\Pi\| \leq 1 + (1/4)2^{L/2}\tilde{\varepsilon} \tag{4}$$

We note that it is not too bad to show that the first three conditions hold with high probability since they are all JL-type conditions. The third one is a bit less obvious since it's about dot products instead of norms. But notice that $\|u+v\|^2 - \|u-v\|^2 = 4\langle u, v\rangle$. So if $\|u\| = \|v\| = 1$, then Π preserving $u+v$ and $u-v$ means that $\langle \Pi u, \Pi v\rangle = \frac{1}{4}(\|\Pi u + \Pi v\|^2 - \|\Pi u + \Pi v\|^2) = \langle u, v\rangle \pm O(\varepsilon)$. If u and v don't have unit norm you can scale them to achieve the above condition. So the third condition also follows from the DJL assumption. (We neglect the fourth condition above for now, since it's based on a standard argument which is based on a constant-sized net of S^{n-1} that we'll see later in the course.)

We now argue that the lemma suffices to prove our theorem.

Claim 1. *Lemma 1 implies Theorem 3.*

Proof. Define $\tilde{L} = \lceil \log(1/\tilde{\varepsilon}^2) \rceil \leq L$ (Note that if $\tilde{L} > L$ then we're not interested because then we're not reducing dimensionality!) Fix $x \in T$. We will show

$$\left| \|\Pi x\|^2 - \|x\|^2 \right| < \varepsilon$$

Define $e_r(T) = d(x, T_r)$, and define

$$\tilde{\gamma}_2(T) = \sum_{r=1}^{L} 2^{r/2} \cdot e_r(T).$$

Clearly $\tilde{\gamma}_2(T) \leq \gamma_2(T)$.

Also define

$$z_r = \text{argmin}_{y \in T_r} \|x - y\|_2$$

$$\begin{aligned}
|\|\Pi x\|^2 - \|x\|^2| &\leq |\|\Pi z_{\tilde{L}}\|^2 - \|z_{\tilde{L}}\|^2| + |\|\Pi x\|^2 - \|\Pi z_{\tilde{L}}\|^2| + |\|x\|^2 - \|z_{\tilde{L}}\|^2| \\
&\leq \underbrace{|\|\Pi z_0\|^2 - \|z_0\|^2|}_{\alpha} + \underbrace{|\|\Pi x\|^2 - \|\Pi z_{\tilde{L}}\|^2|}_{\beta} + \underbrace{|\|x\|^2 - \|z_{\tilde{L}}\|^2|}_{\Gamma} \\
&\quad + \underbrace{\sum_{r=1}^{\tilde{L}} (|\|\Pi z_r\|^2 - \|z_r\|^2| - |\|\Pi z_{r-1}\|^2 - \|z_{r-1}\|^2|)}_{\Delta} \qquad (5)
\end{aligned}$$

In class we bounded α and Γ, as the bound on α is simple and the bound on Γ sufficiently captures the proof idea. However, in these notes we bound all of $\alpha, \beta, \Gamma, \Delta$.

Bounding α: We have $\alpha \leq \max\{\tilde{\varepsilon}, \tilde{\varepsilon}^2\} \leq \tilde{\varepsilon}$ by Eq. (2).

Bounding β: We have

$$\begin{aligned}
|\|\Pi x\|^2 - \|\Pi z_{\tilde{L}}\|^2| &= |\|\Pi x\| - \|\Pi z_{\tilde{L}}\|| \cdot |\|\Pi x\| + \|\Pi z_{\tilde{L}}\|| \\
&\leq |\|\Pi x\| - \|\Pi z_{\tilde{L}}\|| \cdot \left(|\|\Pi x\| - \|\Pi z_{\tilde{L}}\|| + 2 \cdot \|\Pi z_{\tilde{L}}\| \right) \\
&= |\|\Pi x\| - \|\Pi z_{\tilde{L}}\||^2 + 2|\|\Pi x\| - \|\Pi z_{\tilde{L}}\|| \cdot \|\Pi z_{\tilde{L}}\| \qquad (6)
\end{aligned}$$

We thus need to bound $|\|\Pi x\| - \|\Pi z_{\tilde{L}}\||$ and $\|\Pi z_{\tilde{L}}\|$. By Eq. (1) we have $\|\Pi z_{\tilde{L}}\| \leq (1 + 2^{\tilde{L}/2}\tilde{\varepsilon}) \leq 2$.

Next, we have

$$\begin{aligned}
|\|\Pi x\| - \|\Pi z_{\tilde{L}}\|| &= |\|\Pi x\| - \|\Pi z_L\| + \|\Pi z_L\| - \|\Pi z_{\tilde{L}}\|| \\
&\leq \|\Pi(x - z_L)\| + \|\Pi(z_L - z_{\tilde{L}})\| \\
&\leq \|\Pi\| \cdot \|x - z_L\| + \| \sum_{r=\tilde{L}+1}^{L} \Pi(z_r - z_{r-1})\|
\end{aligned}$$

$$\leq \|\Pi\| \cdot e_L(T) + \sum_{r=\tilde{L}+1}^{L} \|\Pi(z_r - z_{r-1})\| \tag{7}$$

By Eq. (4), $\|\Pi\| \leq \frac{1}{4}2^{L/2}\tilde{\varepsilon} + 1$. Also by Eq. (1), $\|\Pi(z_r - z_{r-1})\| \leq (1 + 2^{r/2}\tilde{\varepsilon})\|z_r - z_{r-1}\|$. Thus, using $2^{r/2}\tilde{\varepsilon} \geq 1$ for $r > \tilde{L}$,

$$\begin{aligned}
(7) &\leq (\frac{1}{4}2^{L/2}\tilde{\varepsilon} + 1)e_L(T) + \sum_{r=\tilde{L}+1}^{L} (1 + 2^{r/2}\tilde{\varepsilon})\|z_r - z_{r-1}\| \\
&\leq (\frac{1}{4}2^{L/2}\tilde{\varepsilon} + 1)e_L(T) + \sum_{r=\tilde{L}+1}^{L} (1 + 2^{r/2}\tilde{\varepsilon})\|z_r - z_{r-1}\| \\
&\leq \frac{5}{4}2^{L/2}\tilde{\varepsilon}e_L(T) + \sum_{r=\tilde{L}+1}^{L} 2^{r/2+1}\tilde{\varepsilon}\|z_r - z_{r-1}\| \\
&\leq \frac{5}{4}2^{L/2}\tilde{\varepsilon}e_L(T) + 4\sqrt{2}\tilde{\varepsilon} \sum_{r=\tilde{L}+1}^{L} 2^{(r-1)/2} \cdot e_{r-1}(T) \\
&\leq 4\sqrt{2}\tilde{\varepsilon} \sum_{r=\tilde{L}}^{L} 2^{r/2} \cdot e_r(T) \\
&\leq 4\sqrt{2}\tilde{\varepsilon} \cdot \tilde{\gamma}_2(T)
\end{aligned} \tag{8}$$

Thus in summary,

$$\beta \leq (6) \leq 32\tilde{\varepsilon}^2\tilde{\gamma}_2^2(T) + 16\sqrt{2}\tilde{\varepsilon}\tilde{\gamma}_2(T)$$

Bounding Γ: Note $2^{r/2}\tilde{\varepsilon} \geq 1/\sqrt{2}$ for $r \geq \tilde{L}$. Thus

$$\begin{aligned}
|\|x\| - \|z_{\tilde{L}}\|| &\leq e_{\tilde{L}}(T) \\
&\leq \sqrt{2} \cdot 2^{\tilde{L}/2}\tilde{\varepsilon}e_{\tilde{L}}(T) \\
&\leq \sqrt{2}\tilde{\varepsilon} \cdot \tilde{\gamma}_2(T).
\end{aligned}$$

Thus

$$\begin{aligned}
\Gamma &= |\|x\|^2 - \|z_{\tilde{L}}\|^2| \\
&= |\|x\| - \|z_{\tilde{L}}\|| \cdot |\|x\| + \|z_{\tilde{L}}\|| \\
&\leq |\|x\| - \|z_{\tilde{L}}\||^2 + 2|\|x\| - \|z_{\tilde{L}}\|| \cdot \|z_{\tilde{L}}\|| \\
&\leq 2\tilde{\varepsilon}^2 \cdot \tilde{\gamma}_2^2(T) + 2\sqrt{2}\tilde{\varepsilon} \cdot \tilde{\gamma}_2(T)
\end{aligned}$$

Bounding Δ: By the triangle inequality, for any $r \geq 1$

$$\begin{aligned}
|\|\Pi z_r\|^2 - \|z_r\|^2| &= |\|\Pi(z_r - z_{r-1}) + \Pi z_{r-1}\|^2 - \|(z_r - z_{r-1}) + z_{r-1}\|^2| \\
&= |\|\Pi(z_r - z_{r-1})\|^2 + \|\Pi z_{r-1}\|^2 + 2\left\langle \Pi(z_r - z_{r-1}), \Pi z_{r-1} \right\rangle \\
&\quad - \|z_r - z_{r-1}\|^2 - \|z_{r-1}\|^2 - 2\left\langle z_r - z_{r-1}, z_{r-1} \right\rangle|
\end{aligned} \tag{9}$$

$$\leq |\|\Pi(z_r - z_{r-1})\|^2 - \|z_r - z_{r-1}\|^2| + |\|\Pi z_{r-1}\|^2 - \|z_{r-1}\|^2| + 2|\langle \Pi(z_r - z_{r-1}), \Pi z_{r-1}\rangle - \langle z_r - z_{r-1}, z_{r-1}\rangle|. \quad (10)$$

By Eq. (2) we have

$$|\|\Pi(z_r - z_{r-1})\|^2 - \|z_r - z_{r-1}\|^2| \leq \max\{2^{r/2}\tilde{\varepsilon}, 2^r\tilde{\varepsilon}^2\} \cdot 2e_{r-1}^2(T) \leq 2^{r/2+2}\tilde{\varepsilon}e_{r-1}^2(T),$$

with the second inequality holding since $2^{r/2}\tilde{\varepsilon} \leq 1$ for $\leq \tilde{L}$.

By Eq. (3) we also have

$$|\langle \Pi(z_r - z_{r-1}), \Pi z_{r-1}\rangle - \langle z_r - z_{r-1}, z_{r-1}\rangle| \leq 2^{r/2+1}\tilde{\varepsilon}e_{r-1}.$$

Therefore

$$|\|\Pi z_r\|^2 - \|z_r\|^2| - |\|\Pi z_{r-1}\|^2 - \|z_{r-1}\|^2| \leq \tilde{\varepsilon}(2e_{r-1}(T) + 4e_{r-1}^2(T))2^{r/2}$$

Noting $e_r(T) \leq 1$ for all r,

$$\begin{aligned}\Delta &\leq 10\tilde{\varepsilon}\left(\sum_{r=1}^{\tilde{L}} 2^{r/2}e_{r-1}(T)\right)\\ &= 10\sqrt{2}\tilde{\varepsilon}\left(\sum_{r=0}^{\tilde{L}-1} 2^{r/2}e_r(T)\right)\\ &\leq 10\sqrt{2}\tilde{\varepsilon}\tilde{\gamma}_2(T).\end{aligned}$$

Finishing up: We have thus established

$$\begin{aligned}|\|\Pi x\|^2 - \|x\|^2| &\leq \tilde{\varepsilon} + 32\tilde{\varepsilon}^2\tilde{\gamma}_2^2(T) + 16\sqrt{2}\tilde{\varepsilon}\tilde{\gamma}_2(T) + 8\tilde{\varepsilon}^2\tilde{\gamma}_2^2(T) + 2\sqrt{2}\tilde{\varepsilon}\tilde{\gamma}_2(T) + 10\sqrt{2}\tilde{\varepsilon}\tilde{\gamma}_2(T)\\ &= \tilde{\varepsilon} + 28\sqrt{2}\tilde{\varepsilon}\tilde{\gamma}_2(T) + 40\tilde{\varepsilon}^2\tilde{\gamma}_2^2,\end{aligned}$$

which is at most ε for $\tilde{\varepsilon} \leq \varepsilon/(84\sqrt{2}\tilde{\gamma}_2(T))$.

□

3 Doing JL fast

Typically we have some high-dimensional computational geometry problem, and we use JL to speed up our algorithm in two steps: (1) apply a JL map Π to reduce the problem to low dimension m, then (2) solve the lower-dimensional problem. As m is made smaller, typically (2) becomes faster. However, ideally we would also like step (1) to be as fast as possible. So far the dimensionality reduction has been a dense matrix-vector multiplication. So we can ask: can we do better in terms of runtime?

There are two possible ways of doing this: one is to make Π sparse. We saw in pset 1 that this sometimes works: we replaced the AMS sketch with a matrix each of whose columns has exactly 1 non-zero entry. The other way is to make Π structured, i.e., it's still dense but has some structure that allows us to multiply faster. We'll start talking today about sparse JL.

3.1 Sparse JL

One natural way to speed up JL is to make Π sparse. If Π has s non-zero entries per column, then Πx can be multiplied in time $O(s \cdot \|x\|_0)$, where $\|x\|_0 = |\{i : x_i \neq 0\}|$. The goal is then to make s, m as small as possible.

First some history: [2] showed that you can set

$$\Pi_{ij} = \begin{cases} +/\sqrt{m/3} & \text{w.p. } \frac{1}{6} \\ -\sqrt{m/3} & \text{w.p. } \frac{1}{6} \\ 0 & \text{w.p. } \frac{2}{3} \end{cases}$$

and that this gives DJL, even including constant factors. But it provides a factor-3 speedup since in expectation only one third of the entries in Π are non-zero. On the other hand, [5] proved that if Π has independent entries then you can't speed things up by more than a constant factor.

The first people to exhibit a Π without independent entries and therefore to break this lower bound were [3], who got $m = O(\frac{1}{\varepsilon^2}\log(1/\delta))$, $s = \tilde{O}(\frac{1}{\varepsilon}\log^3(1/\delta))$ nonzeros per column of Π. So depending on the parameters this could either be an improvement or not.

Today we'll see [4], which showed that you can take $m = O(\frac{1}{\varepsilon^2}\log(1/\delta))$ and $s = O(\frac{1}{\varepsilon}\log(1/\delta))$, a strict improvement. You do this by choosing exactly s entries in each column of Π to have non-zero entries and then choosing the signs of those entries at random and normalizing appropriately. Alternatively, you can break each column of Π up into s blocks of size m/s, and choose exactly 1 non-zero entry in each block. The resulting matrix is exactly the count sketch matrix.

The analysis uses Hanson-Wright. Previously, for dense Π, we observed that $\Pi x = A_x\sigma$ where A_x was an $m \times mn$ matrix whose i-th row had $x^T/\sqrt{m}$ in the i-th block of size n and zeros elsewhere. Then we said $\|\Pi x\|^2 = \sigma^T A_x^T A_x \sigma$, which was a quadratic form, which allowed us to appeal to HW. We'll do something similar here.

First some notation: we'll write $\Pi_{ij} = \frac{\sigma_{ij}\delta_{ij}}{\sqrt{s}}$ where $\delta_{ij} \in \{0,1\}$ is a random variable indicating whether the corresponding entry of Π was chosen to be non-zero. (So the δ_{ij} are not independent.) For every $r \in [m]$, define $x(r)$ by $(x(r))_i = \delta_{ri}x_i$. The claim is now that $\Pi x = A_x\sigma$ where A_x is an $m \times mn$ matrix whose i-th row contains $x(r)^T/\sqrt{s}$ in the i-th block of size n and zeros elsewhere. This allows us to again use the HW trick: specifically, we observe that $A_x^T A_x$ is a block-diagonal matrix as before. And since we're bounding the difference between $\sigma^T A_x^T A_x \sigma$ and its expectation, it is equivalent to bound $\sigma^T B_x \sigma$ where B_x is $A_x^T A_x$ with its diagonals zeroed out.

Now condition on B_x and recall that HW says that for all $p \geq 1$, $\|\sigma^T B_x\sigma\|_p \leq p\|B_x\| + \sqrt{p}\|B_x\|_F$. Then, taking p-norms with respect to the δ_{ij} and using the triangle inequality, we obtain the bound

$$\|\sigma^T B_x\sigma\|_p \leq p\|\|B_x\|\|_p + \sqrt{p}\|\|B_x\|_F\|_p$$

If we can bound the right-hand-side, we'll obtain required DJL result by application of Markov's inequality, since $\sigma^T B_x\sigma$ is positive. Therefore, it suffices to bound the p-norms with respect to the δ_{ij} of the operator and Frobenius norms of B_x.

We start with the operator norm: since B_x is block-diagonal and its i-th block is $x(r)x(r)^T - \Lambda(r)$ where $\Lambda(r)$ is the diagonal of $x(r)x(r)^T$, we have $\|B_x\| = \frac{1}{s}\max_{1\leq r\leq m}\|x(r)x(r)^T - \Lambda(r)\|$. But the operator norm of the difference of positive-definite matrices is at most the max of either operator norm. Since both matrices have operator norm at most 1, this gives us tht $\|B_x\| \leq 1/s$ always.

So it remains only to bound the Frobenius norm. This is where we'll pick up next time.

References

[1] Samet Oymak, Benjamin Recht, Mahdi Soltanolkotabi. Isometric sketching of any set via the Restricted Isometry Property. *CoRR* abs/1506.03521, 2015.

[2] Dimitris Achlioptas. Database-friendly random projections. *J. Comput. Syst. Sci.* 66(4): 671–687, 2003.

[3] Anirban Dasgupta, Ravi Kumar, Tamás Sarlós. A sparse Johnson: Lindenstrauss transform. *STOC*, pages 341–350, 2010.

[4] Daniel M. Kane, Jelani Nelson. Sparser Johnson-Lindenstrauss Transforms. *Journal of the ACM*, 61(1): 4:1–4:23, 2014.

[5] Jirí Matousek. On variants of the Johnson-Lindenstrauss lemma. *Random Struct. Algorithms*, 33(2): 142–156, 2008.

CS 229r: Algorithms for Big Data Fall 2015

Lecture 14 — October 20, 2015

Prof. Jelani Nelson *Scribe: Rachit Singh*

1 Overview

In the last lecture we discussed how distributional JL implies Gordon's theorem, and began our discussion of sparse JL. We wrote $\|\Pi x\|^2 = \sigma^T A_x^T A_x \sigma$ and bounded the expression using Hanson-Wright in terms of the Frobenius norm.

In this lecture we'll bound that Frobenius norm and then discuss applications to fast nearest neighbors.

2 Sparse JL from last time

Note that we defined $B_x = A_x^T A_x$ as the center of the product from before, but with the diagonals zeroed out. B_x is a block-diagonal matrix with m blocks $B_{x,1}, \ldots, B_{x,r}$ with

$$(B_{x,r})_{i,j} = \begin{cases} 0, & i = j \\ \delta_{r,i}\delta_{r,j}x_i x_j, & i \neq j. \end{cases}$$

2.1 Frobenius norm

We can write the Frobenius norm as

$$\begin{aligned} \|B_x\|_F^2 &= \frac{1}{s^2}\sum_{r=1}^{m}\sum_{i\neq j}\delta_{r_i}\delta_{r_j}x_i^2 x_j^2 \\ &= \frac{1}{s^2}\sum x_i^2 x_j^2 (\sum_{r=1}^{m}\delta_{r_i}\delta_{r_j}) \end{aligned}$$

where we define the expression in the parentheses to be Q_{ij}.

Claim 1.

$$\|Q_{ij}\|_p \lesssim p$$

Let's assume the claim and show that the Frobenius norm is correct.

$$
\begin{aligned}
\|\|B_x\|_F\|_p &= (\mathbb{E}[\|B_x\|_F]^p)^{1/p} \\
&= (((\mathbb{E}[\|B_x\|_F]^2)^{p/2})^{2/p})^{1/2} \\
&= \| \, \|B_x\|_F^2 \, \|_{p/2}^{1/2} \\
&\leq \| \, \|B_x\|_F^2 \, \|_p^{1/2} \\
&= (\| \frac{1}{s^2} \sum_{i \neq j} x_i^2 x_j^2 Q_{ij} \|_p)^{1/2} \\
&\leq \frac{1}{s} \| \sum_{i \neq j} x_i^2 x_j^2 Q_{ij} \|_p^{1/2} \\
&\lesssim \frac{\sqrt{p}}{s} (\sum_{i \neq j} x_i^2 x_j^2)^{1/2} \\
&\leq \frac{\sqrt{p}}{s} \\
&\simeq \frac{\epsilon}{\sqrt{\ln 1/\delta}} \simeq \frac{1}{\sqrt{m}}
\end{aligned}
$$

Now, we can do the following:

$$
\begin{aligned}
\|\|\Pi x\|_2^2 - 1\|_p = \|\sigma^T B_x \sigma\|_p &\leq \sqrt{\frac{p}{m}} + \frac{p}{s} \\
\text{(Markov)} \implies \qquad \mathbb{P}(|\, \|\Pi x\|^2 - 1 \,| > \epsilon) &\leq \frac{\|\|\Pi x\|_2^2 - 1\|_p^p}{\epsilon^p} \\
&\leq 2^p \cdot \left(\frac{\max(\sqrt{\frac{p}{m}}, \frac{p}{s})}{\epsilon} \right)^p < \delta
\end{aligned}
$$

Now we can prove the claim

Proof. Let's just fix column i. It has s nonzero elements somewhere. There's another column j, and the question is how many of the nonzero locations of i match with nonzero elements of j. Let's have Y_t be an indicator random variable for column j having a nonzero element in the tth nonzero row of i (*note*: this is not the tth row of all the elements). Then $Q_{ij} = \sum_{t=1}^{s} Y_t$. If we had independence across the entries, this would just be a Chernoff bound. But we don't, so it isn't.

However, the moments are dominated by the independent case.

$$
\mathbb{E}[\sum_t Y_t]^p = \sum_{s=1}^{\min(p,s)} \sum_{d_1, d_2, \ldots, d_l \sum d_j = p} \sum_{i_1 < i_2 < \ldots < i_l} \mathbb{E}[\prod_{q=1}^{s} Y_{i_q}]
$$

Remember that the expected value of any Y_t is s/n. The product at the end is just $(s/n)^l$ in the

independent case. In our case, it's a conditional product:

$$\begin{aligned}\mathbb{E}\left[\prod_{q=1}^{l} Y_l\right] &= \mathbb{P}(Y_{i_1}=1)\cdot\mathbb{P}(Y_{i_2}=1|Y_{i_1}=1)\dots \\ &= \frac{s}{m}\cdot\frac{s-1}{m-1}\cdots\frac{s-l+1}{m-l+1} \\ &\le (s/m)^l\end{aligned}$$

So the sum is actually dominated by the independent case, which can be handled via Bernstein's inequality. □

Note the runtime to apply the sparse JL map is $O(s \times \text{supp}(x))$

3 Fast JL Transform (FJLT)

Now we'll use a different approach that'll give $O(n \lg n)$ time, which is better in cases where x is dense. This is due to Ailon & Chazelle '09 [AC09]. It is called, as the section title suggests, the FJLT.

Here's their definition of Π:

$$\Pi = \frac{1}{\sqrt{m}}\cdot PHD$$

where P is an $m \times n$ sampling matrix (note that differs slightly from the paper to make the analysis easier). H is $\sqrt{n}$ times an orthogonal $n \times n$ matrix, i.e. $H^T H = n \cdot I$. Also $\max|H_{ij}| = O(1)$, and computing Hx should be fast for any x. D is an $n \times n$ diagonal matrix with random signs $\alpha_1, \dots, \alpha_n$ along the diagonal.

Today we'll let $P = S_\eta$ be an $n \times n$ diagonal matrix where the ith diagonal entry η_i equals 1 with probability m/n and 0 otherwise, and the η_i are independent across i. Note that an example of H could be the unnormalized discrete Fourier transform. Another possibility for H is the unnormalized Hadamard matrix where $H_{i,j} = (-1)^{\langle i,j\rangle}$. Here n is a power of 2 and we are interpreting i, j as elements of $\mathbb{F}_2^{\log_2 n}$. Both of these matrices allow Hx to be computed in time $O(n \log n)$. In general, $n \times n$ matrices F which are orthogonal with $\max_{i,j}|F_{i,j}| = O(1/\sqrt{n})$ are called *bounded orthonormal systems*.

Motivation: what if we just sampled coordinates from x? That would be Px; let $y = (1/\sqrt{m})Px$. Then

$$\mathbb{E}\, y_i^2 = \frac{\|x\|_2^2}{m} = \frac{1}{m} \implies \mathbb{E}\,\|y\|^2 = 1$$

Note that the expected value is good, but the variance is pretty bad: what if all the mass of x is at a single index? We can take intuition from the Heisenberg uncertainty principle, which says that both x and Hx cannot have their mass concentrated on few coordinates.

In [AC09] the following is shown via the Khintchine inequality:

Claim 2.

$$\forall x, \|x\|_2 = 1, \mathop{\mathbb{P}}_{\alpha}\left(\|HDx\|_\infty > c \cdot \sqrt{\frac{\lg(n/\delta)}{n}}\right) < \delta/2$$

If we condition on α so that the event of the above claim holds, then Bernstein implies that for

$$m \geq \frac{\log(1/\delta)\log(n/\delta)}{\epsilon^2},$$

we will have $\|(1/\sqrt{m})PHDx\|_2^2 = (1 \pm \varepsilon)\|x\|_2^2$ with probability $1 - \delta/2$. Thus by a union bound, the overall failure probability is δ.

If we actually want to have $O(\varepsilon^{-2}\log(1/\delta))$ rows, one way to achieve this is to set use the matrix $\Pi' \cdot (1/\sqrt{m})PHD$, where Π' is for example a dense random sign matrix with $m' = O(\varepsilon^{-2}\log(1/\delta))$ rows.

The total time to apply $\Pi' \cdot \Pi$ is then $O(n \log n + m' \cdot m)$.

A slightly different analysis can improve the $\log(n/\delta)$ dependence in m to actually be $\log(m/\delta)$ as follows.

Theorem 3. *Let $x \in \mathbb{R}^n$ be an arbitrary unit norm vector, and suppose $0 < \varepsilon, \delta < 1/2$. Also let $\Pi = S_\eta HD$ as described above with a number of rows equal to $m \gtrsim \varepsilon^{-2}\log(1/\delta)\log(1/(\varepsilon\delta))$. Then*

$$\mathop{\mathbb{P}}_{\Pi}(|\|\Pi x\|_2^2 - 1| > \varepsilon) < \delta.$$

Proof. We use the moment method. Let η' be an independent copy of η, and let $\sigma \in \{-1, 1\}^n$ be uniformly random. Write $z = HDx$ so that $\|\Pi x\|_2^2 = \sum_i \eta_i z_i^2$. Then

$$\begin{aligned}
\|\frac{1}{m}\sum_{i=1}^n \eta_i z_i^2 - 1\|_p &= \|\|\frac{1}{m}\sum_i \eta_i z_i^2 - 1\|_{L^p(\eta)}\|_{L^p(\alpha)} \qquad (1)\\
&= \|\|\frac{1}{m}\sum_i \eta_i z_i^2 - \frac{1}{m}\mathop{\mathbb{E}}_{\eta'}\sum_i \eta_i' z_i^2\|_{L^p(\eta)}\|_{L^p(\alpha)}\\
&\leq \|\|\frac{1}{m}\sum_i z_i^2(\eta_i - \eta_i')\|_{L^p(\eta,\eta')}\|_{L^p(\alpha)} \text{ (Jensen)}\\
&= \|\|\frac{1}{m}\sum_i \sigma_i z_i^2(\eta_i - \eta_i')\|_{L^p(\eta,\eta')}\|_{L^p(\alpha)} \text{ (equal in distribution)}\\
&\leq \frac{2}{m}\cdot\|\|\sum_i \sigma_i \eta_i z_i^2\|_{L^p(\eta)}\|_{L^p(\alpha)} \text{ (triangle inequality)}\\
&\leq \frac{2}{m}\cdot\|\sum_i \sigma_i \eta_i z_i^2\|_p\\
&\lesssim \frac{\sqrt{p}}{m}\cdot\|(\sum_i \eta_i z_i^4)^{1/2}\|_p \text{ (Khintchine)}\\
&\leq \frac{\sqrt{p}}{m}\cdot\|(\max_i \eta_i|z_i|)\cdot(\sum_i \eta_i z_i^2)^{1/2}\|_p\\
&\leq \frac{\sqrt{p}}{m}\cdot\|\max_i \eta_i z_i^2\|_p^{1/2}\cdot\|\sum_i \eta_i z_i^2\|_p^{1/2} \text{ (Cauchy-Schwarz)}
\end{aligned}$$

$$\leq \sqrt{\frac{p}{m}} \cdot \| \max_i \eta_i z_i^2 \|_p^{1/2} \cdot (\| \frac{1}{m} \sum_i \eta_i z_i^2 - 1 \|_p^{1/2} + 1) \text{ (triangle inequality)} \quad (2)$$

We will now bound $\| \max_i \eta_i z_i^2 \|_p^{1/2}$. Define $q = \max\{p, \log m\}$ and note $\| \cdot \|_p \leq \| \cdot \|_q$. Then

$$\begin{aligned}
\| \max_i \eta_i z_i^2 \|_q &= \left(\mathop{\mathbb{E}}_{\alpha,\eta} \max_i \eta_i z_i^{2q} \right)^{1/q} \\
&\leq \left(\mathop{\mathbb{E}}_{\alpha,\eta} \sum_i \eta_i z_i^{2q} \right)^{1/q} \\
&= \left(\sum_i \mathop{\mathbb{E}}_{\alpha,\eta} \eta_i z_i^{2q} \right)^{1/q} \\
&\leq \left(n \cdot \max_i \mathop{\mathbb{E}}_{\alpha,\eta} \eta_i z_i^{2q} \right)^{1/q} \\
&= \left(n \cdot \max_i (\mathop{\mathbb{E}}_{\eta} \eta_i) \cdot (\mathop{\mathbb{E}}_{\alpha} z_i^{2q}) \right)^{1/q} \quad (\alpha, \eta \text{ independent}) \\
&= \left(m \cdot \max_i \mathop{\mathbb{E}}_{\alpha} z_i^{2q} \right)^{1/q} \\
&\leq 2 \cdot \max_i \| z_i^2 \|_q \ (m^{1/q} \leq 2 \text{ by choice of } q) \\
&= 2 \cdot \max_i \| z_i \|_{2q}^2 \\
&\lesssim q \text{ (Khintchine)} \quad (3)
\end{aligned}$$

Eq. (3) uses that H is an unnormalized bounded orthonormal system.

Defining $E = \| \frac{1}{m} \sum_i \eta_i z_i^2 - 1 \|_p^{1/2}$ and combining (1), (2), (3), we find that for some constant $C > 0$

$$E^2 - C\sqrt{\frac{pq}{m}} E - C\sqrt{\frac{pq}{m}} \leq 0,$$

implying $E^2 \lesssim \max\{\sqrt{pq/m}, pq/m\}$. By the Markov inequality

$$\mathbb{P}(|\|\Pi x\|_2^2 - 1| > \varepsilon) \leq \varepsilon^{-p} \cdot E^{2p},$$

and thus to achieve the theorem statement it suffices to set $p = \log(1/\delta)$ then choose $m \gtrsim \varepsilon^{-2} \log(1/\delta) \log(m/\delta)$. □

Remark 4. Note that the FJLT as analyzed above provides suboptimal m. If one desired optimal m, one can instead use the embedding matrix $\Pi'\Pi$,where Π is the FJLT and Π' is, say, a dense matrix with Rademacher entries having the optimal $m' = O(\varepsilon^{-2} \log(1/\delta))$ rows. The downside is that the runtime to apply our embedding worsens by an additive $m \cdot m'$. [AC09] slightly improved this additive term (by an ε^2 multiplicative factor) by replacing the matrix S with a random sparse matrix P.

Remark 5. The usual analysis for the FJLT, such as the approach in [AC09], would achieve a bound on m of $O(\varepsilon^{-2} \log(1/\delta) \log(n/\delta))$. Such analyses operate by, using the notation of the proof

of Theorem 3, first conditioning on $\|z\|_\infty \lesssim \sqrt{\log(n/\delta)}$ (which happens with probability at least $1-\delta/2$ by the Khintchine inequality), then finishing the proof using Bernstein's inequality. In our proof above, we improved this dependence on n to a dependence on the smaller quantity m by avoiding any such conditioning.

3.1 Application: High-dimensional approximate nearest neighbors search (ANN)

Let's assume that we're working with $L2$ distances in $\mathbb{R}^d$. Let's define the *exact nearest neighbors problem* as follows: we're given n points $P = \{p_1, p_2 \dots p_n\}, p_i \in \mathbb{R}^d$. We need to create a data structure such that a query on point $q \in \mathbb{R}^d$ returns a point $p \in P$ such that the distance $\|p-q\|_2$ is minimized. An example application might be image retrieval (*similar images*). In the approximate case, we want to return a point p such that $\|p-q\|_2 \leq c \cdot \min_{p' \in P} \|p'-q\|_2$. Note that the simple solution is to store all the points in a list and just check them all on query, but that requires nd time to calculate.

3.1.1 Voronoi diagrams

One way to solve this problem is to construct the *Voronoi diagram* for the points in the space, which is the division of the space into areas A_i such that all points $x \in A_i$ are closest to p_i. Then on a query we do *planar point location* to find the correct Voronoi cell for a point. However, when $d \neq 2$, the curse of dimensionality strikes. In d dimensions, the Voronoi diagram requires $n^{\theta(d)}$ space to store. Note that this is a lower bound!

3.1.2 Approximate Nearest Neighbor (ANN)

This reduces to the problem c-NN ([HPIM12]).

(c, r)-NN: If there exists $p \in P$ such that $\|p-q\| \leq r$, then return $p' \in P$ such that $\|p'-q\| \leq cr$. If there doesn't exist such a p, then FAIL.

The easiest reduction is just binary search on r, but the above reference avoids some problems.

Space	Time	Ref.
$dn + n^{O(1/\epsilon^2)}$	$(d + \frac{\log n}{\epsilon})^{O(1)}$	[KOR98][IM98]
$dn + n^{1+\rho(c)}$	$n^{\rho(c)}$	
	$\rho(c) =$	Ref.
	$1/c$	[GIM+99]
	$\frac{1}{c^2} + o(1)$	[AI06]
	$(7/8)c^2 + o(1/c^3)$	[AINR14]
	$\frac{1}{(2c^2-1)} + o(1)$	[AR15]
$O(dn)$	$\frac{2.06}{c}$	[MNP07]

Today we just show the following result:

1. ANN with $n^{O(\log(1/\epsilon)/\epsilon^2)}$ space.

2. First, $dn + n \cdot O(c/\epsilon^d)$ space
3. Pretend $r = 1$. Impose uniform $\epsilon/\sqrt{d}$ grid on $\mathbb{R}^d$
4. for each $p_i \in P$, let $B_i = B_{l_2}(p_i, 1)$
5. let B_i' = set of the grid cells that B_i intersects
6. Store $B' = \cup B_i'$ in a hash table (key = grid cell ID, value = i).
7. # of grid cell intersected $\leq Vol(B_{l_2^d}(1+\epsilon)/Vol(\text{grid cell}))$
8. The volumn of the ball is $R^d 2^{O(d)}/d^{d/2}$
9. so we have # of grid cell intersected $\leq (c/\epsilon)^d$

Now note d can be reduced to $O(\varepsilon^{-2} \log n)$ using the JL lemma, giving the desired space bound.

References

[AC09] Nir Ailon and Bernard Chazelle. The fast johnson-lindenstrauss transform and approximate nearest neighbors. *SIAM Journal on Computing*, 39(1):302–322, 2009.

[AI06] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 459–468. IEEE, 2006.

[AINR14] Alexandr Andoni, Piotr Indyk, Huy L Nguyen, and Ilya Razenshteyn. Beyond locality-sensitive hashing. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1018–1028. SIAM, 2014.

[AR15] Alexandr Andoni and Ilya P. Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *STOC*, pages 793–801, 2015.

[GIM+99] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. Similarity search in high dimensions via hashing. In *VLDB*, volume 99, pages 518–529, 1999.

[HPIM12] Sariel Har-Peled, Piotr Indyk, and Rajeev Motwani. Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory of computing*, 8(1):321–350, 2012.

[IM98] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998.

[KOR98] Eyal Kushilevitz, Rafail Ostrovsky, and Yuval Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 614–623, 1998.

[MNP07] Rajeev Motwani, Assaf Naor, and Rina Panigrahy. Lower bounds on locality sensitive hashing. *SIAM Journal on Discrete Mathematics*, 21(4):930–935, 2007.

CS 229r: Algorithms for Big Data Fall 2015

Lecture 15 — October 22, 2015

Prof. Jelani Nelson *Scribe: Brabeeba Wang*

1 Overview

We are going to focus on large sclae linear algebra and today it is on approximation matrix multiplication

1. $A \in \mathbb{R}^{n\times d}$, $B \in \mathbb{R}^{d\times p}$
2. Want to compute $A^T B$

Straight forward algorithm have $O(ndp)$ for loop. Alternatively, we can break out A^T, B into several $d \times d$ blocks and multiply it block by block. So we can use fast square matrix multiplication in $O(d^\omega)$.

1. $\omega < \log_2 7$ (Strassen)
2. $\omega < 2.376$ (Coppersmith, Winograd)
3. $\omega < 2.374$ (Stothevs)
4. $\omega < 2.3728642$ (Vassilevke-Williams)
5. $\omega < 2.3728639$ (Le Gell)

We can also reduce to rectangular matrix multiplication like we can multiply $r \times r^\alpha$ by $r^\alpha \times r$ in $r^{\alpha+o(1)}$ where $\alpha > 0.30298$ (Le Gall). But today we are going to settle for computing $C \in \mathbb{R}^{d\times p}$ such that $\|A^T B - C\|_X$ small. For example $\|\|\|_X = \|\|\|_F$.
Two approaches:

1. Sampling (Drineas, Kannan, Mahoney SIJC'06)[1]
2. JL (Sarlos FOCS'06)[2]

2 Sampling

1. $A^T B = \sum_{k=1}^n a_k b_k^T$
2. $C = (\Pi A)^T(\Pi B), \Pi \in \mathbb{R}^{m\times n}$
3. Π is a sampling matrix with rows $\Pi_1, \cdots, \Pi_m$
4. Π_t are independent across t

5. $\Pi_t = e_i/\sqrt{mp_i}$ with probability p_i proportional to $\|a_i\|_2 \cdot \|b_i\|_2$
6. $C = \sum_{t=1}^{m} 1/m a_{k_t} b_{k_t}^T / p_{k_t}$
7. Define $Z_t = 1/m a_{k_t} b_{k_t}^T / p_{k_t}$

Claim 1. $\mathbb{E}C = A^T B$

Proof. It is trivial by linear expectation since $\mathbb{E}Z_t = A^t B/m$ □

Claim 2. *If $m > 1/\epsilon^2\delta$, then*

$$\mathbb{P}(\|A^T B - (\Pi A)^T(\Pi B)\|_F > \epsilon\|A\|_F\|B\|_F) < \delta \tag{1}$$

Proof. By Markov,

$$\mathbb{P}(\|A^T B - (\Pi A)^T(\Pi B)\|_F > \epsilon\|A\|_F\|B\|_F) < \mathbb{E}(\|A^T B - C\|_F^2)/\epsilon^2\|A\|_F^2\|B\|_F^2 \tag{2}$$

1. WLOG, $\|A\|_F = \|B\|_F = 1$
2. $A^T B - C = \sum_{t=1}^{m}(Z_t - \mathbb{E}(Z_t))$
3. $\mathbb{E}(A^T B - C) = \sum_{i,j}\mathbb{E}(\sum_{t=1}^{m}(Z_t - \mathbb{E}Z_t)_{ij})^2 = \sum_{i,j} Var[\sum_{t=1}^{m}(Z_t)_{ij}] = \sum_{i,j}\sum Var[(Z_t)_{ij}] = m\sum_{i,j} Var[Z_{ij}]$
4. $Z_i j = 1/m\sum_{k=1}^{n}\rho_k/p_k \cdot a_{k_i} b_{k_j}$ where $\rho_\alpha = 1$ if tth row of Π sampled row k.
5. $Var[Z_{ij}] \le \mathbb{E}(Z_{ij})^2 = 1/m^2 \cdot \sum_{k=1}^{n}\mathbb{E}(\rho_k)/p_k^2 \cdot a_{k_i}^2 b_{k_j}^2$
6. $\mathbb{E}\|A^T B - C\|_F^2 \le 1/m\cdot\sum_{k=1}^{n} 1/p_k(\sum_{i,j} a_{k_i}^2 b_{k_j}^2) = 1/m\cdot\sum_{k=1}^{n} 1/p_k\|a_k\|_2^2\|b_k\|_2^2 = 1/m(\sum_{k=1}^{n}\|a_k\|_2^2\|b_k\|_2^2)^2 \le 1/m(\sum_{k=1}^{n}\|a_k\|_2^2)(\sum_{k=1}^{n}\|b_k\|_2^2)$.
7. This gives us $\mathbb{P}(\|A^T B - (\Pi A)^T(\Pi B)\|_F > \epsilon\|A\|_F\|B\|_F) < 1/\epsilon^2 m < \delta$

□

We want to improve the runtime to $\log(1/\delta)$. Following is the trick (Clarkson, Woodruff STOC'09)[3].

1. set $r = \Theta(\log(1/\delta))$
2. Compute $C_1, \cdots, C_r$ as before each with failure probability 1/10
3. Know by Chernoff $> 2/3$ of the C_i give low error $(\epsilon/4\|A\|_F\|B\|_F)$
4. Check which one is good:

```
for i = 1 to r:
    if ||A^T B − C_i||_F ≤ ε/4||A||_F||B||_F :
        return C_i
```

5. Trick:
   for $i = 1$ to r:
      $ctr \leftarrow 0$
      for $j = 1$ to r:
         if $\|C_i - C_j\|_F < \epsilon/2\|A\|_F\|B\|_F$:
            $ctr \leftarrow ctr + 1$
         if $ctr > r/2$
            return C_i

Analysis:

1. # of good i is $> 2r/3$
2. if i is good, then for good j we have $\|C_i - C_j\|_F \leq \|A^TB - C_i\|_F + \|A^TB - C_j\|_F \leq \epsilon/4 + \epsilon/4 \leq \epsilon/2\|A\|_F\|B\|_F$

3 JL Approach

Definition 3. *$\Pi \in \mathbb{R}^{m\times n}$ and D is a distribution over Π satisfies the $(\epsilon, \delta, p)-$JL moment property if for any $x \in S^{n-1}$ we have $\mathbb{E}_{\Pi\sim D}|\|\Pi x\|_2^2 - 1|^p < \epsilon^p\delta$*

Example 4. *1. $\Pi_{ij} = \pm 1/\sqrt{m}$. This induces $(\epsilon, \delta, 2)-$ JL moment property with $m \geq 1/\epsilon^2\delta$ and $(\epsilon, \delta, \log(1/\delta))-$ JL moment property with $m \geq \log(1/\delta)/\epsilon^2$*

2. Based on the pset 1 problem 4, we have $(\epsilon, \delta, 2)-$ JL moment property with $m \geq 1/\epsilon^2\delta$

Claim 5. *Suppose Π comes from $(\epsilon, \delta, p)-$ JL moment property for some $p \geq 2$. Then for any A, B with n rows, we have*

$$\mathbb{P}_{\Pi\sim D}(\|A^TB - (\Pi A)^T(\Pi B)\|_F > \epsilon\|A\|_F\|B\|_F) < \delta \tag{3}$$

Proof.
1. By Markov, $\mathbb{P}_{\Pi\sim D}(\|A^TB - (\Pi A)^T(\Pi B)\|_F > \epsilon\|A\|_F\|B\|_F) < \mathbb{E}\|A^TB - (\Pi A)^T(\Pi B)\|_F^p/\epsilon^p\|A\|_F^p\|B\|_F^p$.
2. So we want to bound $\|\|A^TB - (\Pi A)^T(\Pi B)\|_F^2\|_{p/2}^{1/2}$
3. $\|A^TB - (\Pi A)^T(\Pi B)\|_F^2 = \sum_{i,j}\|a_i\|_2^2\|b_j\|_2^2X_{i,j}^2$
4. $X_{i,j} = \langle \Pi a_i/\|a_i\|_2, \Pi b_j\|b_j\|_2\rangle - \langle a_i/\|a_i\|_2, b_j\|b_j\|_2\rangle$
5. $\|\|A^TB - (\Pi A)^T(\Pi B)\|_F^2\|_{p/2} \leq \sum_{i,j}\|a_i\|_2^2\|b_j\|_2^2\|X_{ij}^2\|_{p/2} \leq \max_{i,j}\|a_i\|_2^2\|b_j\|_2^2\|X_{ij}^2\|_{p/2}$
6. Fix i, j we have $X_{i,j} = \langle \Pi x, \Pi y\rangle - \langle x, y\rangle$ where $\|x\|_2 = \|y\|_2 = 1$
7. $\|X_{ij}^2\|_{p/2} = \|X_{ij}^2\|_p$
8. $X_{ij} = 1/2[(\|\Pi(x-y)\|_2^2) + (\|\Pi x\|_2^2 - 1) + (\|\Pi y\|_2^2 - 1)]$
9. $\|X_{ij}\|_p \leq 1/2[\|x-y\|_2^2\|\frac{\|\Pi(x-y)\|^2}{\|x-y\|^2} - 1\|_p] + \|\|\Pi x\|_2^2 - 1\|_p + \|\|\Pi y\|_2^2 - 1\|_p \leq 1/2\epsilon\delta^{1/p}[\|x-y\|_2^2 + 1 + 1] \leq 3\epsilon\delta^{1/p}$

10. And this gives us $\mathbb{P}_{\Pi\sim D}(\|A^TB-(\Pi A)^T(\Pi B)\|_F > \epsilon\|A\|_F\|B\|_F) \leq \|A\|_F^2\|B\|_F^2 9\epsilon^2\delta^{2/p}/(3\epsilon\|A\|_F\|B\|_F)^p = \delta$

□

4 Next class

We are going to get some results on operator bound. $\|(\Pi A)^T(\Pi B) - A^TB\| < \epsilon\|A\|\|B\|$

1. WLOG $\|A\| = \|B\| = 1$
2. WLOG $A = B$ because consider m as the justaposition of A^T and B. We can easily see that $\|m\| = 1$ And this gives us $\|m\frac{x}{y}\|_2^2 = \|Ax\|^2 + \|By\|^2 \leq \|x\|^2 + \|y\|^2$

And we will get something stronger. From now on $A = B$ and we want for any x, $\|\Pi Ax\|_2^2 = (1\pm\epsilon)\|Ax\|_2^2$. This is stronger because $\|(\Pi A)^T(\Pi A)-A^TA\| = \sup_{\|x\|=1} |\|\Pi Ax\|_2^2-\|Ax\|_2^2| < \epsilon\|Ax\|_2^2$

References

[1] Petros Drineas, Ravi Kannan, Michael Mahoney. Fast Monte Carlo Algorithms for Matrices I: Approximating Matrix Multiplication. *SIAM J. Comput* 36(1):132157, 2006.

[2] Tamas Sarlos. Improved Approximation Algorithms for Large Matrices via Random Projections. *FOCS* 2006.

[3] Kenneth Clarkson, David Woodruff. Numerical Linear Algebra in the Streaming Model. *STOC*, 205–214, 2009.

CS 229r: Algorithms for Big Data Fall 2015

Lecture 16 — Tuesday, Oct 27 2015

Prof. Jelani Nelson *Scribe: Jefferson Lee*

1 Overview

Last class, we looked at methods for approximate matrix multiplication, one based on sampling and another based on JL. The problem was to find a C such that for two matrices $A \in \mathbb{R}^{n\times d}$ and $B \in \mathbb{R}^{n\times p}$:

$$\|A^T B - C\| \leq \epsilon \|A\| \|B\|$$

Where the norms here were Frobenius norms. Then, by analyzing the matrix M that is the horizontal concatenation of A and B, we showed that it is sufficient to be able to find a C such that:

$$\|A^T A - C\| \leq \epsilon \|A\|^2$$

Today, we'll look at operating norms rather than Frobenius norms, and achieve an even stronger bound. In particular, we will be looking at subspace embeddings, and how to use them to get fast algorithms for least squares regression.

2 Subspace embeddings

Definition 1. *Given $E \subset \mathbb{R}^n$ a linear subspace, $\Pi \in \mathbb{R}^{m\times n}$ is an ε-**subspace embedding** for E if*

$$\forall z \in E : (1-\varepsilon)\|z\|_2^2 \leq \|\Pi z\|_2^2 \leq (1+\varepsilon)\|z\|_2^2$$

For us, we can frame these subspace embeddings in a similar light to the approximate matrix multiplication methods:

- $E = \text{colspace}(A) \implies \forall x \in R^d : (1-\epsilon)||Ax||_2^2 \leq ||\Pi Ax||_2^2 \leq (1+\epsilon)||Ax||_2^2$
- $C = D^T D = (\Pi A)^T (\Pi A)$. The previous statement $\implies \forall x \in R^d : |x^T [A^T A - (\Pi A)^T (\Pi A)]x| \leq \epsilon ||Ax||_2^2$. Note that this is a stronger bound than our last - it preserves x.

Note that any linear subspace is the column space of some matrix. From now on, we will represent them as these matrices.

Claim 2. *For any A of rank d, there exists a 0-subspace embedding $\Pi \in \mathbb{R}^{d\times n}$ with $m = d$, but no ε-subspace embedding $\Pi \in \mathbb{R}^{m\times n}$ with $\varepsilon < 1$ if $m < d$.*

Proof. Now, assume there is an ε-subspace embedding $\Pi \in \mathbb{R}^{m \times n}$ for $m < d$. Then, the map $\Pi : E \to \mathbb{R}^m$ has a nontrivial kernel. In particular there is some $x \neq 0$ such that $\Pi x = 0$. But $\|\Pi x\|_2 \geq (1-\varepsilon)\|x\|_2 > 0$, which is a contradiction. For the $m \leq d$ case, first rotate the subspace E to become $\text{span}(e_1, \ldots, e_d)$ (via multiplication by an orthogonal matrix), and then project to the first d coordinates. □

How can we find the orthogonal matrix used in the proof efficiently?

Theorem 3 (Singular value decomposition)**.** *Every $A \in \mathbb{R}^{n \times d}$ of rank r has a "singular value decomposition"*

$$A = U \Sigma V^T$$

where $U \in \mathbb{R}^{n \times r}$ has orthonormal columns, $r = \text{rank}(A)$, $\Sigma \in \mathbb{R}^{r \times r}$ is diagonal with strictly positive entries σ_i (referred to as the singular values) on the diagonal, and $V \in \mathbb{R}^{d \times r}$ has orthonormal columns so $U^T U = I$ and $V^T V = I$

Note that once we have $U \Sigma V^T$, we can set $\Pi = U^T$ (U does not have an influence on the norm). There are algorithms to compute U, Σ, V^T in $O(nd^2)$ time, or the following:

Theorem 4 (Demmel, Dumitru, Holtz [1])**.** *In the setting of the previous theorem, we can approximate SVD well in time $\tilde{O}(nd^{\omega-1})$ where ω is the constant in the exponent of the complexity of matrix multiplication. Here the tilde hides logarithmic factors in n.*

This is still relatively slow. Before considering find subspace embeddings more quickly, let us first consider a potential application.

3 Least squares regression

3.1 Definition and motivation

Definition 5. *Suppose we're given $A \in \mathbb{R}^{n \times d}$, $b \in \mathbb{R}^n$ where $n \gg d$. We want to solve $Ax = b$; however, since the system is over-constrained, an exact solution does not exist in general. In the* ***least squares regression*** *problem, we instead want to solve the equation in a specific approximate sense: we want to compute*

$$x^{LS} = \text{argmin}_{x \in \mathbb{R}^d} \|Ax - b\|_2^2$$

The choice of the function to be optimized is not arbitrary. For example, assume that we have some system, and one of its parameters is a linear function of d other parameters. How can we find the coefficients of that linear function? In reality, we experimentally observe a linear function + some random error. Under certain assumptions - errors have mean 0, same variance, and are independent, then least squares regression is provably the best estimator out of a certain class of estimators (see the Gauss-Markov theorem).

3.2 How do we solve least squares in general?

What's the best x to choose? Notice that $\{Ax : x \in \mathbb{R}^d\}$ is the column span of A. Part of b lives in this column space, and part of it lives orthogonally. Then, the x^{LS} we need is the projection of b on that column span. Let $A = U\Sigma V^T$ be the SVD of A. Then the projection of b satisfies

$$\text{Proj}_{\text{Col}(A)} b = UU^T b$$

hence we can set $x^{LS} = V\Sigma^{-1}U^T b = (A^T A)^{-1}A^T b$ (Assuming A has full column rank - otherwise we need to use the pseudo-inverse). Then we have $Ax^{LS} = U\Sigma V^T V\Sigma^{-1}U^T b = UU^T b$. Thus, we can solve LSR in $O(nd^2)$ time.

3.3 Using subspace embeddings

Claim 6. *If* $||Dx||^2 = (1+\epsilon)||A'x||^2$ *for all* x $A' = [A|b]$ *then if* $\tilde{x}^{LS} = argmin_{x'=[x|-1]}||Dx||_2^2$, *then:*

$$||Ax - b||_2^2 \leq \frac{1+\epsilon}{1-\epsilon}||Ax^{LS} - b||_2^2$$

We're going to replace A with ΠA. Then we just need the SVD of ΠA, which only takes us $O(md^2)$ time. If m is like d then this is faster. However, we still need to find Π and apply it. If you find it using SVD, what's the point? - we already spent a lot of time calculating Π itself.

Claim 7. *(Restatement of last claim) If* Π *is* ε*-s.e. for span*$(cols(A, b))$ *then if* $\tilde{x}^{LS} = \text{argmin}_{|}||\Pi Ax - \Pi b||_2^2$, *then:*

$$||A\tilde{x}^{LS} - b||_2^2 \leq \frac{1+\epsilon}{1-\epsilon}||Ax^{LS} - b||_2^2$$

Proof. $||\Pi A\tilde{x}^{LS} - \Pi b||_2^2 \leq ||\Pi A\tilde{x}^{LS} - \Pi b||_2^2 \leq (1+\epsilon)||Ax^{LS} - b_2^2||$. Similarly for the left side of the inequality. □

The total time to find $\tilde{x}^{LS}$ includes the time to find Π, the time to compute $\Pi A, \Pi b$, and $O(md^2)$ (the time to find the SVD for ΠA). But still, how do we find these Π quickly?

4 Getting Subspace Embeddings

As with approximate matrix multiplication, there are two possible methods we will examine: sampling, and a JL method.

4.1 The Sampling Approach - Leverage Scores

Let $\Pi \in \mathbb{R}^{n\times n}$ be a diagonal matrix with diagonal elements η_i. η_i is 1. if we sample the ith row i of A ($which can be written as a_i^T$), 0 otherwise. $\mathbb{E}[\eta_i] = p_i$.

$$A^T A = \sum_{k=1}^{n} a_k a_k^T$$

$$(\Pi A)^T(\Pi A) = \sum_{k=1}^{n} \frac{\eta_k}{p_k} a_k a_k^T$$

$$\mathbb{E}(\Pi A)^T(\Pi A) = \sum_{k=1}^{n} \frac{\mathbb{E}[\eta_k]}{p_k} a_k a_k^T = A^T A$$

In the last class, when we used the sampling approach for approximate matrix multiplication, we chose proportional to the l_2 norm for each row. We need to decide what p_k should be here. Note that the number of rows you get is non-deterministic. You could set that number, but this approach is easier to numerically analyze. // Before analyzing, some intuition for the probabilities p_k:

- I don't want any p_k's to be 0 - then we just miss a row.
- Define $R_i = \sup_{x \in R^d} \frac{|a_i^T x|^2}{||Ax||_2^2}$. If we don't set $p_k \geq R_k$, it doesn't make sense. Why?
- Look at the event that we did sample row i. Then

$$(\Pi A)^T(\Pi A) = \frac{1}{p_i} a_i a_i^T + \sum_{k \neq i}^{n} \frac{\eta_k}{p_k} a_k a_k^T$$

- Pick the x which achieves the sup in the definition of R_i. Then $x^T(\Pi A)^T(\Pi A)x = \frac{|a_i^T x|^2}{p_i} + \sum(\text{non-negative terms}) \geq \frac{|a_i^T x|^2}{p_i} = ||Ax||_2^2 \frac{R_i}{p_i}$
- If $p_i < R_i/2$, the previous expression evaluates to $2||Ax||_2^2$ - therefore, we are guaranteed to mess up because there is some x which makes our error too big. Therefore, we need some $p_i > R_i/2$.

Definition 8. *Given A, the ith **leverage score** l_i is $l_i = a_i^T(A^T A)^{-1} a_i$, (once again if A has full column rank, otherwise use the pseudo-inverse instead).*

Claim 9. $l_i = R_i$

Proof. Note that both R_i and l_i are both basis independent i.e. if M is square/invertible, then:

- $l_i(A) = l_i(AM)$ and $R_i(A) = R_i(AM)$
- $R_i(AM) = \sup_x \frac{|a_i^T M x|^2}{||AMx||_2^2} = \sup_y \frac{|a_i y t|}{||Ay||_2^2} = R_i(A)$

Choose M s.t. $\tilde{A} = AM$ has orthonormal columns: $M = V\Sigma^{-1}$ (from SVD). Then, wlog $A = \tilde{A} = AM$ and:

- $R_i = \sup_x \frac{|a_i^T x|^2}{||x||_2^2}$

- Which x achieves the sup in R_i? The vector $\|a_i\|$ itself. Thus $R_i = \|a_i\|_2^2$
- $l_i = a_i^T (A^T A)^{-1} a_i = \|a_i\|_2^2$.

□

How do we calculate the leverage scores? Notice that $AM = U$. We sample according to the $||u_i||_2^2$, where U is the matrix of orthonormal columns forming the basis of A. We need the SVD again to actually calculate these probabilities then. But it's actually okay if we sample based on the approximation of the leverage scores. This will be a pset problem - it can be done using JL approach seen next (there are other algorithms as well, such as iterative row sampling).

How do we know it actually works? If we sample according to $||u_i||_2^2$ then:

$$\mathbb{E}[(\text{number of rows sampled})] = \sum_i (p_i) = \sum_i ||u_i||_2^2 = ||U||_F^2 = d$$

Theorem 10 (Drineas, Mahoney, Muthukrishnan [2])**.** *If we choose* $p_i \geq \min\{1, \frac{\lg\,(d/\delta)||u_i||_2^2}{\varepsilon^2}\}$, *then* $\mathbb{P}(\Pi$ *is not* $\varepsilon -$ *s.e. for* $A) < \delta$.

- Note $||u_i||_2^2 = ||UU^T e_i||_2^2 = ||\text{Proj}_A e_i||_2^2 \leq ||e_i||_2^2 = 1$. Thus none of the leverage scores can be bigger than 1, and they sum up to d. The minimum with 1 is needed to the multiplicative factor times the legepave score.
- We can analyze this using Matrix Chernoff (or non-commutative khintchine)

Let's look at the analysis by non-commutative khintchine.

Definition 11. *: The* ***Schatten-p norm*** *of* A *for* $1 \leq p \leq \infty$ *is* $||A||_{S_p} = l_p$*-norm of singular values of* A.

If A has rank $\leq d$, note that $||A||_{S_p} = \Theta(||A||) = ||A||_{S_\infty}$ for $p \geq \lg\, d$ (by Holder's inequality).

Theorem 12 (Lust-Piquard and Pisier [3])**.**

$$\mathbb{E}(||\sum_i \sigma_i A_i||_{S_p}^p)^{1/p}) \leq \sqrt{p}\, \max\{||\sum_i A_i^T A_i||_{S_{p/2}}^{1/2}, ||\sum_i A_i A_i^T||_{S_{p/2}}^{1/2}||\}$$

The total samples we'll need is at most $\frac{d\lg\,(d/\delta)}{\varepsilon^2}$. Note:

- Wanting $||\Pi Ax||_2^2 = ||Ax||_2^2(1 \pm \varepsilon)$ for all x is the same as wanting $||\Pi U \Sigma V^T y||_2^2(1 \pm \varepsilon)$ for all y. Call $\Sigma V^T y = x$.
- thus we want $\forall x, ||\Pi U||_2^2 = (1 \pm \epsilon)||Ux||_2^2 = (1 \pm \varepsilon)||x||_2^2$
- therefore, want $\sup_{||x||_2=1} x^T[(\Pi U)^T(\Pi U) - I]x < \epsilon$, i.e. $||(\Pi U)^T(\Pi U) - I|| < \varepsilon$

4.2 The Oblivious Subspace Embedding Approach

Notice that the columns of U form an orthonormal basis for E.

- We want $\forall x \in E$, $||\Pi x||_2^2 = (1+\epsilon)||x||_2^2$ i.e. $\sup_{x \in E \cap S^{n-1}} |||\Pi x||_2^2 - 1| < \epsilon$. This should look familiar, from Gordon's theorem.
- Gordon: If $\Pi_{i,j} = \pm 1/\sqrt{m}$ then suffices to have $m > \ g^2(T) + 1/\epsilon^2$.

$$\begin{aligned} g(E \cap S^{n-1}) &= \mathop{\mathbb{E}}_{g \in \mathbb{R}^n} \sup_{||x||_2=1} \langle g, Ux \rangle \\ &= \mathop{\mathbb{E}}_{g \in \mathbb{R}^n} \sup_{||x|_2=1} \langle U^T g, x \rangle \\ &= \mathop{\mathbb{E}}_{g' \in \mathbb{R}^d} \sup_{||x|_2=1} \langle g', x \rangle \\ &= ||g'||_2 \\ \leq \mathbb{E}(||g'||_2^2)^{1/2} &= \sum_i (\mathbb{E}(g_i')^2)^{1/2} = \sqrt{d} \end{aligned}$$

Thus, if we take a random Gaussian matrix, by Gordon's theorem, it will preserve the subspace as long as it has at least $\frac{d}{\varepsilon^2}$ rows.

- We want our Π to have few rows. We should be able to find Π quickly. Multiplication with A should be fast.
- Trouble: ΠA takes time $O(mnd)$ using for loops, which takes time $> nd^2$.
- want to use "fast Π" - fast JL or sparse JL.

Possible: (Sarlós [4])

Definition 13. *: An (ϵ, δ, d) **oblivious subspace embedding** is a distribution D over $\mathbb{R}^{m \times n}$ s.t. $\forall U \in \mathbb{R}^{n \times d}$, $U^T U = I$: $\mathbb{P}_{\Pi \sim D}(||(\Pi U)^T(\Pi U) - I|| > \epsilon) < \delta$*

Note that this distribution doesn't depend on A or U. The Gaussian matrix provides an oblivious subspace embedding, but Sarlós showed that other, faster methods, like a fast JL matrix, work as well.

References

[1] James Demmel, Ioana Dumitriu, Olga Holtz, Robert Kleinberg. Fast Matrix Multiplication is Stable. *Numerische Mathematik*, 106 (2) (2007), 199-224

[2] Petros Drineas, Michael W. Mahoney, S. Muthukrishnan. Sampling algorithms for l_2 regression and applications. *SODA 2006*: 1127–1136

[3] Franoise Lust-Piquard and Gilles Pisier. Non commutative Khintchine and Paley inequalities. *Arkiv fr Matematik*, 1991, Volume 29, Issue 1, pp 241-260

[4] Tamas Sarlós. Improved approximation algorithms for large matrices via random projections. *FOCS 2006*, 143–152

CS 229r: Algorithms for Big Data Fall 2015

Lecture 17 — 10/28

Prof. Jelani Nelson *Scribe: Morris Yau*

1 Overview

In the last lecture we defined subspace embeddings a *subspace embedding* is a linear transformation that has the Johnson-Lindenstrauss property for all vectors in the subspace:

Definition 1. *Given $W \subset \mathbb{R}^n$ a linear subspace and $\varepsilon \in (0,1)$, an ε-**subspace embedding** is a matrix $\Pi \in \mathbb{R}^{m \times n}$ for some m such that*

$$\forall x \in W : (1-\varepsilon)\|x\|_2 \leq \|\Pi x\|_2 \leq (1+\varepsilon)\|x\|_2$$

And an oblivious subspace embedding

Definition 2. *An (ϵ, δ, d) oblivious subspace embedding is a distribution D over R^{mxn} such that $\forall U \in R^{mxn}$, $U^T U = I$*

$$P_{\Pi \sim D}(\|(\Pi U)^T(\Pi U)\| > \epsilon) < \delta$$

In this lecture we go over ways of getting oblivious subspace embeddings and then go over applications to linear regression. Finally, time permitting, we will go over low rank approximations.

2 General Themes

Today:

- ways of getting OSE's
- More regression
- Low rank approximation

We can already get OSE's with Gordon's theorem. The following are five ways of getting OSE's

- net argument
- noncommutative kintchine with matrix chernoff
- moment method
- approximate matrix multiplication with Frobenius error
- chaining

2.1 Net Argument

Concerning the net argument which we'll see the details in the pset. For any d-dimensional subspace $E \in R^n$ there exists a set $T \subset E \cap S^{n-1}$ of size $O(1)^d$ such that if Π preserves every $x \in T$ up to $1 + O(\epsilon)$ then Π preserves all of E up to $1 + \epsilon$

So what does this mean, if we have distributional JL than that automatically implies we have an oblivious subspace embedding. We would set the failure probability in JL to be $\frac{1}{O(1)^d}$ which by union bound gives us a failure probability of OSE of δ.

2.2 Noncommutative Khintchine

For Noncommutative Khintchine let $\|M\|_p = (\mathbb{E}\,\|M\|_{S_p}^p)^{\frac{1}{p}}$ with $\sigma_1, ..., \sigma_n$ are $\{1, -1\}$ independent bernoulli. Than

$$\|\sum_i \sigma_i A_i\|_p \leq \sqrt{p}\max\left\{\|(\sum_i A_i A_i^T)^{\frac{1}{2}}\|_p, \|(\sum A_i^T A_i)^{\frac{1}{2}}\|_p\right\}$$

To take the square root of a matrix just produce the singular value decomposition $U\Sigma V^T$ and take the square root of each of the singular values.

Now continuing we want

$$P(\|(\Pi U)^T(\Pi U) - I\| > \epsilon) < \delta$$

. We know the above expression is

$$P(\|(\Pi U)^T(\Pi U) - I\| > \epsilon) < \frac{1}{\epsilon^p}E\|(\Pi U)^T(\Pi U) - I\|^p \leq \frac{C^p}{\epsilon^p}E\|(\Pi U)^T(\Pi U) - I\|_{S_p}^p$$

We want to bound $\|(\Pi U)^T(\Pi U) - I\|_p$ and we know

$$(\Pi U)^T(\Pi U) = \sum_i z_i z_i^T$$

where z_i is the i'th row of ΠU This all implies

$$\|(\Pi U)^T(\Pi U) - I\|_p = \|\sum_i z_i z_i^T - E\sum y_i y_i^T\|_p$$

where $y_i \sim z_i$. Now we do the usual trick with proving bernstein. By convexity we interchange the expectation with the norm and obtain

$$\leq \|\sum_i (z_i z_i^T - y_i y_i^T)\|_p$$

which is just the usual symmetrization trick assuming row of Π are independent. Then we simplify

$$\leq 2\|\sum_i \sigma_i z_i z_i^T\|_{L^p(\sigma,z)} \leq \sqrt{p}\|(\sum_i \|z_i\|_2^2 z_i z_i^T)^{\frac{1}{2}}\|_p$$

This approach of using matrix concentration inequalities has beene used by

The following was observed by Cohen, noncommutative khintchine can be applied to sparse JL

$$m \geq \frac{dpolylog(\frac{1}{\delta})}{\epsilon^2}, s \geq \frac{polylog(\frac{d}{\delta})}{\epsilon^2}$$

but Cohen is able to obtain $m \geq \frac{d\log(\frac{d}{\delta})}{\epsilon^2}, s \geq \frac{\log(\frac{d}{\delta})}{\epsilon}$ for s containing dependent entries as opposed to independent entries. There is a conjecture that the multiplies in $d\log(\frac{d}{\delta})$ is actually an addition. This will have significance in compressed sensing.

2.3 Moment Chernoff

Consider the following combinatorial argument

$$P(\|(\Pi U)^T(\Pi U) - I\| > \epsilon) < \frac{1}{\epsilon^p} E\|(\Pi U)^T(\Pi U) - I\|^p \leq \frac{1}{\epsilon^p} Etr((\Pi U)^T(\Pi U) - I)$$

We know that the trace of an exponentiated matrix is

$$E(tr(B^p)) = \sum_{i_1, i_2, .. i_{p+1}} \prod_{t=1}^{p} B_{i_t i_{t+1}}$$

The rest is just combinatorics.

2.4 $\mathbf{AMM}_F$

For the main result of this section see [6] The basic observation by Nguyen is that

$$\|(\Pi U)^T(\Pi U) - I\| < \|(\Pi U)^T(\Pi U) - I\|_F$$

so what we want is

$$P_\Pi(\|(\Pi U)^T(\Pi U) - I\| > \epsilon) < \delta$$

We know that $U^T U = I$ so this is exactly the form of matrix multiplication discussed two lectures before. So rewriting we obtain

$$P_\Pi(\|(\Pi U)^T(\Pi U) - U^T U\| > \epsilon' \|U\|_F^2) < \delta$$

Where the Frobenius norm of U is d because it's composed of d orthonormal vectors. So we may set $\epsilon = \frac{\epsilon}{d}$ and we need $O(\frac{1}{\epsilon'^2 \delta}) = O(\frac{d}{\epsilon^2 \delta})$ rows.

2.5 Chaining

The basic idea in chaining is to do a more clever net argument than previously discussed. See for example Section 3.2.1 of the Lecture 12 notes on methods of bounding the gaussian width $g(T)$. *Chaining* is the method by which, rather than using one single net for T, one uses a sequence of nets (as in Dudley's inequality, or the generic chaining methodology to obtain the γ_2 bound discussed there).

See [3] by Clarkson and Woodruff for an example of analyzing the SJLT using a chaining approach. They showed it suffices to have $m \geq \frac{d^2 \log^{O(1)}(\frac{d}{\epsilon})}{\epsilon^2}, s = 1$. As we saw above, in later works it was shown that the logarthmic factors are not needed (e.g. by using the moment method, or the AMM_F approach). It would be an interesting exercise though to determine whether the [3] chaining approach is capable of obtaining the correct answer without the extra logarithmic factors.

Note: $s = 1$ means we can compute ΠA in time equal to the number of nonzero entries of A .

3 Other ways to use subspace embeddings

3.1 Iterative algorithms

This idea is due to Tygert and Rokhlin [7] and Avron et al. [2]. The idea is to use gradient descent. The performance of the latter depends on the *condition number* of the matrix:

Definition 3. *For a matrix A, the **condition number** of A is the ratio of its largest and smallest singular values.*

Let Π be a $1/4$ subspace embedding for the column span of A. Then let $\Pi A = U\Sigma V^T$ (SVD of ΠA). Let $R = V\Sigma^{-1}$. Then by orthonormality of U

$$\forall x : \|x\| = \|\Pi ARx\| = (1 \pm 1/4)\|ARx\|$$

which means $AR = \tilde{A}$ has a good condition number. Then our algorithm is the following

1. Pick $x^{(0)}$ such that

   $$\|\tilde{A}x^{(0)} - b\| \leq 1.1\|\tilde{A}x^* - b\|$$

   (which we can get using the previously stated reduction to subspace embeddings with ε being constant).

2. Iteratively let $x^{(i+1)} = x^{(i)} + \tilde{A}^T(b - \tilde{A}x^{(i)})$ until some $x^{(n)}$ is obtained.

We will give an analysis following that in [3] (though analysis of gradient descent can be found in many standard textbooks). Observe that

$$\tilde{A}(x^{(i+1)} - x^*) = \tilde{A}(x^{(i)} + \tilde{A}^T(b - \tilde{A}x^{(i)}) - x^*) = (\tilde{A} - \tilde{A}\tilde{A}^T\tilde{A})(x^{(i)} - x^*),$$

where the last equality follows by expanding the RHS. Indeed, all terms vanish except for $\tilde{A}\tilde{A}^Tb$ vs $\tilde{A}\tilde{A}^T\tilde{A}x^*$, which are equal because x^* is the optimal vector, which means that x^* is the projection of b onto the column span of $\tilde{A}$.

Now let $AR = U'\Sigma' V'^T$ in SVD, then

$$\begin{aligned}
\|\tilde{A}(x^{(i+1)-x^*})\| &= \|(\tilde{A} - \tilde{A}\tilde{A}^T\tilde{A})(x^{(i)} - x^*)\| \\
&= \|U'(\Sigma' - \Sigma'^3)V'^T(x^{(i)} - x^*)\| \\
&= \|(I - \Sigma'^2)U'\Sigma'V'^T(x^{(i)} - x^*)\| \\
&\leq \|I - \Sigma'^2\| \cdot \|U'\Sigma'V'^T(x^{(i)} - x^*)\| \\
&= \|I - \Sigma'^2\| \cdot \|\tilde{A}(x^{(i)} - x^*\| \\
&\leq \frac{1}{2} \cdot \|\tilde{A}(x^{(i)} - x^*)\|
\end{aligned}$$

by the fact that $\tilde{A}$ has a good condition number. So, $O(\log 1/\varepsilon)$ iterations suffice to bring down the error to ε. In every iteration, we have to multiply by AR; multiplying by A can be done in time proportional to the number of nonzero entries of A, $\|A\|_0$, and multiplication by R in time proportional to d^2. So the dominant term in the time complexity is $\|A\|_0 \log(1/\varepsilon)$, plus the time to find the SVD.

3.2 Sarlos' Approach

This approach is due to Sarlós [8]. First, a bunch of notation: let

$$\begin{aligned}
x^* &= \text{argmin}\|Ax - b\| \\
\tilde{x}^* &= \text{argmin}\|\Pi Ax - \Pi b\|. \\
A &= U\Sigma V^T \text{ in SVD} \\
Ax^* &= U\alpha \text{ for } \alpha \in \mathbb{R}^d \\
Ax^* - b &= -w \\
A\tilde{x}^* - Ax^* &= U\beta
\end{aligned}$$

Then, $OPT = \|w\| = \|Ax^* - b\|$. We have

$$\begin{aligned}
\|A\tilde{x}^* - b\|^2 &= \|A\tilde{x}^* - Ax^* + Ax^* - b\|^2 \\
&= \|A\tilde{x}^* - Ax^*\|^2 + \|Ax^* - b\|^2 \text{ (they are orthogonal)} \\
&= \|A\tilde{x}^* - Ax^*\|^2 + OPT^2 = OPT^2 + \|\beta\|^2
\end{aligned}$$

We want $\|\beta\|^2 \leq 2\varepsilon OPT^2$. Since $\Pi A, \Pi U$ have same column span,

$$\begin{aligned}
\Pi U(\alpha + \beta) = \Pi A\tilde{x}^* &= \text{Proj}_{\Pi A}(\Pi b) = \text{Proj}_{\Pi U}(\Pi b) \\
&= \text{Proj}_{\Pi U}(\Pi(U\alpha + w)) = \Pi U\alpha + \text{Proj}_{\Pi U}(\Pi w)
\end{aligned}$$

so $\Pi U\beta = \text{Proj}_{\Pi U}(\Pi w)$, so $(\Pi U)^T(\Pi U)\beta = (\Pi U)^T\Pi w$. Now, let Π be a $(1 - 1/\sqrt[4]{2})$-subspace embedding – then ΠU has smallest singular value at least $1/\sqrt[4]{2}$. Therefore

$$\|\beta\|^2/2 \leq \|(\Pi U)^T(\Pi U)\beta\|^2 = \|(\Pi U)^T\Pi w\|^2$$

Now suppose Π also approximately preserves matrix multiplication. Notice that w is orthogonal to the columns of A, so $U^T w = 0$. Then, by the general approximate matrix multiplication property,

$$\mathop{\mathbb{P}}_{\Pi}\left(\|(\Pi U)^T\Pi w - U^T w\|_2^2 > \varepsilon'^2\|U\|_F^2\|w\|_2^2\right) < \delta$$

We have $\|U\|_F = \sqrt{d}$, so set error parameter $\varepsilon' = \sqrt{\varepsilon/d}$ to get

$$\mathbb{P}\left(\|(\Pi U)^T \Pi w\|^2 > \varepsilon \|w\|^2\right) < \delta$$

so $\|\beta\|^2 \leq 2\varepsilon \|w\|^2 = 2\varepsilon OPT^2$, as we wanted.

So in conclusion, we don't need Π to be an ε-subspace embedding. Rather, it suffices to simply be a c-subspace embedding for some fixed constant $c = 1 - 1/\sqrt{2}$, while also providing approximate matrix multiplication with error $\sqrt{\varepsilon/d}$. Thus for example using the Thorup-Zhang sketch, using this reduction we only need $m = O(d^2 + d/\varepsilon)$ and still $s = 1$, as opposed to the first reduction in these lecture notes which needed $m = \Omega(d^2/\varepsilon^2)$.

References

[1] Noga Alon, Yossi Matias, Mario Szegedy. The Space Complexity of Approximating the Frequency Moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.

[2] Haim Avron and Petar Maymounkov and Sivan Toledo. Blendenpik: Supercharging LAPACK's least-squares solver *SIAM Journal on Scientific Computing*, 32(3) 1217–1236, 2010.

[3] Kenneth L. Clarkson and David P. Woodruff. Low rank approximation and regression in input sparsity time *Proceedings of the 45th Annual ACM Symposium on the Theory of Computing (STOC)*, 81–90, 2013.

[4] James Demmel, Ioana Dumitriu, and Olga Holtz. Fast linear algebra is stable. *Numer. Math.*, 108(1):59-91, 2007.

[5] Xiangrui Meng and Michael W. Mahoney. Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression *Proceedings of the 45th Annual ACM Symposium on the Theory of Computing (STOC)*, 91–100, 2013.

[6] Jelani Nelson and Huy L. Nguyễn. OSNAP: Faster numerical linear algebra algorithms via sparser subspace embeddings. *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2013.

[7] Vladimir Rokhlin and Mark Tygert. A fast randomized algorithm for overdetermined linear least-squares regression. *Proceedings of the National Academy of Sciences*, 105 (36) 13212–13217, 2008.

[8] Tamas Sarlós. Improved approximation algorithms for large matrices via random projections. *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 143–152, 2006.

[9] Mikkel Thorup, Yin Zhang. Tabulation-Based 5-Independent Hashing with Applications to Linear Probing and Second Moment Estimation. *SIAM J. Comput.* 41(2): 293–331, 2012.

CS 229r: Algorithms for Big Data Fall 2015

Lecture 18 — Nov 3rd, 2015

Prof. Jelani Nelson *Scribe: Jefferson Lee*

1 Overview

- Low-rank approximation,
- Compression Sensing

2 Last Time

We looked at three different regression methods. The first was based on ε-subspace embeddings. The second was an iterative approach, building a well-conditioned matrix good for stochastic gradient descent. The third was formulated as follows:

For the least square problem $\min_x \|Sx-b\|_2$, which has optimal solution $x^* = S^+b$, and approximate solution $\tilde{x}^* = \text{argmin}_x\|\Pi Sx - \Pi x\|_2$, we let $x^* = U\alpha$, $w = Sx^* - b$, $U\beta = S\tilde{x}^* - Sx^*$ where $S = U\Sigma V^T$. We proved that $(\Pi U)^T(\Pi U)\beta = (\Pi U)^T\Pi w$ last time.

These results from regression will appear in our work for low-rank approximation.

3 Low-rank approximation

The basic idea is a huge matrix $A \in \mathbb{R}^{n\times d}$ with n, d both very large - say, n users rating d movies. We might believe that the users are linear combinations of a few (k) basic types. We want to discover this low-rank structure. More formally:

Given a matrix $A \in \mathbb{R}^{n\times d}$, we want to compute $A_k := \text{argmin}_{\text{rank}(B)\le k}\|A - B\|_X$.

Some now argue that we should look for a non-negative matrix factorization; nevertheless, this version is still used.

Theorem 1 (Eckart-Young)**.** *Let $A = U\Sigma V^T$ be a singular-value decomposition of A where $\text{rank}(A) = r$ and Σ is diagonal with entries $\sigma_1 \ge \sigma_2 \ge \ldots \ge \sigma_r > 0$, then under $\|\cdot\|_X = \|\cdot\|_F$, $A_k = U_k\Sigma_kV_k^T$ is the minimizer where U_k and V_k are the first k columns of U and V and $\Sigma_k = \text{diag}(\sigma_1, \ldots, \sigma_k)$.*

Our output is then U_k, Σ_k, V_k. We can calculate A_k in $O(nd^2)$ time, by calculating the SVD of A. We would like to do better. First, a few definitions:

Definition 2. $\text{Proj}_A B$ *is the projection of the columns of B onto the* $\text{colspace}(A)$.

Definition 3. *Let $A = U\Sigma V^T$ be a singular decomposition. $A^+ = V\Sigma^{-1}U^T$ is called* Moore-Penrose pseudoinverse *of A.*

3.1 Algorithm

Today we are going to use a sketch which is used both in subspace embedding and approximate matrix multiplication to compute $\tilde{A}_k$ with rank at most k such that $\|A-\tilde{A}_k\|_F \leq (1+\epsilon)\|A-A_k\|_F$, following Sarlós' approach [8]. The first works which got some decent error (like $\varepsilon\|A\|_F$) was due to Papadimitriou [7] and Frieze, Kanna and Vempala [5].

Theorem 4. *Define $\tilde{A}_k = \text{Proj}_{A\Pi^T,k}(A)$. As long as $\Pi \in \mathbb{R}^{m\times n}$ is an $1/2$ subspace embedding for a certain k-dimensional subspace V_k and satisfies approximate matrix multiplication with error $\sqrt{\varepsilon/k}$, then*

$$\|A - \tilde{A}_k\|_F \leq (1 + O(\varepsilon))\|A - A_k\|_F,$$

where $\text{Proj}_{V,k}(A)$ is the best rank k approximation to $\text{Proj}_V(A)$, i.e., projecting the columns of A to V.

Before we prove this theorem, let us first convince ourselves that this algorithm is fast, and that we can compute $\text{Proj}_{A\Pi^T,k}(A)$ quickly. To satisfy the conditions in the above theorem, we know that $\Pi \in \mathbb{R}^{m\times d}$ can be chosen with $m = O(k/\varepsilon)$ e.g. using a random sign matrix (or slightly larger m using a faster subspace embedding). We need to multiply $A\Pi^T$. We can use a fast subspace embedding to compute $A\Pi^T$ quickly, then we can compute the SVD of $A\Pi^T = U'\Sigma'V'^T$ in $O(nm^2)$ time. Let $[\cdot]_k$ denote the best rank-k approximation under Frobenius norm. We then want to compute $[U'U'^TA]_k = U'[U'^TA]_k$. Computing U'^TA takes $O(mnd)$ time, then computing the SVD of U'^TA takes $O(dm^2)$ time. Note that this is already better than the $O(nd^2)$ time to compute the SVD of A, but we can do better if we approximate. In particular, by using the right combination of subspace embeddings, for constant ε the scheme described here can be made to take $O(nnz(A)) + \tilde{O}(ndk)$ time (where $\tilde{O}$ hides $\log n$ factors). We will shoot instead for $O(nnz(A)) + \tilde{O}(nk^2)$.

Consider that:

- We want to compute $\tilde{A}_k = argmin_{X:rank(X)\leq k}\|U'X - A\|_F^2$. If X^+ is the argmin without the rank constraint, then the argmin with the rank constraint is $[U'X^+]_k = U'[X^+]_k$, where $[\cdot]_k$ denotes the best rank-k approximation under Frobenius error.
- Rather than find X^+, we use *approximate regression* to find an approximately optimal $\tilde{X}$. That is, we compute $\tilde{X} = argmin_X \|\Pi'U'X - \Pi'A\|_F^2$ where Π' is an α-subspace embedding for the column space of U' (note U' has rank m). Then our final output is $U'[\tilde{X}]_k$.

Why does the above work? (Thanks to Michael Cohen for describing the following simple argument.) First note

$$\begin{aligned}\left(\frac{1+\alpha}{1-\alpha}\right)\cdot\|U'X^+ - A\|_F^2 &\geq \|U'\tilde{X} - A\|_F^2 \\ &= \|(U'X^+ - A) + U'(\tilde{X} - X^+)\|_F^2 \\ &= \|U'X^+ - A\|_F^2 + \|U'(\tilde{X} - X^+)\|_F^2 \\ &= \|U'X^+ - A\|_F^2 + \|\tilde{X} - X^+\|_F^2\end{aligned}$$

and thus $\|\tilde{X} - X^+\|_F^2 \leq O(\alpha) \cdot \|U'X^+ - A\|_F^2$. The second equality above holds since the matrix U' preserves Frobenius norms, and the first equality since $U'X^+ - A$ has a column space orthogonal to the column space of U'. Next, suppose $f, \tilde{f}$ are two functions mapping the same domain to $\mathbb{R}$ such that $|f(x) - \tilde{f}(x)| \leq \eta$ for all x in the domain. Then clearly $f(argmin_x \tilde{f}(x)) \leq \min_x f(x) + 2\eta$. Now, let the domain be the set of all rank-k matrices, and let $f(Z) = \|U'X^+ - Z\|_F$ and $\tilde{f}(Z) = \|U'\tilde{X} - Z\|_F$. Then $\eta = \|U'X^+ - U'\tilde{X}\|_F = \|X^+ - \tilde{X}\|_F$. Thus

$$\begin{aligned}
\|U'[\tilde{X}]_k - A\|_F^2 &= \|U'[\tilde{X}]_k - U'X^+\|_F + \|(I - U'U'^T)A\|_F^2 \\
&\leq (\|U'[X^+]_k - U'X^+\|_F + 2 \cdot \|X^+ - \tilde{X}\|_F)^2 + \|(I - U'U'^T)A\|_F^2 \\
&\leq (\|U'[X^+]_k - U'X^+\|_F + O(\sqrt{\alpha}) \cdot \|U'X^+ - A\|_F)^2 + \|(I - U'U'^T)A\|_F^2 \\
&= (\|U'[X^+]_k - U'X^+\|_F + O(\sqrt{\alpha}) \cdot \|U'X^+ - A\|_F)^2 + \|U'X^+ - A\|_F^2 \\
&= \|U'[X^+]_k - U'X^+\|_F^2 + O(\sqrt{\alpha}) \cdot \|U'[X^+]_k - U'X^+\|_F \cdot \|U'X^+ - A\|_F \\
&\qquad + O(\alpha) \cdot \|U'X^+ - A\|_F^2 + \|U'X^+ - A\|_F^2 \\
&= \|U'[X^+]_k - A\|_F^2 + O(\sqrt{\alpha}) \cdot \|U'[X^+]_k - U'X^+\|_F \cdot \|U'X^+ - A\|_F \\
&\qquad + O(\alpha) \cdot \|U'X^+ - A\|_F^2 \qquad (1) \\
&\leq (1 + O(\alpha)) \cdot \|U'[X^+]_k - A\|_F^2 + O(\sqrt{\alpha}) \cdot \|U'[X^+]_k - U'X^+\|_F \cdot \|U'X^+ - A\|_F \qquad (2) \\
&\leq (1 + O(\alpha)) \cdot \|U'[X^+]_k - A\|_F^2 + O(\sqrt{\alpha}) \cdot \|U'[X^+]_k - A\|_F^2 \qquad (3) \\
&= (1 + O(\sqrt{\alpha})) \cdot \|U'[X^+]_k - A\|_F^2
\end{aligned}$$

where (1) used that $\|U'[X^+]_k - U'X^+ + U'X^+ - A\|_F^2 = \|U'[X^+]_k - A\|_F^2 + \|U'[X^+]_k - U'X^+\|_F^2$ since $U'X^+ - A$ has columns orthogonal to the column space of U'. Also, (2) used that

$$\|U'X^+ - A\|_F \leq \|U'[X^+]_k - A\|_F,$$

since $U'X^+$ is the best Frobenius approximation to A in the column space of U'. Finally, (3) again used

$$\|U'X^+ - A\|_F \leq \|U'[X^+]_k - A\|_F,$$

and also used the triangle inequality

$$\|U'[X^+]_k - U'X^+\|_F \leq \|U'[X^+]_k - A\|_F + \|U'X^+ - A\|_F \leq 2 \cdot \|U'[X^+]_k - A\|_F.$$

Thus we have established the following theorem, which follows from the above calculations and Theorem 4.

Theorem 5. *Let $\Pi_1 \in \mathbb{R}^{m_1 \times n}$ be a $1/2$ subspace embedding for a certain k-dimensional subspace V_k, and suppose Π_1 also satisfies approximate matrix multiplication with error $\sqrt{\varepsilon/k}$. Let $\Pi_2 \in \mathbb{R}^{m_2 \times n}$ be an α-subspace embedding for the column space of U', where $A\Pi_1^T = U'\Sigma'V'^T$ is the SVD (and hence U' has rank at most m_1). Let $\tilde{A}'_k = U'[\tilde{X}]_k$ where*

$$\tilde{X} = \underset{X}{argmin} \, \|\Pi_2 U'X - \Pi_2 A\|_F^2.$$

Then $\tilde{A}'_k$ has rank k and

$$\|A - \tilde{A}'_k\|_F \leq (1 + O(\varepsilon) + O(\sqrt{\alpha}))\|A - A_k\|_F.$$

In particular, the error is $(1 + O(\varepsilon))\|A - A_k\|_F$ for $\alpha = \varepsilon$.

In the remaining part of these lecture notes, we show that $\mathrm{Proj}_{A\Pi^T,k}(A)$ actually is a good rank-k approximation to A (i.e. we prove Theorem 4). In the following proof, we will denote the first k columns of U and V as U_k and V_k and the remaining columns by $U_{\bar{k}}$ and $V_{\bar{k}}$.

Proof. Let Y be the column span of $\mathrm{Proj}_{A\Pi^T}(A_k)$ and the orthogonal projection operator onto Y as P. Then,

$$\|A - \mathrm{Proj}_{A\Pi^T,k}(A)\|_F^2 \leq \|A - PA\|_F^2 = \|A_k - PA_k\|_F^2 + \|A_{\bar{k}} - PA_{\bar{k}}\|_F^2$$

Then we can bound the second term in that sum:

$$\|A_{\bar{k}}\| = \|(I-P)A_{\bar{k}}\|_F^2 \leq \|A_{\bar{k}}\|_F^1$$

Now we just need to show that $\|A_k - PA_k\|_F^2 \leq \varepsilon\|A_{\bar{k}}\|_F^2$:

$$\begin{aligned}\|A - PA\|_F^2 = \|A_k - (A\Pi^T)(A\Pi^T)^+A_k)\|_F^2 &\leq \|A_k - (A\Pi^T)(A\Pi^T)^+A_k\|_F^2 \\ &= \|A_k^T - A_k^T(\Pi A^T)^+(\Pi A^T)\|_F^2 \\ &= \sum_{i=1}^n \|A_k^{T(i)} - A_k^T(\Pi A^T)^+(\Pi A^T)^{(i)}\|_2^2\end{aligned}$$

Here superscript (i) means the ith column. Now we have a bunch of different approximate regression problems which have the following form:

$$\min_x \|\Pi A_k^T x - \Pi(A^T)^{(i)}\|_2,$$

which has optimal value $\tilde{x}^* = (\Pi A_k^T)^+(\Pi A^T)^{(i)}$. Consider the problem $\min_x \|\Pi A_k^T x - (A^T)^{(i)}\|_2$ as original regression problem. In this case optimal x^* gives $A_k^T x^* = \mathrm{Proj}_{A_k^T}((A^T)^{(i)}) = (A_k^T)^{(i)}$. Now we can use the analysis on the approximate least square from last week.

In our problem, we have a bunch of w_i, β_i, α_i with $S = A_k^T = V_k\Sigma_k U_k^T$ and $b_i = (A^T)^{(i)}$. Here, $\|w_i\|^2 = \|Sx^* - b\|^2 = \|(A_k^T)^{(i)} - (A^T)^{(i)}\|^2$. Hence $\sum_i \|w_i\|^2 = \|A - A_k\|_F^2$. On the other hand, $\sum_i \|\beta_i\|^2 = \|A_k^T - A_k^T(\Pi A_k^T)^+(\Pi A^T)\|_F^2$. Since $(\Pi V_k)^T(\Pi V_k)\beta_i = (\Pi V_k)^T\Pi w_i$, if all singlar values of ΠV_k are at least $1/2^{1/4}$, we have

$$\frac{\sum_i \|\beta_i\|^2}{2} \leq \sum_i \|(\Pi V_k)^T(\Pi V_k)\beta_i\|^2 = \sum_i \|(\Pi V_k)^T\Pi w_i\|^2 = \|(\Pi V_k)^T\Pi W\|_F^T$$

where W has w_i as ith column. What does it look like? $(\Pi V_k)^T\Pi W$ exactly look like approximate matrix multiplication of V_k and W. Since columns of W and V_k are orthogonal, we have $V_k^T W = 0$, hence if Π is a sketch for approximate matrix multiplication of error $\varepsilon' = \sqrt{\varepsilon/k}$, then

$$\mathop{\mathbb{P}}_{\Pi}(\|(\Pi V_k)^T(\Pi W)\|_F^2 > \varepsilon\|W\|_F^2) < \delta$$

since $\|V_k\|_F^2 = k$. Clearly $\|W\|_F^2 = \sum_i \|w_i\|^2 = \|A - A_k\|_F^2$, we get the desired result. □

3.2 Further results

What we just talked about gives a good low-rank approximation but every column of $\tilde{A}_k$ is a linear combination of potentially all columns of A. In applications (e.g. information retrieval), we want a few number of columns be spanning our low dimensional subspace. There has been work on finding fewer columns of A (call them C) such that $\|A - (CC^+A)_k\|_F^2$ is small, but we will not talk about it deeply.

- Boutsidis et al. [1] showed that we can take C with $\approx 2k/\varepsilon$ columns and error $\leq \epsilon\|A - A_k\|_F$.
- Guruswami and Sinop got C with $\leq \frac{k}{\varepsilon} + k - 1$ columns such that $\|A - CC^+A\|_F \leq (1 + \epsilon)\|A - A_k\|_F$.

3.3 K-Means as a Low-Rank Approximation Problem

The k-means problem, which was stated on the problem set, involved a set of points $(x_1, ..., x_n) \in \mathbb{R}^d$. Let A be the matrix with the ith row equal to x_i^T. Given a partition $\mathcal{P}(P_1, ..., P_k)$ of points into k clusters, then the best centroids are averages of the clusters. Define the matrix $X_p \in \mathbb{R}^{n\times k}$ such that:

$$(X_p)_{i,j} = \begin{cases} \frac{1}{\sqrt{|P_j|}} & \text{if} i \in P_j \\ 0 & \text{otherwise} \end{cases}$$

Note that $X_p^T X_p = I$. It can e shown that the ith row of $X_p X_p^T A$ is the centroid of the cluster that X_i belongs to. Thus, solving k-means is equivalent to finding some $\mathcal{P}^* = \text{argmin}\|A - X_p X_p^T A\|$ - this is a constrained rank-k approximation problem. Cohen et. al[3] show that Π can have $m = O(k/\varepsilon^2)$ for a $(1 + \varepsilon)$ approximation, or a $m = O(\lg\, k/\varepsilon^2)$ for a $(9 + \varepsilon)$ approximation (the second bound is specifically for the k-means problem). It is an open problem whether this second bon can get a better approximation.

4 Compressed Sensing

4.1 Basic Idea

Consider $x \in \mathbb{R}^n$. If x is a k sparse vector, we could represent it in a far more compressed manner. Thus, we define a measure of how "compressible" a vector is as a measure of how close it is to being k sparse.

Definition 6. *Let $x_{head(k)}$ be the k elements of largest magnitude in x. Let $x_{tail(k)}$ be the rest of x.*

Therefore, we call x compressible if $\|x_{tail(k)}\|$ is small.

The goal here is to approximately recover x from few linear measurements. Consider we have a matrix Πx such that each the ith row is equal to $\langle \alpha_i, x\rangle$ for some $\alpha_1, ..., \alpha_m \in \mathbb{R}^n$. We want to recover a $\tilde{x}$ from ΠX such that $\|x - \tilde{x}\|_p \leq C_{\varepsilon,p,q}\|x_{tail(k)}\|_q$, where $C_{\varepsilon,p,q}$ is some constant dependent on ε, p and q. Depending on the problem formulation, I may or may not get to choose this matrix Π.

4.2 Approximate Sparsity

There are many practical applications in which approximately sparse vectors appear. Pixelated images, for example, are usually approximately sparse in some basis U. For example, consider an n by n image $x \in \mathbb{R}^{n^2}$. then $x = Uy$ for some basis U, and y is approximately sparse. Thus we can get measurements from $\Pi U y$.

Images are typically sparse in the wavelet basis. We will describe how to transform to the Haar wavelet basis here. Assume that n is a power of two. Then:

1. Break the image x into squares of size four pixels.
2. Initialize a new image, with four regions R_1, R_2, R_3, R_4.
3. Each block of four pixels, b, in x has a corresponding single pixel in each of R_{1b}, R_{2b}, R_{3b}, and R_{4b} based on its location. For each block of four b:
   - Let the b have pixel values p_1, p_2, p_3, and p_4.
   - $R_{1b} \leftarrow \frac{1}{4}(p_1 + p_2 + p_3 + p_4)$
   - $R_{2b} \leftarrow \frac{1}{4}(p_1 - p_2 + p_3 - p_4)$
   - $R_{3b} \leftarrow \frac{1}{4}(p_1 - p_2 - p_3 + p_4)$
   - $R_{4b} \leftarrow \frac{1}{4}(p_1 - p_2 + p_3 - p_4)$
4. Recurse on R_1, R_2, R_3, and R_4.

The general idea is this: usually, pixels are relatively constant in certain regions. Thus, the values in all regions except for the first are usually relatively small. If you view images after this transform, the upper left hand regions will often be closer to white, while the rest will be relatively sparse.

Theorem 7 (Candès, Romberg, Tao [2], Donoho [4])**.** *There exists a* $\Pi \in \mathbb{R}^{m \times n}$ *with* $m = O(k\lg(n/k))$ *and a poly-time algorithm Alg s.t. if* $\tilde{x} = Alg(\Pi x)$ *then* $\|x - \tilde{x}\|_2 \leq O(k^{-1/2})\|x_{tail(k)}\|_1$

If x is actually k-spares, $2k$ measurements are necessary and sufficient. We will see this by examining Prony's method in one of our problem sets, and investigate compressed sensing further next class.

References

[1] Christos Boutsidis, Petros Drineas, Malik Magdon-Ismail. Near Optimal Column-based Matrix Reconstruction. *FOCS*, 305-314, 2011.

[2] Emmanuel J. Candès, Justin K. Romberg, Terence Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489-509, 2006.

[3] Michael B. Cohen, Sam Elder, Cameron Musco, Christopher Musco, Madalina Persu. Dimensionality Reduction for k-Means Clustering and Low Rank Approximation. *STOC*, 163-172, 2015.

[4] David L. Donoho. Compressed sensing, *IEEE Transactions on Information Theory*, 52(4):1289-1306, 2006.

[5] Alan M. Frieze, Ravi Kannan, Santosh Vempala. Fast Monte-carlo Algorithms for Finding Low-rank Approximations. *J. ACM*, 51(6):1025-1041, 2004.

[6] Venkatesan Guruswami, Ali Kemal Sinop. Optimal Column-based Low-rank Matrix Reconstruction. *SODA*, 1207-1214, 2012.

[7] Christos H. Papadimitriou, Prabhakar Raghavan, Hisao Tamaki, Santosh Vempala. Latent Semantic Indexing: A Probabilistic Analysis. *J. Comput. Syst. Sci.*, 61(2):217-235, 2000.

[8] Tamás Sarlós. Improved Approximation Algorithms for Large Matrices via Random Projections. *FOCS*, 143-152, 2006.

CS 229r: Algorithms for Big Data Fall 2015

Lecture 19 — Nov 5

Prof. Jelani Nelson *Scribe: Abdul Wasay*

1 Overview

In the last lecture, we started discussing the problem of compressed sensing where we are given a sparse signal and we want to reconstruct is using as little measurements as possible. We looked into how sparsity manifests itself in images which are approximately sparse in Haar wavelet basis.

In this lecture we will look at

- Recap a bit about compressive sensing.
- RIP and connection to incoherence
- Basis pursuit
- Krahmer-Ward theorem

2 Review from Last time

2.1 Compressed Sensing

In compressed sensing, we are given a "compressible" signal $x \in \mathbb{R}^n$, and our goal is use few linear measurements of x to approximately recover x. Here, a linear measurement of x is its dot product with another vector in $\mathbb{R}^n$. We can arrange m such linear measurements to form the rows of a matrix $\Pi \in \mathbb{R}^{m \times n}$, so the goal now becomes to approximately recover x from Πx using $m \ll n$.

Note that if $m < n$, then any Π has a non-trivial kernel, so we have no hope of exactly recovering every $x \in \mathbb{R}^n$. This motivates our relaxed objective of only recovering *compressible* signals.

So what exactly do we mean by compressible? A compressible signal is one which is (approximately) sparse in some basis – but not necessarily the standard basis. Here an approximately sparse signal is a sum of a sparse vector with a low-weight vector.

2.2 Algorithmic Goals

The compressed sensing algorithms we discuss will achieve the following. If x is actually sparse, we will recover x exactly in polynomial time. And if x is only approximately sparse, then we will recover x approximately, again in poly-time.

More formally, we seek to meet "ℓ_p/ℓ_q guarantees": Given Πx, we will recover $\tilde{x}$ such that

$$\|x - \tilde{x}\|_p \leq C_{k,p,q} \min_{\|y\|_0 \leq k} \|y - x\| q,$$

where the ℓ_0-norm of a vector is the count of its non-zero coordinates. Observe that the minimizer y in the statement above picks out the largest (in absolute value) k coordinates of x and zeroes out the rest of them. Also, note that the right-hand side is zero when x is actually k-sparse.

3 Main Result

Theorem 1. *There is a polynomial-time algorithm which, given Πx for $\Pi \in \mathbb{R}^{m \times n}$ and $x \in \mathbb{R}^n$, can recover $\tilde{x}$ such that*

$$\|x - \tilde{x}\|_2 \leq O\left(\frac{1}{\sqrt{k}}\right) \|x_{tail(k)}\|_1$$

where $x_{tail(k)}$ is x with its top k coordinates zeroed out.

3.1 Exact recovery in the sparse case

As a first step toward proving the theorem, let's examine what we need to recover x exactly when we actually have $\|x\|_0 \leq k$. Information- theoretically, it's necessary and sufficient to have $\Pi x \neq \Pi x'$ whenever $x \neq x'$ are both k-sparse. This is equivalent to requiring any $2k$-sparse vector to lie outside $\ker \Pi$, i.e., requiring each restriction Π_S of Π to the columns in a set S to have full column rank for every $S \subseteq [n]$ with $|S| \leq 2k$.

How can we use this characterization to recover x given $y = \Pi x$ when Π has this property? One way is to find the minimizer z in

$$\begin{aligned} \min_{z \in \mathbb{R}^n} \quad & \|z\|_0 \\ \text{s.t.} \quad & \Pi z = y. \end{aligned}$$

Unfortunately, this optimization problem is NP-hard to solve in general [GJ79, Problem MP5]. In what follows, we will show that with an additional constraint on Π, we can approximately solve this optimization problem using linear programming.

4 RIP Matrices

Definition 2. *A matrix $\Pi \in \mathbb{R}^{m \times n}$ satisfies the (ε, k)- restricted isometry property (RIP) if for all k-sparse vectors x:*

$$(1 - \varepsilon)\|x\|_2^2 \leq \|\Pi x\|_2^2 \leq (1 + \varepsilon)\|x\|_2^2.$$

Equivalently, whenever $|S| \leq k$, we have

$$\|\Pi_S^T \Pi_S - I_k\|_2 \leq \varepsilon.$$

5 Calculating RIP matrices

RIP matrices are obtainable from the following methods:

- Use **JL** to preserve each of the k-dimensional subspace. This can be done by applying JL to the requisite $\binom{n}{k}e^{O(k)}$ vectors yields, by Stirling's approximation,

$$m \lesssim \frac{1}{\varepsilon^2} k \log\left(\frac{n}{k}\right).$$

- Use **incoherent matrices**. The good thing about incoherent matrices is that they are explicit from codes as we have looked in problem set 1.
- From **first principles**. A matrix Π might not satisfy JL, but might still preserve the norms of k-sparse vectors. For example, we can take Π to sample m rows from a Fourier matrix. Recall that for the FJLT, we needed to subsequently multiply by a diagonal sign matrix, but there is no need to do so in the particular case of sparse vectors.

We will focus on the second method.

Recall 3. *A matrix Π is $\alpha-$coherent if:*

- $\|\Pi^i\|_2 = 1$ *for all i, and*
- $|\langle \Pi^i, \Pi^j \rangle| \leq \alpha$ *for all $i \neq j$.*

Claim 4. *Incoherent matrices can be used to explicitly construct RIP.*

We will use the **Gershgorin Circle Theorem** to prove the above claim.

Lemma 5. *Given a matrix A, all its eigenvalues, lie within a complex disk of radius $\sum_{j\neq i} |a_{ij}|$.*

Proof. Let x be an eigenvector of A with corresponding eigenvalue λ. Let i be an index such that $|x_i| = \|x\|_\infty$. Then $(Ax)_i = \lambda x_i$ so

$$\lambda x_i = \sum_{j=1}^{n} a_{ij}x_j \Rightarrow |(\lambda - a_{ii})x_i| = \left|\sum_{j\neq i} a_{ij}x_j\right|$$
$$\Rightarrow |\lambda - a_{ii}| \leq \sum_{j\neq i} \left|\frac{a_{ij}x_j}{x_i}\right| \leq \sum_{j\neq i} |a_{ij}|$$

by our choice of i. □

Theorem 6. *If Π is $(\frac{\varepsilon}{k})$ incoherent, it implies that it is (ε, k)-RIP.*

Proof. Suppose we have an $\alpha-$incoherent matrix where $\alpha = \frac{\varepsilon}{k}$. Let us analyze A=$(\Pi I_s)^T(\Pi I_s)$ for some $|s| \leq k$. Notice that A is a symmetric matrix and it has real eigenvalues. Also, $a_{ii} = 1$ and $a_{ij} = \alpha = \frac{\varepsilon}{k}$. Therefore, the eigenvalues of A lie in the interval of radius:

$$\sum_{j \neq i} |\langle \Pi_S^i, \Pi_S^j \rangle| \leq \alpha(k-1)$$

We set $\alpha = \frac{\varepsilon}{k}$ and get an (ε, k)-RIP matrix. □

6 Basis Pursuit OR RIP to Recovery

Theorem 7. *If Π is $(\varepsilon_{2k}, 2k)$-RIP with $\varepsilon_{2k} \leq \sqrt{2} - 1$, and $\tilde{x} = x + h$ is the solution to the "basis pursuit" linear program*

$$\begin{aligned} \min_{z \in \mathbb{R}^n} \quad & \|z\|_1 \\ \text{s.t.} \quad & \Pi z = \Pi x, \end{aligned}$$

then

$$\|h\|_2 \leq O\left(\frac{1}{\sqrt{k}}\right) \|x_{tail(k)}\|_1.$$

Remark: A *linear program* (LP) is an optimization problem in which one seeks to optimize a linear objective function subject to linear constraints. The above problem is indeed a linear program with polynomially many variables and constraints, since it is equivalent to

$$\begin{aligned} \min_{y \in \mathbb{R}^n} \quad & \sum_i y_i \\ \text{s.t.} \quad & \Pi z = \Pi x, \\ & z_i \leq y_i \quad \forall i, \\ & -z_i \leq y_i \quad \forall i. \end{aligned}$$

It is known (e.g. via Khachiyan's analysis of the ellipsoid method) that LPs can be solved in polynomial time.

We will now present a proof along the lines of [Candes08].

Proof. First, we define some notation.

For a vector $x \in \mathbb{R}^n$ and a set $S \subseteq [n]$, let x_S be the vector with all of its coordinates outside of S zeroed out. We will use T_i^c to indicate the complement of T_i

- Let $T_0 \subseteq [n]$ be the indices of the largest (i absolute value) k coordinates of x.
- Let T_1 be the indices of the largest k coordinates of $h_{T_0^c} = h_{tail(k)}$.
- Let T_2 be the indices of the *second* largest k coordinates of $h_{T_0^c}$.
- ...and so forth, for $T_3, \ldots$

By the triangle inequality, we can write

$$\begin{aligned}\|h\|_2 &= \|h_{T_0 \cup T_1} + h_{(T_0 \cup T_1)^c}\|_2 \\ &\leq \|h_{T_0 \cup T_1}\|_2 + \|h_{(T_0 \cup T_1)^c}\|_2.\end{aligned}$$

Our strategy for bounding h will be to show:

1. $$\|h_{(T_0 \cup T_1)^c}\|_2 \leq \|h_{T_0 \cup T_1}\|_2 + O\left(\frac{1}{\sqrt{k}}\right)\|x_{tail(k)}\|_1.$$
2. $$\|h_{T_0 \cup T_1}\|_2 \leq O\left(\frac{1}{\sqrt{k}}\right)\|x_{tail(k)}\|_1)$$

Both parts rely on the following lemma.

Lemma 8.

$$\sum_{j \geq 2} \|h_{T_j}\|_2 \leq \frac{2}{\sqrt{k}}\|x_{T_0^c}\|_1 + \|h_{T_0 \cup T_1}\|_2.$$

Proof. We first get an upper bound on the left-hand side by applying a technique known as the "shelling trick."

$$\begin{aligned}\sum_{j \geq 2} \|h_{T_j}\|_2 &\leq \sqrt{k} \sum_{j \geq 2} \|h_{T_j}\|_\infty \\ &\leq \frac{1}{\sqrt{k}} \sum_{j \geq 2} \|h_{T_{j-1}}\|_1 \\ &\leq \frac{1}{\sqrt{k}}\|h_{T_0^c}\|_1. \qquad (1)\end{aligned}$$

The first inequality holds because each h_{T_j} is k-sparse, and the second holds because the size of every term in h_{T_j} is bounded from above by the size of every term in $h_{T_{j-1}}$.

Now since $\tilde{x} = x + h$ is the minimizer of the LP, we must have

$$\begin{aligned}\|x\|_1 &\geq \|x + h\|_1 \\ &= \|(x + h)_{T_0}\|_1 + \|(x + h)_{T_0^c}\|_1 \\ &\geq \|x_{T_0}\|_1 - \|h_{T_0}\|_1 + \|h_{T_0^c}\|_1 - \|x_{T_0^c}\|_1\end{aligned}$$

by two applications of the reverse triangle inequality. Rearranging, we obtain

$$\begin{aligned}\|h_{T_0^c}\|_1 &\leq \|x\|_1 - \|x_{T_0}\|_1 + \|h_{T_0}\|_1 + \|x_{T_0^c}\|_1 \\ &= 2\|x_{T_0^c}\|_1 + \|h_{T_0}\|_1 \\ &\leq 2\|x_{T_0^c}\|_1 + \sqrt{k}\|h_{T_0}\|_2 \qquad \text{by Cauchy- Schwarz} \\ &\leq 2\|x_{T_0^c}\|_1 + \sqrt{k}\|h_{T_0 \cup T_1}\|_2\end{aligned}$$

Combining this upper bound with Inequality (1) yields the claim.

□

Returning to the main proof, let us first upper bound the size of $h_{(T_0 \cup T_1)^c}$. We get:

$$\begin{aligned}
\|h_{(T_0\cup T_1)^c}\|_2 &= \left\| \sum_{j\geq 2} h_{T_j} \right\|_2 \\
&\leq \sum_{j\geq 2} \|h_{T_j}\|_2 \\
&\leq \|h_{T_0\cup T_1}\|_2 + \frac{2}{\sqrt{k}} \|x_{T_0^c}\|_1 \qquad \text{by the claim} \\
&= \|h_{T_0\cup T_1}\|_2 + \frac{2}{\sqrt{k}} \|x_{tail(k)}\|_1.
\end{aligned}$$

Now to bound the size of $h_{T_0\cup T_1}$, we need another lemma.

Lemma 9. *If x, x' are supported on disjoint sets T, T' respectively, where $|T| = k$ and $|T'| = k'$, then*

$$|\langle \Pi x, \Pi x' \rangle| \leq \varepsilon_{k+k'} \|x\|_2 \|x'\|_2,$$

where Π is $(\varepsilon_{k+k'}, k+k')$-RIP.

Proof. We can assume WLOG that x, x' are unit vectors. Write

$$\|\Pi x + \Pi x'\|_2^2 = \|\Pi x\|_2^2 + \|\Pi x'\|_2^2 + 2\langle \Pi x, \Pi x' \rangle, \quad \text{and}$$

$$\|\Pi x - \Pi x'\|_2^2 = \|\Pi x\|_2^2 + \|\Pi x'\|_2^2 - 2\langle \Pi x, \Pi x' \rangle.$$

Taking the difference gives

$$\begin{aligned}
|\langle \Pi x, \Pi x' \rangle| &= \frac{1}{4} \left| \|\Pi(x+x')\|_2^2 - \|\Pi(x-x')\|_2^2 \right| \\
&\leq \frac{1}{4} ((1+\varepsilon_{k+k'}) \|x+x'\|_2^2 - (1-\varepsilon_{k+k'}) \|x-x'\|_2^2) \\
&= \frac{1}{4} ((1+\varepsilon_{k+k'})2 - (1-\varepsilon_{k+k'})2) \\
&= \varepsilon_{k+k'}
\end{aligned}$$

since $x \pm x'$ are $(k+k')$-sparse, and x, x' are disjointly supported. This proves the lemma. □

To bound the size of $h_{T_0\cup T_1}$, first observe that

$$\Pi h_{T_0\cup T_1} = \Pi h - \sum_{j\geq 2} \Pi h_{T_j} = -\sum_{j\geq 2} \Pi h_{T_j}$$

since $h \in \ker \Pi$. Therefore,

$$\|\Pi h_{T_0 \cup T_1}\|_2^2 = -\sum_{j \geq 2} \langle \Pi h_{T_0 \cup T_1}, \Pi h_{T_j} \rangle \leq \sum_{j \geq 2} (|\langle \Pi h_{T_0}, \Pi h_{T_j} \rangle| + |\langle \Pi h_{T_1}, \Pi h_{T_j} \rangle|).$$

By Lemma 9, each summand is at most

$$\varepsilon_{2k}(\|h_{T_0}\|_2 + \|h_{T_1}\|_2)\|h_{T_j}\|_2 \leq \varepsilon_{2k}\sqrt{2}\|h_{T_0 \cup T_1}\|_2 \|h_{T_j}\|_2.$$

Thus

$$\begin{aligned}(1 - \varepsilon_{2k})\|h_{T_0 \cup T_1}\|_2^2 &\leq \|\Pi h_{T_0 \cup T_1}\|_2^2 \\ &\leq \varepsilon_{2k}\sqrt{2}\|h_{T_0 \cup T_1}\|_2 \sum_{j \geq 2} \|h_{T_j}\|_2 \\ &\leq \varepsilon_{2k}\sqrt{2}\|h_{T_0 \cup T_1}\|_2 \left(\frac{2}{\sqrt{k}} \|x_{T_0^c}\|_1 + \|h_{T_0 \cup T_1}\|_2 \right)\end{aligned}$$

by Claim 8. Cancelling a factor of $\|h_{T_0 \cup T_1}\|_2$ from both sides and rearranging gives

$$\|h_{T_0 \cup T_1}\|_2 \leq \frac{\varepsilon_{2k} 2\sqrt{2}}{(1 - \varepsilon_{2k} - \varepsilon_{2k}\sqrt{2})\sqrt{k}} \|x_{T_0^c}\|_1 = O\left(\frac{1}{\sqrt{k}}\right) \|x_{tail(k)}\|_1.$$

Putting everything together:

$$\begin{aligned}\|h\|_2 &\leq \|h_{(T_0 \cup T_1)^c}\|_2 + \|h_{T_0 \cup T_1}\|_2 \\ &\leq 2\|h_{T_0 \cup T_1}\|_2 + \frac{2}{\sqrt{k}} \|x_{tail(k)}\|_1 \\ &\leq O\left(\frac{1}{\sqrt{k}}\right) \|x_{tail(k)}\|_1.\end{aligned}$$

□

7 Krahmer and Ward

Theorem 10. *Let $\Pi \in \mathbb{R}^{m \times n}$ be a matrix satisfying the $(\varepsilon, 2k)$-RIP. Let $\sigma \in \{+1, -1\}^n$ be uniformly random and D_σ the diagonal matrix with σ on the diagonal. Then ΠD_σ satisfies the $(O(\varepsilon), 2^{-\Omega(k)})$-distributional JL property.*

This theorem states that given a matrix satisfying the RIP property, we can construct a distribution on matrices satisfying the JL property.

References

[BDFKK11] Jean Bourgain, Stephen Dilworth, Kevin Ford, Sergei Konyagin, and Denka Kutzarova. Breaking the k^2 Barrier for Explicit RIP Matrices. In *STOC*, pages 637–644, 2011.

[Candes08] Emmanuel Candès. The restricted isometry property and its implications for compressed sensing. *C.* R. Acad. Sci. Paris, Ser. I 346:589–592, 2008.

[CRT04] Emmanuel Candès, Justin Romberg, and Terence Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. Inf. Theory*, 52(2):489–509, 2006.

[Don04] David Donoho. Compressed Sensing. *IEEE Trans. Inf. Theory*, 52(4):1289–1306, 2006.

[GJ79] Michael R. Garey, David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* San Francisco, CA: Freeman, 1979.

CS 229r: Algorithms for Big Data Fall 2015

Lecture 20 — November 10, 2015

Prof. Jelani Nelson *Scribe: Yakir Reshef*

1 Recap and overview

Last time we stated a converse to the fact that JL implies RIP, i.e., that RIP implies JL. This is the Krahmer-Ward theorem [2]. Specifically, it says that if Π satisfies $(\varepsilon, 2k)$-RIP and D_σ has random signs on the diagonal then ΠD_σ satisfies $(c\varepsilon, 2^{-c'k})$-DJL.

We started the proof by showing that for any vector x, $\|\Pi D_\sigma x\|_2^2 = \|\Pi D_x \sigma\|^2$, which equals $\sigma^T X \sigma$ where $X = D_x \Pi^T \Pi D_x$. That is where we left off. Today we'll use Hanson-Wright to obtain the final result.

2 Continuing

Without loss of generality $|x_1| \geq \cdots |x_n|$. Let $x_{(i)}$ denote the vector with the k coordinates from i-th block of size k. Finally, let $\Pi_{(i)}$ denote the k columns of Π corresponding to the i-th block of size k.

We'll now partition the n-by-n matrix X into several pieces. First, break X into k-by-k blocks. Let A be the matrix containing just the diagonal blocks. Let B be the matrix containing just the $k-1$ right-most blocks in the top row of blocks. Let B^T denote the transpose of B. And let C denote the rest, i.e., $X - (A + B + B^T)$. Putting all this notation together, we can say, for example, that $A_{(i),(i)} = D_{x(i)} \Pi_{(i)}^T \Pi_{(i)} D_{x(i)}$.

Our proof strategy will be to show that w.p. $1 - 2^{c'k}$ the following hold.

1. $\sigma^T A \sigma \in 1 \pm \varepsilon$
2. $|\sigma^T B \sigma| \leq O(\varepsilon)$
3. $\sigma^T B^T \sigma| \leq O(\varepsilon)$
4. $|\sigma^T C \sigma| \leq O(\varepsilon)$

We begin with $\sigma^T A \sigma$.

$$\begin{aligned}
\sigma^T A \sigma &= \sum_{i=1}^{n/k} \sigma_{(i)} D_{x(i)} \Pi_{(i)}^T \Pi_{(i)} D_{x(i)} \sigma_{(i)} \\
&= \sum_i x_{(i)}^T D_{\sigma(i)} \Pi_{(i)}^T \Pi_{(i)} D_{\sigma(i)} x_{(i)} && \text{(swapping } x \text{ and } \sigma \text{ as before)} \\
&= \sum_i \|\Pi_{(i)} D_{\sigma(i)} x_{(i)}\|^2 \\
&= (1 \pm \varepsilon) \sum_i \|D_{\sigma(i)} x_{(i)}\|_2^2 && \text{(RIP)} \\
&= (1 \pm \varepsilon) \sum_i \|x_{(i)}\|^2 && \text{(Triangle inequality)} \\
&= (1 \pm \varepsilon) \|x\|_2^2 \\
&= (1 \pm \varepsilon)
\end{aligned}$$

Now for B. Let $x_{(-1)} = (x_{(2)}, \ldots, x_{(n/k)})$.

$$\begin{aligned}
B &= D_{x(1)} \Pi_{(1)}^T \Pi_{(-1)} D_{x(-1)} \\
\Rightarrow \sigma^T B \sigma &= \sigma_{(1)} D_{x(1)} \Pi_{(1)}^T \Pi_{(-1)} D_{x(-1)} \sigma_{(-1)}
\end{aligned}$$

Let $v = \sigma_{(1)} D_{x(1)} \Pi_{(1)}^T \Pi_{(-1)} D_{x(-1)}$. We will prove the bound on $\sigma^T B \sigma$ by first bounding v as follows.

Claim 1. $\|v\|_2 = O(\varepsilon/\sqrt{k})$.

Proof. $\|v\|_2 = \sup_{\|y\|=1} \langle v, y \rangle$. And

$$\begin{aligned}
\langle v, y \rangle &= \sigma_{(1)} D_{x(1)} \Pi_{(1)}^T \Pi_{(-i)} \Pi_{(-1)} D_{x(-1)} y_{(-1)} \\
&= \sum_{j>1} \sigma_{(1)} D_{x(1)} \Pi_{(1)}^T \Pi_{(j)} D_{x(j)} y_{(j)} \\
&= \sum_{j>1} x_{(1)} D_{\sigma(1)} \Pi_{(1)}^T \Pi_{(j)} D_{x(j)} y_{(j)} \\
&\leq \sum_{j>1} \|x_{(1)}\|_2 \cdot \|D_{\sigma(1)}\| \cdot \|\Pi_{(1)}^T \Pi_{(j)}\| \|D_{x(j)}\| \|y_{(j)}\|_2
\end{aligned}$$

Now we know trivially that $\|x_{(1)}\|_2 \leq 1$ and that the operator norm of D_σ equals 1. Also, by RIP we know that $\Pi_{(1)} \Pi_{(j)} \leq O(\varepsilon)$ since $j \neq 1$. Finally, $\|D_{x(j)}\|$ is just $\|x\|_\infty$. Putting this together, we

get

$$\begin{aligned}
\langle v, y\rangle &\leq \sum_{j>1} O(\varepsilon)\|x_{(j)}\|_\infty \|y_{(j)}\|_2 \\
&\leq \sum_{j>1} O(\varepsilon)\left(\frac{\|x_{(j-1)}\|_1}{k}\right)\|y_{(j)}\|_2 && \text{(shelling)} \\
&\leq \sum_{j>1} O(\varepsilon)\left(\frac{\|x_{(j-1)}\|_2}{\sqrt{k}}\right)\|y_{(j)}\|_2 && \text{(Cauchy-Schwarz)} \\
&\leq \frac{O(\varepsilon)}{\sqrt{k}} \sum_{j>1} \|x_{(j-1)}\|_2 \cdot \|y_{(j)}\|_2 \\
&\leq \frac{O(\varepsilon)}{\sqrt{k}} \sum_{j>1} \left(\|x_{(j-1)}\|_2^2 + \|y_{(j)}\|_2^2\right) && \text{(AM-GM)} \\
&\leq \frac{O(\varepsilon)}{\sqrt{k}}
\end{aligned}$$

□

Going back to our proof about $\sigma^T B\sigma$, we then see that

$$\begin{aligned}
P(|\sigma^T B\sigma| > c\varepsilon) &= P_{\sigma_{(-1)}}(|v^T\sigma| > c\varepsilon) \\
&\lesssim e^{-c'\varepsilon^2/\|v\|_2^2} && \text{(Khintchine)} \\
&= 2^{-\Omega(k)}
\end{aligned}$$

So now the only thing left is to address C. We do this using Hanson-Wright, noting that C has no non-zero diagonal entries which means that $E(\sigma^T C\sigma) = 0$. So H-W gives us that

$$P(|\sigma^T C\sigma| > \lambda) \lesssim e^{-c'\lambda^2/\|C\|_F^2} + e^{-c'\lambda/\|C\|}$$

setting $\lambda = c\varepsilon$ yields the result, provided we bound the norms. We do so below.

$$\begin{aligned}
\|C\|_F^2 &= \sum_{i,j} \|D_{x(i)}\Pi_{(i)}^T \Pi_{(j)} D_{x(j)}\|_F^2 \\
&\leq \sum_{i\neq j} \left(\|D_{x(i)}\| \cdot \|\Pi_{(i)}^T\Pi_{(j)}\| \cdot \|D_{x(j)}\|_F\right)^2 && (\|AB\|_F \leq \|A\| \cdot \|B\|_F) \\
&\leq O(\varepsilon^2) \sum_{i\neq j} \|x(i)\|_\infty^2 \cdot \|x(j)\|_2^2 && \text{(RIP since } i \neq j\text{, and } D \text{ matrices are diagonal)} \\
&\leq \frac{O(\varepsilon^2)}{k} \sum_{i\neq j} \|x_{(i-1)}\|_2^2 \cdot \|x_{(j)}\|_2^2 && \text{(shelling + norm inequalities)} \\
&\leq \frac{O(\varepsilon^2)}{k} \left(\sum_i \|x_{(i)}\|_2^2\right)^2 && \text{(monomials a superset of the terms in previous line)} \\
&\leq \frac{O(\varepsilon^2)}{k}
\end{aligned}$$

$$
\begin{aligned}
\|C\| &= \sup_{\|y\|_2=1} |y^T C y| \\
&= \left| \sum_{i \neq j} y_{(i)} D_{x(i)} \Pi_{(i)}^T \Pi_{(j)} D_{x(j)} y_{(j)} \right| \\
&\leq \left| \sum_{i \neq j} \|y_{(i)}\|_2 \cdot \|D_{x(i)}\| \cdot \|\Pi_{(i)}^T \Pi_{(j)}\| \cdot \|D_{x(j)}\| \cdot \|y_{(j)}\|_2 \right| \\
&\leq \frac{O(\varepsilon)}{k} \sum_{i \neq j} \|y_{(i)}\|_2 \cdot \|x_{(i-1)}\|_2 \cdot \|x_{(j-1)}\|_2 \cdot \|y_{(j)}\|_2 && \text{(shelling twice)} \\
&\leq \frac{O(\varepsilon)}{k} \left(\sum_{i>1} \|y_{(i)}\|_2 \cdot \|x_{(i-1)}\|_2 \right)^2 \\
&\leq \frac{O(\varepsilon)}{k} \left(\sum_{i} \frac{1}{2} (\|y_{(i)}\|_2^2 + \|x_{(i-1)}\|_2^2 \right)^2 && \text{(AM-GM)} \\
&= O\left(\frac{\varepsilon}{k}\right)
\end{aligned}
$$

Plugging these bounds into H-W completes the proof of Krahmer-Ward.

Before we leave the topic, we observe that K-W can be used to prove "Gordon-like" theorems, because we can construct a matrix SH that is RIP across many different scales simultaneously. K-W then gives us that SHD_σ is DJL, and the fact that DJL implies "Gordon-like" theorems can then be used to obtain the result.

3 Sparse reconstruction faster than basis pursuit?

We showed that basis pursuit can give ℓ_2/ℓ_1 sparse signal recovery. But basis pursuit can be slow if, e.g. we're looking at an image with 1M pixels. It turns out there are faster ways to do it. The algorithms are mostly iterative. Today we'll look at *iterative hard thresholding*, due to [1].

Let's first write down the algorithm. The algorithm takes as input (y, Π, T) where y is the encoded signal, T is an iteration count, and $y = \Pi x + e$ where e denotes an error term. Let $H_k(z)$, the "hard-thresholding operator", restrict z to its largest k entries in magnitude. The algorithm is:

1. $x^{[1]} \leftarrow 0$.
2. For $i = 1$ to T:
   (a) $x^{[i]} \leftarrow H_k(x^{[i]} + \Pi^T(y - \Pi x^{[i]}))$
3. return $x^{[T+1]}$.

We observe that this algorithm makes sense if we pretend that $e = 0$ and that $\Pi^T \Pi = I$. Because then $\Pi^T(y - \Pi x^{[i]}) = x - x^{[i]}$.

Theorem 1. *If Π satisfies $(\varepsilon, 3k)$-RIP for $\varepsilon < 1/(4\sqrt{2})$, then*

$$\|x^{[T]} - x\|_2 \lesssim 2^{-T}\|H_k(x)\|_2 + \|x - H_k(x)\|_2 + O\left(\frac{1}{\sqrt{k}}\right) \cdot \|x - H_k(x)\|_1 + \|e\|_2$$

Proof. Let x^k denote $H_k(x)$. We will assume that $x = x^k$; the rest we put into the noise. As formal justification, we write $y = \Pi x + e$ as $y = \Pi(x_k + x - x_k) + e = \Pi x_k + \tilde{e}$ for $\tilde{e} = e + \Pi(x - x_k)$. Now note $\|\tilde{e}\|_2 \le \|e\|_2 + \|\Pi(x - x_k)\|_2$. Then define S_1 as the coordinates of the largest k entries (in magnitude) of x, S_2 the next k largest, etc. Then

$$\begin{aligned}
\|\Pi(x - x_k)\|_2 &= \|\Pi \sum_{j>1} x_{S_j}\|_2 \\
&\le \sum_{j>1} \|\Pi x_{S_j}\|_2 \\
&\le \sqrt{1+\varepsilon} \cdot \left(\|x_{S_2}\|_2 + \sum_{j>2} \|x_{S_j}\|_2 \right) \quad \text{(by RIP)} \\
&\le \sqrt{1+\varepsilon} \cdot \left(\|x_{S_2}\|_2 + \sum_{j>2} \|x_{S_j}\|_\infty \cdot \sqrt{k} \right) \\
&\le \sqrt{1+\varepsilon} \cdot \left(\|x_{S_2}\|_2 + \sum_{j>2} \|x_{S_{j-1}}\|_1 \cdot \frac{1}{\sqrt{k}} \right) \quad \text{(shelling)} \\
&\le \sqrt{1+\varepsilon} \cdot \left(\|x - x^k\|_2 + \frac{1}{\sqrt{k}} \|x - x^k\|_1 \right)
\end{aligned}$$

Hence, going forward we can assume $x = x^k$.

Define $r^{[t]} = x^k - x^{[t]}$, and define $a^{[t+1]} = x^{[t]} + \Pi^T(y - x^{[t]})$. This implies that $x^{[t+1]} = H_k(a^{[t+1]})$. We define $\Gamma^*_k = \text{support}(x^k) \subset [n]$, and $\Gamma^{[t]} = \text{support}(x^{[t]})$, and $B^{[t]} = \Gamma^*_k \cup \Gamma^{[t]}$.

What we want is

$$\begin{aligned}
\|x - x^{[t+1]}\|_2 &= \|x^k_{B^{[t+1]}} - x^{[t+1]}_{B^{[t+1]}}\|_2 \\
&\le \|x_{B^{[t+1]}} - a_{B^{[t+1]}}\|_2 + \|a_{B^{[t+1]}} - x^{[t+1]}_{B^{[t+1]}}\|_2 \\
&\le 2\|x_{B^{[t+1]}} - a_{B^{[t+1]}}\|_2 \qquad \text{(definitions)} \\
&= 2\|x_{B^{[t+1]}} - x^{[t]}_{B^{[t+1]}} - (\Pi^T(y - \Pi x^{[t]}))_{B^{[t+1]}}\|_2 \\
&= 2\|r^{[t]}_{B^{[t+1]}} - (\Pi^T \Pi r^{[t]})_{B^{[t+1]}} - \Pi_{B^{[t+1]}} e\|_2 \\
&= 2\|r^{[t]}_{B^{[t+1]}} - \Pi^T_{B^{[t+1]}} \Pi r^{B^{[t+1]}} - \Pi^T_{B^{[t]} - B^{[t+1]}} \Pi r^{B^{[t]} - B^{[t+1]}} - \Pi_{B^{[t+1]}} e\|_2 \\
&\qquad\qquad (r = r_{B^{[t+1]}} + r_{B^{[t]} - B^{[t+1]}}) \\
&\le 2\|I - \Pi^T_{B^{[t+1]}} \Pi_{B^{[t+1]}}\| \cdot \|r^{[t]}_{B^{[t+1]}}\|_2 + 2\|\Pi^T_{B^{[t+1]}} \Pi_{B^{[t]} - B^{[t+1]}}\| \cdot \|r^{[t]}_{B^{[t]} - B^{[t+1]}}\|_2 + \|\Pi_{B^{[t+1]}}\| \cdot \|e\|_2 \\
&\le 2\varepsilon \|r^{[t]}_{B^{[t+1]}}\|_2 + 2\varepsilon \|r^{[t]}_{B^{[t]} - B^{[t+1]}}\|_2 + 2\sqrt{1+\varepsilon}\|e\|_2 \\
&= 2\varepsilon(\|r^{[t]}_{B^{[t+1]}}\|_2 + \|r^{[t]}_{B^{[t]} - B^{[t+1]}}\|_2) + 2\sqrt{1+\varepsilon}\|e\|_2
\end{aligned}$$

Now we recall that when x and y have disjoint support, we have in general that $\|x\|_2 + \|y\|_2 \leq \sqrt{2}\|x+y\|_2$. This means that our bound is at most

$$2\sqrt{2}\varepsilon\|r^{[t]}\|_2 + 2\sqrt{1+\varepsilon}\|e\|_2 \leq \frac{1}{2}\|r^{[t]}\|_2 + 3\|e\|_2$$

and so by choosing our constants correctly we can make the induction on t go through. □

References

[1] Thomas Blumensath, Mike E. Davies A simple, efficient and near optimal algorithm for compressed sensing. *ICASSP*, 2009.

[2] Felix Krahmer, Rachel Ward. New and improved Johnson-Lindenstrauss embeddings via the Restricted Isometry Property. *SIAM J. Math. Anal.*, 43(3):1269–1281, 2011.

CS 229r: Algorithms for Big Data Fall 2015

Lecture 21 — November 12, 2015

Prof. Jelani Nelson *Scribe: Zezhou Liu*

1 Overview

Previously, we looked at basis pursuit and *iterative hard thresholding (IHT)* for ℓ_2/ℓ_1 sparse signal recovery. Recall that from before the ℓ_2/ℓ_1 guarantee: time per iteration depends on matrix-vector multiplication time by Π, Π^T in the IHT algorithm. To make this fact, the only way we know is to use a fast such Π, such as sampling rows from the Fourier matrix. However, for those Π we do not know how to get RIP with only $O(k\log(n/k))$ sampled rows (the best known proof currently requires an extra factor of $\log^2 k$ in the number of measurements).

Today, we take a look at ℓ_1/ℓ_1 sparse signal recovery using expanders. Our goal is to get a good recovery guarantee using $O(k\log(n/k))$ measurements with fast time per iteration in an iterative algorithm; for this though we will relax from achieving the ℓ_2/ℓ_1 guarantee to achieving the weaker ℓ_1/ℓ_1 guarantee (as you will show on the current problem set, ℓ_1/ℓ_1 is indeed a weaker guarantee). Specifically, we aim for the guarantee of finding $\tilde{x}$ such that

$$\|x - \tilde{x}\| \leq C \cdot \|x_{tail(k)}\|_1 \tag{1}$$

2 RIP_1 matrices for signal recovery

Definition 1. *A matrix Π satisfies the (ε, k)-RIP_1 property if for all k-sparse vectors x*

$$(1-\varepsilon)\|x\|_1 \leq \|\Pi x\|_1 \leq \|x\|_1$$

Definition 2. *Let $G = (U, V, E)$ be a left d-regular bipartite graph (every node on left is adjacent to d nodes on the right) with left vertices U, right vertices V and edges E. The graph G is a (k, ε)-expander if for all $S \subseteq U$ where $|S| \leq k$:*

$$|\Gamma(S)| \geq (1-\varepsilon)d|S|.$$

Here, $\Gamma(S)$ denotes the neighborhood of S. Intuitively, this means that no matter what S you choose (as long as it is relatively small $|S| \leq k$), you won't have too many collisions amongst the neighbors of vertices in S.

Claim 3. *There exist d-regular (k, ε)-expanders satisfying*

- $n = |U|$
- $m \lesssim |V| = \mathcal{O}\left(\frac{k}{\varepsilon^2}\log\left(\frac{n}{k}\right)\right)$

- $d \leq \mathcal{O}\left(\frac{1}{\varepsilon^2}\log\left(\frac{n}{k}\right)\right)$

This claim can be proven by picking G at random with the specified n, d, m, then showing that G satisfies the expansion condition with high probability (by union bounding over all $S \subset [n]$ of size at most k). Thus, this approach is not constructive, although it does provide a simple Monte Carlo randomized algorithm to get a good expander with high probability.

In fact [4] shows that you can construct d-regular (k, ε)-expanders deterministically with:

- $n = |U|$
- $m = |V| = \mathcal{O}\left(d^2 k^{1+\alpha}\right)$
- $d = \mathcal{O}\left(\frac{1}{\varepsilon}\log(k)\log(n)\right)^{1+\frac{1}{\varepsilon}}$

where α is an arbitrarily small constant.

Given a d-regular (k, ε)-expander G, we can construct the following $\Pi = \Pi_G$:

$$\Pi = \Pi_G = \frac{1}{d} A_G$$

where $A_G \in \mathbb{R}^{m \times n}$ is the bipartite adjacency matrix for the expander G and each column of A_G contains exactly d non-zero (1) entries and the rest zeros. Then, $\Pi_G \in \mathbb{R}^{|V| \times |U|}$ has exactly d non-zero entries (equal to $1/d$) in each column. The average number of non-zero entries per row will be nd/m; and indeed, if Π is picked as a random graph as above then each row will have $O(nd/m)$ non-zeroes with high probability. Thus Π is both row-sparse and column-sparse.

Theorem 4 ([1, Theorem 1])**.** *If G is a d-regular (k, ε)-expander, then Π_G is $(2\varepsilon, k)$-RIP$_1$.*

Definition 5. *The matrix Π satisfies the "C-restricted nullspace property of order k" if for all $\eta \in \mathrm{Ker}(\Pi)$ and for all $S \subseteq [n]$ where $|S| = k$*

$$\|\eta\|_1 \leq C\|\eta_{\bar{S}}\|_1.$$

It is known that if Π satisfies $(C, 2k)$-RNP, then for small enough C, if $\tilde{x}$ is the solution form basis pursuit, then

$$\|x - \tilde{x}\|_1 \leq O(1) \cdot \|x_{tail(k)}\|_1$$

For example see the proof in [5].

Note the above is not nice for a few reasons. First, it is an abstraction violation! We would like to say RIP$_1$ alone suffices for basis pursuit to give a good result, but unfortunately one can come up with a counter-example showing that's not true. For example, Michael Cohen provided the counter example where Π is $n \times (n-1)$ with ith column e_1 for $i = 1, \ldots, n-1$, and the last column is $(n-1)^{-1}(1, \ldots, 1)$. This Π is RIP$_1$ with good constant even for $k \in \Theta(n)$, but it does not have the restricted nullspace property (consider how it acts on the vector $(1, 1, \ldots, 1, -(n-1))$ in its kernel).

Second, it is slow: we are trying to avoid solving basis pursuit (we could already solve basis pursuit with subgaussian RIP matrices with $O(k\log(n/k))$ rows that provide us with the stronger ℓ_2/ℓ_1 guarantee!). Thus, we will switch to iterative recovery algorithms.

3 Iterative Recovery Algorithms

There are a number of iterative recovery algorithms. Here, we include a couple of them as well as their results.

- Expander Matching Pursuit (EMP) [IR08] does not use the RIP_1 but relies directly on Π coming from an expander; it gets $C = 1+\varepsilon$ for ℓ_1/ℓ_1 recovery using an $(O(k), O(\varepsilon))$-expander. This is the best known iterative ℓ_1/ℓ_1-sparse recovery algoithm in terms of theoretically proven bounds.
- Sparse Matching Pursuit (SMP) [1] also does not use the RIP_1 abstraction but relies on expanders; it gets $C = O(1)$ using an $(O(k), O(1))$-expander. SMP performs better than EMP in practice, despite the theoretical results proven being not as good.
- Sequential Sparse Matching Pursuit (SSMP) [2] was originally analyzed with a reliance on expanders; it gets $C = O(1)$ using an $(O(k), O(1))$-expander. Price [6] later showed how to analyze the same algorithm relying only on Π being $(O(1), O(k))$-RIP$_1$, thus not relying on expanders explicitly.

In the remainder we describe SSMP and the analysis of [6].

4 Sequential Sparse Matching Pursuit (SSMP)

Here is the pseudocode for SSMP recovery given $b = \Pi x + e$, where Π an RIP$_1$ matrix. The vector $e \in \mathbb{R}^m$ is the error vector.

1. Initialize $x^{[0]} \leftarrow 0$
2. for $j = 1$ to T:
   (a) $x^{[j,0]} \leftarrow x^{[j-1]}$
   (b) for $a = 1$ to $(c-1)k$:
      i. $(i, z) = argmin_{(i,z)}(\|b - \Pi(x^j + z \cdot e_i)\|_1)$
      ii. $x^{[j,a]} \leftarrow x^{[j,a-1]} + z \cdot e_i$
   (c) $x^{[j]} \leftarrow H(x^{[j,(c-1)k]})$
3. return $x^{[T]}$

Note finding the (i, z) minimizing the above expression can actually be done quickly using a balanced binary search tree. Suppose Π is d-sparse in each column and r-sparse in each row. We create a max priority queue Q in which there are n keys $1, \ldots, n$, where key i has value equal to how much $\|b - \Pi(x^j + z \cdot e_i)\|_1$ is decreased when z is chosen optimally. Then to find (i, z) which is the argmin above, we remove the key in the tree with the smallest value then make the corresponding update by adding $z \cdot e_i$ to our current iterate. Note that doing this affects the search tree values for every other $j \in [n]$ such that the jth column of Π and the ith column of Π both have a non-zero entry

in the same row for at least one row. Thus the total number of j whose values are affected is at most dr. Finding the new optimal z and calculating the new value for each takes $O(d)$ time for each one. When using an expander to construct Π, r is $O(nd/m)$. Thus we may have to adjust keys for $O(nd^2/m)$ other elements in the priority queue, taking $O((nd^2/m)(d + \log n))$ time total per iteration of the inner loop. There are $O(k)$ iterations through the inner loop, and thus each outer loop takes time $O((nd^2k/m)(d + \log n))$. For an optimal (k, ε)-expander we have $m = \Theta(kd)$ and $d = O(\log n)$, so the total time per outer loop is $O(nd \log n) = O(n \log(n/k) \log n)$.

Now for the error analysis. Without loss of generality, x is k-sparse. If not and $x = x^k + (x - x^k)$ (where x^k is the best k-sparse approximation to x), then we can rewrite $\Pi x + e = \Pi x^k + (e + \Pi(x - x^k))$. Now define $\tilde{e} = (e + \Pi(x - x^k))$. Then

$$\|\tilde{e}\|_1 \leq \|e\|_1 + \|\Pi(x - x^k)\|_1 \leq \|e\|_1 + \|x - x^k\|_1 \tag{2}$$

Although $x - x^k$ is not k-sparse, note that $\|\Pi z\|_1 \leq \|z\|_1$ for *any* vector z, since we can just break up z into a sum of sparse vectors (by partitioning coordinates) then applying the triangle inequality.

Henceforth we assume x is k-sparse, and our goal is to show:

$$\|x - x^{[T+1]}\|_1 \leq C \cdot [2^{-T} \cdot \|x - x^{[0]}\|_1 + \|e\|_1] \tag{3}$$

Note when x isn't actually k-sparse, the $2^{-T}\|x - x^{[0]}\|$ term is actually $2^{-T}\|H_k(x)|$, where H_k is the hard thresholding operator from last lecture. With the exception of the $2^{-T} \cdot \|H_k(x)\|_1$ term (which should decrease exponentially with increasing iterations T), this is our ℓ_1/ℓ_1-error. This means at any particular iteration $j + 1$, we have the following subgoal:

$$\|x - x^{[j+1]}\|_1 \leq 1/2 \cdot \|x - x^{[j]}\|_1 + c' \cdot \|e\|_1 \tag{4}$$

If we can show this, then (3) follows by induction on j (and making C big enough as a function of c' to make the inductive step go through).

5 SSMP Proof

First, let's allow some notation $y^{[j]} = x - x^{[j]}$ and $y^{[j,a]} = x - x^{[j,a]}$. We will show this proof in four steps.

5.1 Step 1.

We show each iteration of the inner loop decreases error by $(1 - \frac{1}{2_{k+a}})^{1/2}$. This is to say, as long as the error is at least $c''\|e\|_1$ for some c'',

$$\|\Pi x^{[j+1,a]} - b\|_1 \leq (1 - \frac{1}{2_{k+a}})^{1/2} \cdot \|\Pi x^{[j+1,a]} - b\|_1 \tag{5}$$

Note the interesting case is indeed when the error is still at least $c''\|e\|_1$ (since iterating beyond that point just always keeps us at error $O(\|e\|_1)$. Let's assume the above is true for the rest of the proof.

5.2 Step 2.

Given Step 1 of the proof, we show that since we run the inner loop $(c-1)k$ times, then at the end of the loop when $a = (c-1)k$, we have:

$$\|\Pi x^{[j+1,(c-1)k]} - b\|_1 \leq [\prod_{a=0}^{t-1}(1 - \frac{1}{2_{k+a}})^{1/2} \cdot \|\Pi x^{[j]} - b\|_1$$

Then $(1 - \frac{1}{2_{k+a}}) = \frac{2k+a-1}{2k+a}$, which implies that when $c = 127$ we get something like $t = (c-1)k = 126k$, which makes this coefficient at most $\frac{1}{8}$. As a result, we get:

$$\|\Pi x^{[j+1,(c-1)k]} - b\|_1 \leq \frac{1}{8}\|\Pi x^{[j]} - b\|_1 \tag{6}$$

5.3 Step 3.

Recall that $b = \Pi x + e$, we can substitute it back in, and then use triangle inequality to show

$$\begin{aligned}
\|\Pi x^{[j+1,t]} - b\|_1 &= \|\Pi(x^{[j+1,t]} - x) - e\|_1 &(7)\\
&\geq \|\Pi(x^{[j+1,t]} - x)\|_1 - \|e\|_1 &(8)\\
&\geq (1-\varepsilon) \cdot \|x^{[j+1,t]} - x\|_1 - \|e\|_1 &(9)
\end{aligned}$$

Re-arranging the equation and then applying some arithmetic and using $\varepsilon < 1/2$ gives us:

$$\begin{aligned}
\|x^{[j+1,t]} - x\|_1 &\leq \frac{1}{1-\varepsilon} \cdot (\|\Pi x^{[j+1,t]} - b\|_1 + \|e\|_1)\\
&\leq 2\|\Pi x^{[j+1,t]} - b\|_1 + 2\|e\|_1\\
&\leq \frac{1}{4}\|\Pi x^{[j]} - b\|_1 + 2\|e\|_1\\
&\leq \frac{1}{4}\|\Pi(x^{[j]} - x)\|_1 + \frac{9}{4}\|e\|_1\\
&\leq \frac{1}{4}\|x^{[j]} - x\|_1 + \frac{9}{4}\|e\|_1
\end{aligned}$$

5.4 Step 4.

Here, we show that the previous result implies $\|x^{[j+1]} - x\|_1 \leq 1/2 \cdot \|x^{[j]} - x]|_1 + \frac{9}{2}\|e\|_1$. Notice the first step is by adding an identity, the second is by triangle inequality, and the last is by using the

results form step 3.

$$\begin{aligned}\|x^{[j+1]} - x\| &= \|x^{[j+1]} - (x^{[j+1,t]} + x^{[j+1,t]}) - x\|_1 \\ &\leq \|x^{[j+1]} - x^{[j+1,t]}\| + \|x^{[j+1,t]} - x\|_1 \\ &\leq 2\|x - x^{[j+1,t]}\|_1 \qquad (10) \\ &\leq 1/2 \cdot \|x^{[j]} - x]|_1 + \frac{9}{2}\|e\|_1\end{aligned}$$

where (10) follows since $x^{[j+1]}$ is the best k-sparse approximation to $x^{[j+1,t]}$, whereas x is some other k-sparse vector.

6 Lemmas

Now it just suffices to establish Step 1. It relies on a few lemmas, which are proven in [6].

Lemma 6. *Suppose you have a bunch of vectors $r_1, ..., r_s \in R^m$ and $z = \mu + \sum_{i=1}^{s} r_i$ where $\|\mu\|_1 \leq c \cdot \|z\|_1$ then if*

$$(1 - \delta) \sum \|r_i\|_1 \leq \| \sum r_i\|_1 \leq \sum \|r_i\|_1$$

then there exists i such that

$$\|z - r_i\|_1 \leq (1 - \frac{1}{s}(1 - 2\delta - 5c))\|z\|_1$$

Intuitively the condition on the r_i implies that there is not much cancellation when they are summed up, so not much ℓ_1 mass is lost by summing. In the case when there is no cancellation at all, then obviously (if μ were zero, say) any non-zero r_i could be subtracted from the sum to decrease $\|z\|_1$. The above lemma captures this intuition even when there can be a small amount of cancellation, and a small norm μ is added as well (think of c as being a very small constant).

Now we have the next lemma, which can be proven by the previous one.

Lemma 7. *If Π is (s, 1/10)-RIP$_1$ and $s > 1$, then if y is s-sparse, $\|w\|_1 \leq 1/30\|y\|_1$ then there exists a 1-sparse z such that $\|\Pi(y - z) + w\|_1 \leq (1 - \frac{1}{s})^{1/2}\|Piy + w\|_1$.*

This is to say that at every step choose the best 1-sparse to add to decrease the error.

Now, step 1 follows by noting that $x^{j,a}$ is $(k + a)$-sparse, so $x^{j,a} - x$ is $2k + a \leq (c + 1)k$ sparse. Thus there is one-sparse update (by the above lemma) which decreases the error (and note SSMP finds the best one-sparse update in each iteration of the inner loop, so it does at least as well).

References

[IR08] Piotr Indyk and Milan Ružić. Near-optimal sparse recovery in the l1 norm. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*, pages 199–207. IEEE, 2008.

[1] Radu Berinde, Anna Gilbert, Piotr Indyk, Howard Karloff, and Martin Strauss. Combining geometry and combinatorics: a unified approach to sparse signal recovery. *Allerton*, 2008.

[2] Radu Berinde, Piotr Indyk. Sequential sparse matching pursuit. *Allerton*, 2009.

[3] Thomas Blumensath, Mike E. Davies. Iterative hard thresholding for compressed sensing. *Appl. Comput. Harmon. Anal.*, 27:265–274, 2009.

[4] Venkatesan Guruswami, Christopher Umans, and Salil P. Vadhan. Unbalanced expanders and randomness extractors from Parvaresh-Vardy codes. In *IEEE Conference on Computational Complexity*, pages 96–108. IEEE Computer Society, 2007.

[5] Piotr Indyk, Ronitt Rubinfeld. Sublinear algorithms. `http://stellar.mit.edu/S/course/6/sp13/6.893/courseMaterial/topics/topic2/lectureNotes/riplp/riplp.pdf`

[6] Eric Price. Improved analysis of Sequential Sparse Matching Pursuit. *Unpublished manuscript*, 2010.

CS 229r: Algorithms for Big Data Fall 2013

Lecture 22 — November 5, 2013

Prof. Jelani Nelson *Scribe: Abdul Wasay*

1 Introduction to the Matrix Completion Problem

(Notes heavily borrow from Fall 2013 notes by Kristan Temme and Yun William Yu)

This is sometimes called the Netflix problem. A motivation for the matrix completion problem comes from user ratings of some products which are put into a matrix M. The entries M_{ij} of the matrix correspond to the j'th user's rating of product i. We assume that there exists an ideal matrix that encodes the ratings of all the products by all the users. However, it is not possible to ask every user his opinion about every product. We are only given some ratings of some users and we want to recover the actual ideal matrix M from this limited data. So matrix completion is the following problem:

Problem: Suppose you are given some matrix $M \in \mathbb{R}^{n_1 \times n_2}$. Moreover, you also are given some entries $(M_{ij})_{ij \in \Omega}$ with $|\Omega| \ll n_1 n_2$.

Goal: We want to recover the missing elements in M.

This problem is impossible if we don't make any additional assumptions on the matrix M since the missing M_{ij} could in principle be arbitrary. We will discuss a recovery scheme that relies on the following three assumptions.

1. M is (approximately) low rank.
2. Both the columns space and the row space are "incoherent". We say a space is incoherent, when the projection of any vector onto this space has a small ℓ_2 norm.
3. If $M = U\Sigma V^T$ then all the entries of UV^T are bounded.
4. The subset Ω is chosen uniformly at random.

Note 1. *There is work on adversarial recovery where the values are not randomly chosen but carefully picked to trick us by an adversary.*

Under these assumptions we show that there exists an algorithm that needs a number of entries in M bounded by $|\Omega| \leq (n_1 + n_2)\, r\, \mathrm{poly}\left(\log(n_1 n_2)\right) \cdot \mu$. Here μ captures to what extent properties 2 and 3 above hold. One would naturally consider the following recovery method for the matrix M:

$$\begin{aligned} &\text{minimize} \quad \mathrm{rank}(X) \\ &\text{subject to:} \quad X_{ij} = M_{ij} \;\; \forall i, j \in \Omega. \end{aligned}$$

Unfortunately this optimization problem is NP-hard. We will therefore consider the following alternative optimization problem in trace norm, or *nuclear norm*.

$$\begin{aligned} &\text{minimize} \quad \|X\|_* \\ &\text{subject to:} \quad X_{ij} = M_{ij} \ \ \forall i,j \in \Omega, \end{aligned}$$

where the nuclear norm of X defined as the sum of the singular values of X, i.e. $\|X\|_* = \sum_i \sigma_i(X)$. This problem is an SDP (semi- definite program), and can be solved in time polynomial in $n_1 n_2$.

2 Work on Matrix Completion

Let's now go through the history of prior work on this problem. Recall the setup and model:

- Matrix completion setup:
  - Want to recover $M \in \mathbb{R}^{n_1 \times n_2}$, under the assumption that $\text{rank}(M) = r$, where r is small.
  - Only some small subset of the entries $(M_{ij})_{ij \in \Omega}$ are revealed, where $\Omega \subset [n_1] \times [n_2], |\Omega| = m \ll n_1, n_2$
- Model:
  - m times we sample i, j uniformly at random + insert into Ω (so Ω is a multiset).
  - Note that the same results hold if we choose m entries without replacement, but it's easier to analyze this way. In fact, if you can show that if recovery works with replacement, then that implies that recovery works without replacement, which makes sense because you'd only be seeing more information about M.
- We recover M by Nuclear Norm Minimization (NNM):
  - Solve the program $\min \|X\|_*$ s.t. $\forall i,j \in \Omega, X_{ij} = M_{ij}$
- [Recht, Fazel, Parrilo '10] [RFP10] was first to give some rigorous guarantees for NNM.
- [Candés, Recht, '09] [CR09] was the first paper to show provable guarantees for NNM applied to matrix completion.
- There were some quantitative improvements (in the parameters) in two papers: [Candés, Tao '09] [CT10] and [Keshavan, Montanari, Oh '09] [KMO10]
- **Today** we're going to cover an analysis given in [Recht, 2011] [Rec11], which has a couple of advantages.
  - First, it has the laxest of all the conditions.
  - Second, it's also the simplest of all the analyses in the papers.
  - Thus, it's really better in every way there is.

The approach of [Rec11] was inspired by work in quantum tomography [GLF+10]. A more general theorem than the one proven in class today was later proven by Gross [Gross].

It is worth noting that there have been other important works on matrix completion which we will not get to in the course. In particular, one particular paradigm is *Alternating Minimization (AM)*.

The basic idea behind AM is as follows. It is an iterative algorithm. We try to find an approximate rank-k factorization $M \approx X \cdot Y$, where X has k columns and Y has k rows. We start off with initial X_0, Y_0. Then we do as follows:

1. initialize X_0, Y_0 (somehow)
2. **for** $\ell = 1, \ldots, T$:
   (a) $X_\ell \leftarrow argmin_X \|R_\Omega(M - XY_{\ell-1})\|_F^2$
   (b) $Y_\ell \leftarrow argmin_Y \|R_\Omega(M - X_\ell Y)\|_F^2$
3. **return** X_T, Y_T

Rigorous analyses of modifications of the above AM template have been carried out in [1, 2]. The work [3] has also shown some performance guarantees when the revealed entries are *adversarial* except for random (though in this case, many more entries have to be revealed).

3 Theorem Statement

We're almost ready to formally state the main theorem, but need a couple of definitions first.

Definition 2. *Let $M = U\Sigma V^*$ be the singular value decomposition. (Note that $U \in \mathbb{R}^{n_1 \times r}$ and $V \in \mathbb{R}^{n_2 \times r}$.)*

Definition 3. *Define the incoherence of the subspace U as $\mu(U) = \frac{n_1}{r} \cdot \max_i \|P_U e_i\|^2$, where P_U is projection onto U. Similarly, the incoherence of V is $\mu(V) = \frac{n_2}{r} \cdot \max_i \|P_V e_i\|^2$, where P_V is projection onto V.*

Definition 4. $\mu_0 \stackrel{\text{def}}{=} \max\{\mu(U), \mu(V)\}$.

Definition 5. $\mu_1 \stackrel{\text{def}}{=} \|UV^*\|_\infty \sqrt{n_1 n_2 / r}$, *where $\|UV\|_\infty$ is the largest magnitude of an entry of UV.*

Theorem 6. *If $m \gtrsim \max\{\mu_1^2, \mu_0\} \cdot n_2 r \log^2(n_2)$ then with high probability M is the unique solution to the semi-definite program $\min \|X\|_*$ s.t. $\forall i, j \in \Omega, X_{ij} = M_{ij}$.*

Note that $1 \leq \mu_0 \leq \frac{n_2}{r}$. The way μ_0 can be $\frac{n_2}{r}$ is if a standard basis vector appears in a column of V, and the way μ_0 can get all the way down to 1 is like the best case scenario where all the entries of V are like $\frac{1}{\sqrt{n_2}}$ and all the entries of U are like $\frac{1}{\sqrt{n_1}}$, so for example if you took a Fourier matrix and cut off some of its columns. Thus, the condition on m is a good bound if the matrix has low incoherence.

One might wonder about the necessity of all the funny terms in the condition on m. Unfortunately, [Candes, Tao, '09] [CT10] showed $m \gtrsim \mu_0 n_2 r \log(n_2)$ is needed (that is, there is a family of examples

M that need this). If you want to have any decent chance of recovering M over the random choice of Ω using this SDP, then you need to sample at least that many entries. The condition isn't completely tight because of the square in the log factor and the dependence on μ_1^2. However, Cauchy-Schwarz implies $\mu_1^2 \leq \mu_0^2 r$.

Just like in compressed sensing, there are also some iterative algorithms to recover M, but we're not going to analyze them in class. For example, the **SparSA** algorithm given in [Wright, Nowak, Figueiredo '09] [WNF09] (thanks for Ben Recht for pointing this out to me). That algorithm roughly looks as follows when one wants to minimize $\|AX - M\|_F^2 + \mu\|X\|_*$:

Pick X_0, and a stepsize t and iterate (a)-(d) some number of times:

(a) $Z = X_k - t \cdot A^T(AX_k - M)$
(b) $[U, \mathbf{diag}(s), V] = \mathbf{svd}(Z)$
(c) $r = \max(s - \mu t, 0)$
(d) $X_{k+1} = U\mathbf{diag}(r)V^T$

As an aside, trace-norm minimization is actually tolerant to noise, but I'm not going to cover that.

4 Analysis

The way that the analysis is going to go is we're going to condition on lots of good events all happening, and if those good events happen, then the minimization works. The way I'm going to structure the proof is I'll first state what all those events are, then I'll show why those events make the minimization work, and finally I'll bound the probability of those events not happening.

4.1 Background and more notation

Before I do that, I want to say some things about the trace norm.

Definition 7. $\langle A, B\rangle \stackrel{\text{def}}{=} \text{Tr}(A^*B) = \sum_{i,j} A_{ij}B_{ij}$

Claim 8. *The dual of the trace norm is the operator norm:*

$$\|A\|_* = \sup_{\substack{B \text{ s.t.} \\ \|B\| \leq 1}} \langle A, B\rangle$$

This makes sense because the dual of ℓ_1 for vectors is ℓ_∞ and this sort of looks like that because the trace norm and operator norm are respectively like the ℓ_1 and ℓ_∞ norm of the singular value vector. More rigorously, we can prove it by proving inequality in both directions. One direction is not so hard, but the other requires the following lemma.

Lemma 9.

$$\underbrace{\|A\|_*}_{(1)} = \underbrace{\min_{\substack{X,Y \text{ s.t.} \\ A=XY^*}} \|X\|_F \cdot \|Y\|_F}_{(2)} = \underbrace{\min_{\substack{X,Y \text{ s.t.} \\ A=XY^*}} \frac{1}{2}\left(\|X\|_F^2 + \|Y\|_F^2\right)}_{(3)}$$

Proof of lemma.

(2) $\leq$ (3):
AM-GM inequality: $xy \leq \frac{1}{2}(x^2 + y^2)$.

(3) $\leq$ (1):

We basically just need to exhibit an X and Y which are give something that is at most the $\|A\|_*$. Set $X = Y^* = A^{1/2}$. In general, given $f : \mathbb{R}^+ \mapsto \mathbb{R}^+$, then $f(A) = Uf(\Sigma)V^*$. i.e. write the SVD of A and apply f to each diagonal entry of Σ. You can easily check that $A^{1/2}A^{1/2} = A$ and that the square of the Frobenius norm of $A^{1/2}$ is exactly the trace norm.

(1) $\leq$ (2):

Let X, Y be some matrices such that $A = XY^*$. Then

$$\begin{aligned}
\|A\|_* &= \|XY^*\|_* \\
&\leq \sup_{\substack{\{a_i\} \text{ orthonormal basis} \\ \{b_i\} \text{ orthonormal basis}}} \sum_i \langle XY^* a_i, b_i \rangle && \text{This can be seen to be true by letting } a_i{=}v_i \text{ and } b_i{=}u_i \text{ (from the SVD), when we get equality.} \\
&= \sup_{\cdots} \sum_i \langle Y^* a_i, X^* b_i \rangle \\
&\leq \sup_{\cdots} \sum_i \|Y^* a_i\| \cdot \|X^* b_i\| \\
&\leq \sup_{\cdots} (\sum_i \|Y^* a_i\|^2)^{1/2} (\sum_i \|X^* b_i\|^2)^{1/2} && \text{(by Cauchy-Schwarz)} \quad (1) \\
&= \|X\|_F \cdot \|Y\|_F && \text{because } \{a_i\},\{b_i\} \text{ are orthonormal bases and the Frobenius norm is rotationally invariant}
\end{aligned}$$

□

Proof of claim.

Part 1:

$$\|A\|_* \leq \sup_{\|B\|=1} \langle A, B \rangle .$$

We show this by writing $A = U\Sigma V^*$. Then take $B = \sum_i u_i v_i^*$. That will give you something on the right that is at least the trace norm.

Part 2:

$$\|A\|_* \geq \langle A, B \rangle \quad \forall B \text{ s.t. } \|B\| = 1.$$

We show this using the lemma.

- Write $A = XY^*$ s.t. $\|A\|_* = \|X\|_F \cdot \|Y\|_F$ (lemma guarantees that there exists such an X and Y).
- Write $B = \sum_i \sigma_i a_i b_i, \forall i, \sigma_i \leq 1$.

Then using a similar argument to (1),

$$\begin{aligned}\langle A, B\rangle &= \left\langle XY^*, \sum_i \sigma_i a_i b_i \right\rangle \\ &= \sum_i \sigma_i \langle Y^* a_i, X^* b_i\rangle \\ &\le \sum_i |\langle Y^* a_i, X^* b_i\rangle| \\ &\le \|X\|_F \|Y\|_F = \|A\|_*\end{aligned}$$

which concludes the proof of the claim. □

Recall that the set of matrices that are $n_1 \times n_2$ is itself a vector space. I'm going to decompose that vector space into T and the orthogonal complement of T by defining the following projection operators.

- $P_{T^\perp}(Z) \stackrel{\text{def}}{=} (I - P_U)Z(I - P_V)$
- $P_T(Z) \stackrel{\text{def}}{=} Z - P_{T^\perp}(Z)$

So basically, the matrices that are in the vector space $T^\perp$ are the matrices that can be written as the sum of rank 1 matrices $a_i b_i^*$ where the a_i's are orthogonal to all the u's and the b_i's are orthogonal to all the v's. Also define $R_\Omega(Z)$ as only keeping entries in Ω, multiplied by multiplicity in Ω. If you think of the operator $R_\Omega : \mathbb{R}^{n_1 n_2} \mapsto \mathbb{R}^{n_1 n_2}$ as a matrix, it is a diagonal matrix with the multiplicity of entries in Ω on the diagonal.

4.2 Good events

We will condition on all these events happening in the analysis. It will turn out that with high probability—probability $1 - \frac{1}{poly(n_2)}$, and you can make the $\frac{1}{poly(n_2)}$ factor decay as much as you want by increasing the constant in from of m—all these events will occur:

1. $\left\|\frac{n_1 n_2}{m} P_T R_\Omega P_T - P_T\right\| \lesssim \sqrt{\frac{\mu_0 r(n_1+n_2)\log(n_2)}{m}} \ll \frac{1}{2}$

   This is simple to understand from the perspective of leverage score sampling for approximate matrix multiplication (AMM) with spectral norm error (as in pset 4, problem 2). Specifically, recall that AMM we have matrices A, B with the same number n of rows and want for some Π with m rows that $\|(\Pi A)^T(\Pi A) - A^T B\| \le \varepsilon \|A\| \cdot \|B\|$. Now, note here that $P_T = P_T P_T$, since P_T is a projection matrix. Thus the above is just an AMM condition for $A = B = P_T$, and $\Pi = R_\Omega^{1/2}$. Now, typically for row sampling we had Π be a diagonal matrix with $\Pi_{i,i} = \eta_i/\sqrt{p_i}$, where η_i is an indicator random variable for the event that we sampled row i, and $p_i = \mathbb{E}\,\eta_i$. In class we discussed that we should set p_i to be roughly proportional to the *leverage score* of row i. The total number of samples is thus on the order of the sum of leverage scores. More specifically, according to pset 4 problem 2, the total number of rows samples will be on the order of $O(q \log(q/\delta)/\varepsilon^2)$ to suceed with probability $1 - \delta$, where q is the sum of the leverage scores (or equivalently, the maximum rank of A, B). In our

case, the rank of P_T is the sum of leverage scores of P_T, which is $\sum_{a,b} \|P_T e_a e_b^*\|_F^2$, which is $\sum_{a,b}(\|P_U e_a\|_2^2 + \|P_V e_b\|_2^2 - \|P_U e_a\|_2^2 \cdot \|P_V e_b\|_2^2)$. Here P_U is orthogonal projection onto U, where $M = U\Sigma V^*$. One can verify that this sum is $n_1 r + n_2 r - r^2 = r(n_1 + n_2 - r) = O((n_1 + n_2)r)$. Thus $q \log(q/\delta)$ is $O(r(n_1 + n_2)\log(n_2/\delta)) = O(rn_2 \log(n_2/\delta))$ (note what m is above!). Now, unfortunately R_Ω samples *uniformly* and not according to leverage scores! This is where μ_0 comes in. We need to make sure we oversample enough so that each row's expected number of occurrences in our sampling is *at least* its leverage score (show via a modification of your pset 4 pset 2 solution that this suffices). To make this oversampling good enough, we need to oversample by a factor related to the maximum leverage score, hence the μ_0 in m.

2. $\|R_\Omega\| \lesssim \log(n_2)$

   This one is actually really easy (also the shortest): it's just balls and bins. We've already said R_Ω is a diagonal matrix, so the operator norm is just the largest diagonal entry. Imagine we have m balls, and we're throwing them independently at random into $n_1 n_2$ bins, namely the diagonal entries, and this is just how loaded is the maximum bin. In particular, $m < n_1 n_2$, or else we wouldn't be doing matrix completion since we'd have the whole matrix. In general, when you throw t balls into t bins, the maximum load by the Chernoff bound is at most $\log t$. In fact, it's at most $\log t / \log\log t$, but who cares, since that would save us an extra $\log\log$ somewhere. Actually, I'm not even sure it would save us that since there are other log's that come into play.

3. $\exists Y$ in $range(R_\Omega)$ s.t.

   (5a) $\|P_T(Y) - UV^*\|_F \leq \sqrt{\frac{r}{2n_2}}$

   (5b) $\|P_{T^\perp}(Y)\| < \frac{1}{2}$

   We will not justify this one in class; see the paper for the argument for the existence of such a Y.

4.3 Recovery conditioned on good events

Now that we've stated all these things, let's show that they imply trace norm minimization actually works. We want to make sure

$$argmin_{\substack{X \text{ s.t.} \\ R_\Omega(X) = R_\Omega(M)}} \|X\|_*$$

is unique and equal to M.

Let $Z \in Ker(R_\Omega), (Z \neq 0)$; we want to show $\|M + Z\|_* > \|M\|_*$.

First we want to argue that $\|P_T(Z)\|_F$ cannot be big.

Lemma 10. $\|P_T(Z)\|_F < \sqrt{\frac{n_2}{2r}} \cdot \|P_{T^\perp}(Z)\|_F$

Proof.

$$0 = \|R_\Omega(Z)\|_F \geq \|R_\Omega(P_T(Z))\|_F - \|R_\Omega(P_{T^\perp}(Z))\|_F$$

Also

$$\begin{aligned}
\|R_\Omega(P_T(Z))\|_F^2 &= \langle R_\Omega P_T Z, R_\Omega P_T Z\rangle \\
&\geq \langle P_T Z, R_\Omega P_T Z\rangle \\
&= \langle Z, P_T R_\Omega P_T Z\rangle \\
&= \langle P_T Z, P_T R_\Omega P_T P_T Z\rangle \\
&= \left\langle P_T Z, \frac{m}{n_1 n_2} P_T P_T Z \right\rangle + \left\langle P_T Z, (P_T R_\Omega P_T - \frac{m}{n_1 n_2}) P_T Z \right\rangle \\
&\geq \frac{m}{n_1 n_2} \|P_T Z\|_F^2 - \left\| P_T R_\Omega P_T - \frac{m}{n_1 n_2} \right\| \cdot \|P_T Z\|_F^2 \\
&\geq \frac{m}{n_1 n_2} \cdot \|P_T Z\|_F^2
\end{aligned}$$

Also have

$$\begin{aligned}
\|R_\Omega(P_{T^\perp}(Z))\|_F^2 &\leq \|R_\Omega\|^2 \cdot \|P_{T^\perp}(Z)\|_F^2 \\
&\lesssim \log^2(n_2) \cdot \|P_{T^\perp}(Z)\|_F^2
\end{aligned}$$

Summarize: combining all the inequalities together, and then making use of our choice of m,

$$\begin{aligned}
\|P_T(Z)\|_F &< \sqrt{\frac{n_1 n_2 \log^2(n_2)}{m}} \cdot \|P_{T^\perp}(Z)\|_F \\
&< \sqrt{\frac{n_2}{2r}} \cdot \|P_{T^\perp}(Z)\|_F
\end{aligned}$$

□

Pick $U_\perp, V_\perp$ s.t. $\langle U_\perp V_\perp^*, P_{T^\perp}(Z)\rangle = \|P_T(Z)\|_*$ and s.t. $[U, U_\perp], [V, V_\perp]$ orthogonal matrices. We know from claim 8 that the trace norm is exactly the sup over all B matrices of the inner product. But the B matrix that achieves the sup has all singular values equal to 1, so $B = U_\perp V_\perp^*$, because $P_{T^\perp}(Z)$ is in the orthogonal space so B should also be in the orthogonal space.

Now we have a long chain of inequalities to show that the trace of any $M + Z$ is greater than the trace of M:

$$
\begin{aligned}
\|M+Z\|_* &\geq \langle UV^* + U_\perp V_\perp^*, M+Z\rangle && \text{by claim 8}\\
&= \|M\|_* + \langle UV^* + U_\perp V_\perp^*, Z\rangle && \text{since } M \perp U_\perp V_\perp^*\\
&= \|M\|_* + \langle UV^* + U_\perp V_\perp^* - Y, Z\rangle && \text{since } Z\in ker(R_\Omega) \text{ and } Y\in range(R_\Omega)\\
&= \|M\|_* + \langle UV^* - P_T(Y), P_T(Z)\rangle + \langle U_\perp V_\perp^* - P_{T^\perp}(Y), P_{T^\perp}(Z)\rangle && \text{decomposition into } T \text{ \& } T^\perp\\
&\geq \|M\|_* - \|UV^* - P_T(Y)\|_F \cdot \|P_T(Z)\|_F && \langle x,y\rangle \leq \|x\|_2\|y\|_2\\
&\qquad + \|P_{T^\perp}(Z)\|_* && \text{by our choice of } UV^*\\
&\qquad - \|P_{T^\perp}(Y)\| \cdot \|P_{T^\perp}(Z)\|_* && \text{norm inequality}\\
&\geq \|M\|_* - \sqrt{\frac{r}{2n_2}} \cdot \|P_T(Z)\|_F + \frac{1}{2}\cdot\|P_{T^\perp}(Z)\|_*\\
&> \|M\|_* - \frac{1}{2}\cdot\left\|P_T^\perp(Z)\right\|_F + \frac{1}{2}\cdot\|P_{T^\perp}(Z)\|_* && \text{by Lemma 10}\\
&\geq \|M\|_* && \text{since } \|\cdot\|_* \geq \|\cdot\|_F
\end{aligned}
$$

Hence, when all of the good conditions hold, minimizing the trace norm recovers M.

5 Concluding remarks

Why would you think of trace minimization as solving matrix completion? Analogously, why would you use ℓ_1 minimization for compressed sensing? In some way, these two questions are very similar in that rank is like the support size of the singular value vector, and trace norm is the ℓ_1 norm of the singular value vector, so the two are very analogous. ℓ_1 minimization seems like a natural choice, since it is the closest convex function to support size from all the ℓ_p norms (and being convex allows us to solve the program in polynomial time).

References

[CR09] Emmanuel J Candès and Benjamin Recht, *Exact matrix completion via convex optimization*, Foundations of Computational mathematics **9** (2009), no. 6, 717–772.

[CT10] Emmanuel J Candès and Terence Tao, *The power of convex relaxation: Near-optimal matrix completion*, Information Theory, IEEE Transactions on **56** (2010), no. 5, 2053–2080.

[Gross] David Gross, *Recovering low-rank matrices from few coefficients in any basis*, Information Theory, IEEE Transactions on (2011), no. 57, :1548-1566.

[GLF+10] David Gross, Yi-Kai Liu, Steven T. Flammia, Stephen Becker, and Jens Eisert. *Quantum state tomography via compressed sensing*, Physical Review Letters (2010), 105(15):150401.

[1] Moritz Hardt. *Understanding Alternating Minimization for Matrix Completion.* FOCS, pages 651–660, 2014.

[2] Moritz Hardt, Mary Wootters. *Fast matrix completion without the condition number*. COLT, pages 638–678, 2014.

[KMO10] Raghunandan H Keshavan, Andrea Montanari, and Sewoong Oh, *Matrix completion from noisy entries*, The Journal of Machine Learning Research **99** (2010), 2057–2078.

[Rec11] Benjamin Recht, *A simpler approach to matrix completion*, The Journal of Machine Learning Research **12** (2011), 3413–3430.

[RFP10] Benjamin Recht, Maryam Fazel, and Pablo A Parrilo, *Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization*, SIAM review **52** (2010), no. 3, 471–501.

[3] Tselil Schramm, Benjamin Weitz. *Low-Rank Matrix Completion with Adversarial Missing Entries*. CoRR abs/1506.03137, 2015.

[WNF09] Stephen J Wright, Robert D Nowak, and Mário AT Figueiredo, *Sparse reconstruction by separable approximation*, Signal Processing, IEEE Transactions on **57** (2009), no. 7, 2479–2493.

CS 229r: Algorithms for Big Data Fall 2015

Lecture 22 — November 19, 2015

Prof. Jelani Nelson *Scribe: Johnny Ho*

1 Overview

Today we're starting a completely new topic, which is the external memory model, also known as the disk access model. One of the earliest works on this was by Aggrawal and Vitter [1].

We've talked about streaming, which is where we cannot store all of the data. Here we imagine instead that our data cannot fit inside RAM, but we can store it in some other disk space. We assume that disk is infinite, but memory is finite.

In the RAM, we assume that memory of size M is divided into pages of size B, i.e. there are M/B pages. The unit here is arbitrary, each item takes up constant size. Similarly, there are infinite pages of size B on disk. When we do memory I/O, we imagine that the cost is free, but when we do disk I/O, we imagine that we can fetch a page, which has a unit cost.

This makes practical sense since memory is orders of magnitude faster than hard drive space, and is often far smaller. Hard drives are usually much faster when reading sequentially, which is why we use the idea of pages.

In terms of exact numbers, hard drives support somewhere around 140MB/sec of sequential reads/writes, but only 120I/O's/sec when doing reads/writes on random 4KB chunks. This is because the disk must physically spin to the read/write location. RAM supports somewhere around 5-10GB/sec of reads/writes, with much less penalty for random access. SSDs/Flash is somewhere in the middle, but with the issue that each index can only be written to a limited number of times.

In the future, we'll work with the idea of algorithms that work even without knowing the exact values of M and B, which are known as cache-oblivious algorithms.

2 Algorithms

2.1 Reading N Items (Streaming)

It takes exactly $\lceil N/B \rceil = O(N/B + 1)$ cost to read N items. Same for any streaming algorithm, since we ignore in-memory operations.

In general, we assume that we can choose the layout of the items on disk, in whatever manner is most convenient.

2.2 Matrix Multiplication

Given two $N \times N$ matrices X and Y, we want their product. We divide X and Y into blocks of size $\sqrt{M}/2$ and $\sqrt{M}/2$, so that each block fits in memory. We then store each block contiguously in row/column order form in disk. If the input is not already in this form, we can rearrange it into this form, which has some complexity depending on the original input format, but again, we choose the original layout, so we can ignore this.

Examine each such block in the output matrix $X \cdot Y$. The resulting block is the sum of products of several pairs of blocks, going across matrix X and down matrix Y. Then there are $(\frac{N}{\sqrt{M}})^2$ output blocks that need to be produced. Each output block requires multiplying $O(\frac{N}{\sqrt{M}})$ pairs of blocks, where each chunk can be read with $O(M/B)$ I/O's. The total number of I/O's required is $O(\frac{N}{\sqrt{M}}^3 \cdot M/B) = O(\frac{N^3}{B\sqrt{M}})$. This can be improved with better matrix multiplication.

2.3 Linked Lists

Dynamic vs. Static Data Structures. A data structure problem is one where you can do updates and queries to some data. A static data structure is one where the input data stays unchanged over time, i.e. no updates.

Linked lists are dynamic and must support insert(x, p), where p is a pointer to an entry you want to insert after. There is similarly remove(p), which requires removing the item pointed to by p. There is also traverse(p, k), where the k elements after p must be touched, i.e. print out their values.

Clearly the standard linked list solves these three operations optimally. The updates are $O(1)$, and traverse takes $O(k)$ time.

One idea is to block up nodes into chunks of size B. If this were static, this would satisfy travesrse easily since this is essentially an array. However, how do we update these over insertions and deletions? While updating/deleting, we can maintain the invariant that all groups must have size at least $B/2$, except possibly the very last group.

When deleting, we can usually just delete the element from the one group. If this pushes its size less than $B/2$, we can merge with an adjacent group. If this pushes the adjacent group's size above B, so we can then split evenly into size between $B/2$ and $3B/4$ elements. This will take two disk operations. All of the rearrangement logic here is free because it can be done in memory.

When inserting, we insert into the block. Again, if this pushes the group's size above B, we can split evenly into two. This again takes two disk operations.

Traverse works as expected, going through blocks until k elements are seen. This requires reading at most $2k/B$ items, since each cost corresponds to reading at least $B/2$ items.

2.4 Predecessor

Here we have N static items, which are ordered and comparable. Each item is a (key,value) pair, where the keys are being compared. Then we need to support query(k), where the predecessor item must be returned, i.e. the item with greatest key less than key k. For example, keys could be IP

addresses, and queries in a router table would return where to forward the IP address.

We can also have the same problem, except dynamic, i.e. there are inserts and deletes.

Static predecessor can be solved by storing a sorted array in memory, i.e. in N space, with $O(\log N)$ queries using binary search. On an actual computer, which follows the word RAM model, we can perform operations on actual bits, with word size W, usually $W = \Theta(\log N)$. Predecessor has been perfectly well understood. We can get $O(N)$ memory with $O(\log \log N)$ queries, with Van Emde Boas trees [4]. With $O(N^{O(1)})$ memory, say $O(N^{1.01})$, we can have $O(\log \log N / \log \log \log N)$ query time. This was shown by Beame and Fich [3]. These bounds have been proven optimal by Patrascu and Thorup [7].

2.5 (a,b)-tree / B-tree

In an external memory model, we can solve this problem with (a, b)-trees, which are used in databases and were invented by Bayer and McCreight [2]. This is kind of like a binary search tree, except each internal node has within $[a, b]$ children (except possibly the root, which has either no children or ≥ 2). Each node, regardless of whether it is a leaf or an internal node, stores an array of B values. Items are stored in leaves. Guides are stored in internal nodes. Each guide at index i in an internal node corresponds to a dividing value between i and $i + 1$, where all values in the subtree at i must have value less than the guide, and the subtree at $i + 1$ must have value $\geq$ than the guide.

Every time an insert happens, traverse the tree downwards, and place the new value into the leaf where it should go. If the leaf overflows, split the leaf into two leaves, and copy one of its keys to the parent. If that parent overflows, split that, etc., up until the root.

On removal, when a leaf is removed, the node may underflow (go under a). If so, merge it a neighboring (sibling) leaf, which removes one guide from its parent. This may cause another underflow, another merge, etc., up until the root. Then we have a guarantee of at least a children, so the height is limited by $O(\log_a(n/a))$, which is usually $O(\log n)$ since a is chosen to be a constant. There is an additional computation time of $\log b$ per operation, since we need to binary search the arrays, but b is usually chosen proportional to a.

A B-tree is just a (B/2, B)-tree, so that we can fit each node in one block. Thus each operation takes at most one I/O per height, which is $O(\log_B N)$.

2.6 (2, 4)-tree with Buffers / Buffered Repository Tree (BRT)

Tradeoffs. Compare the B-tree to a logging model, where we just log all operations and scan through them. This model has query cost $O(N/B + 1)$. An insertion, which just appends one log entry, takes at most 1 disk update. Amortizing this across all inserts, however, only every B inserts needs to actually update the disk if we cache in memory. Thus, updates take amortized $1/B$ cost.

Can we get a better insertion time while still keeping the better query time? We can, by using a (2, 4)-tree with buffers, shown by Buchsbaum et al. [6]. For each node, keep a buffer. When inserting, simply append to the root's buffer. When the buffer becomes full, then flush it to its children. When the buffer at the leaf is full, then split the leaf, and apply the normal promotion operation.

Each query has the same cost, since we are just reading the buffer along with its node. This takes $O(1)$ I/Os per height.

It turns out that the amortized complexity of insertion/deletion is $O(\log N/B)$. We can imagine allocating each insert $\theta(\log N/B)$ dollars, where 1 dollar can pay for 1 disk I/O. Since items are only flushed downwards, we can charge each item $1/B$ dollars. The number of times an item can be flushed is limited by the height, $O(\log N)$, so we have just enough money.

The given scheme has a poor worst-case complexity, since it could potentially recursively flush to both sides. Instead, when flushing, we can choose to only flush to the child that would receive the most items, which flushes at least $B/4$ items. This maintains the amortized complexity above, but decreases the worst case to $O(\log N)$.

An improvement by Brodal and Fagerberg [5] is to use a $(B^\varepsilon, 2B^\varepsilon)$ tree with buffers, called a buffered B-tree. The resulting height is $\log_{B^\varepsilon} N = \frac{1}{\varepsilon} \log_B N$, and we have amortized insertion time $\frac{1}{\varepsilon B^{1-\varepsilon}} \log_B N$, which is better than the standard $B - tree$.

2.7 Sorting

$O(n \log n)$ is optimal for sorting in an comparison model. This is, however, not optimal in the word-RAM model, where sorting can be done in $O(n \log \log n)$ time with Han's sorting algorithm, and $O(n\sqrt{\log \log n})$ with randomization.

In the external memory model, given N elements, divide them into M/B groups, so that we can store one page from each group in memory. Merge sort will then consume $M/B \cdot B = M$ memory at any time. Then we have the recursion $T(N) = 1$ for $N \leq B$, and otherwise $T(N) = O(N/B) + M/B \cdot T(N/(M/B))$. Solving with the Master theorem, we have $O(\frac{N}{B} \log_{\frac{M}{B}} \frac{N}{B})$ as the final complexity.

This can be shown to be optimal in the comparison model. First assume that in the input, each block of B elements is sorted, since we might as well iterate through them and sort them in memory. Then the number of valid permutations is $N!/(B!)^{N/B}$, which has a log of $N \log N - \frac{N}{B} B \log B = N \log \frac{N}{B}$. Given that we can only store M elements in memory, upon reading B elements, we can only learn how their interleave within the M elements. As a result, the amount of information we can gain is $\log(\binom{M+B}{B}) = B \log \frac{M}{B}$. Then, dividing the information needed by that gained per I/O, we have a lower bound of $\Omega(\frac{N}{B} \log_{\frac{M}{B}} \frac{N}{B})$, as shown earlier.

We will learn next week about cache-oblivious versions of these algorithms and data structures, which will have the same performance for B-trees, linked lists, and sorting.

References

[1] Alok Aggarwal, Jeffrey S. Vitter. The input/output complexity of sorting and related problems. *Commun. ACM*, 31(9):1116–1127, 1988.

[2] Rudolf Bayer, Edward M. McCreight. Organization and Maintenance of Large Ordered Indices. *Acta Inf.* 1: 173–189, 1972.

[3] Paul Beame, Faith E. Fich. Optimal bounds for the predecessor problem. *STOC*, 295–304, 1999.

[4] Peter van Emde Boas. Preserving order in a forest in less than logarithmic time and linear space. *SFCS*, 6(3):75–84, 1975.

[5] Gerth S. Brodal, Rolf Fagerberg. Lower bounds for external memory dictionaries. *SODA*, 546–554, 2003.

[6] Adam Buchsbaum, Michael H. Goldwasser, Suresh Venkatasubramanian, Jeffery Westbrook. On external memory graph traversal. *SODA*, 859–860, 2000.

[7] Mihai Pătraşcu, Mikkel Thorup. Time-space trade-offs for predecessor search. *STOC*, 232–240, 2006.

CS 229r: Algorithms for Big Data Fall 2015

Lecture 24 — November 24, 2015

Prof. Jelani Nelson *Scribes: Zhengyu Wang*

1 Cache-oblivious Model

Last time we talked about *disk access model* (as known as DAM, or *external memory model*). Our goal is to minimize I/Os, where we assume that the size of the disk is unbounded, while the memory is bounded and has size M. In particular, the memory is divided into $\frac{M}{B}$ pages, each of which is a block of size B. Today we are going to continue trying to minimize I/Os, but we are going to look at a new model called *cache-oblivious model* introduced in [FLPR99] (you can also refer to the survey [Dem02] for more detail). The new model is similar to DAM, but with two differences (we further refer them as two assumptions):

1. Algorithms are not allowed to know M or B.
2. Algorithms do not control cache replacement policy. Operating system handles cache replacement, and we assume it makes optimal choices. (So in our analysis, we assume that we evict what we want to evict.)

Why do we have cache-oblivious model? First, it makes programs easily portable across different machines. You do not have to find parameters in your code for a specific machine. Note that the block size B is chosen to amortize against the expensive cost of seeking on the disk. In really, B is not fixed, because even on a given disk, there are multiple levels of memory hierarchy (L1/L2 cache, memory and disk), and we have different effective B's to get the optimal amortized performance guarantee. Second, your code might actually be running on a machine that are also running a lot of other processes at the same time. So the effective M used by your process might change over time. Therefore, our first assumption that we do not know M or B makes the model more robust.

When first looking at the second assumption, it seems unrealistic to know optimal choices. In particular, the optimal choices depend on the future, because we should evict pages that would not be used in the near future. Actually, we can show that Assumption 2 is not a very idealized assumption, and it is fine to assume that the operation system knows about the future. The reason for that is related to some facts about online algorithms. In the following, we have a brief detour for online algorithms, in order to justify the optimal choices from the operating system.

Before doing the justification, let us make sure the model is not completely crazy. We actually have I/O efficient algorithms that do not know M or B beforehand. We have already seen one in the last lecture, which was scanning an array. Even if we do not know B, we can store elements in a continuous array. Then when you scan the array, the I/O complexity is $\frac{M}{B}$. So the bound depends on B, although the code does not know B.

1.1 Online Algorithms

The idea of the online algorithms is, we have a sequence of events, and after each event we must make an irreversible decision. One example of online problem is ski rental problem. Assume that you and your friends are on vacation. You do not have preference on how long the vacation is.

- Every morning you wake up at the resort, you ask your friends if "you want to continue skiing tomorrow" or "finish the vacation". Your friends say "continue skiing" or "we're done".
- In terms of expenses for skis, you have two options. The first option is to buy skis, which takes \$10 (no refunds). The second option is to rent skis, which takes \$1.

On each day, if your friends decide to continue skiing, you need to decide whether buy skis or rent skis. Once you buy skis, in the future you do not need to pay extra expenses. The goal is to minimize cost ratio versus an omniscient being who knows future. The ratio is called "competitive ratio". Let D =#days skiing, then $OPT = \min\{D, 10\}$. If our strategy is to rent for the first 9 days, and buy on 10-th day, then we have worst case ratio 1.9.

If we are allowed to use randomness when making decisions, an expected competitive ratio of $\frac{e}{e-1}$ can be achieved[1]. On the other hand, we have lower bound for deterministic algorithm, and a competitive ratio of $2 - o(1)$ is the best possible (as the cost of buying skis goes to infinity).

1.2 Paging Problem

The problem is studied in [ST85]. In the paging problem, the memory can hold $k = \frac{M}{B}$ pages, and we have a sequence of page access requests. Just like the DAM model we have seen, if the page (a page is a block now) is in the memory, we can get access to it for free; if the page is not in the memory, we have to fetch it, bring it in, and evict some page in the memory (if the cost is 1, we get exactly the DAM model). In our situation, the online problem is choosing how to evict memory. Again, we do not know the future. We have to decide which to evict on the fly. The omniscient algorithm would evict the page that will be fetched again farthest in the future (in time). But we don't know the future, so what to do in the real system? Two commonly used strategies/algorithms are:

LRU (least recently used): for each page in the memory, keep track of when most recently I touched the page. And the page furthest back to the past is the one that we choose to evict.

FIFO (first-in / first-out): we evict the oldest page in memory.

These strategies are nice because of the following fact.

Theorem 1 (Sleator-Tarjan [ST85])**.** *FIFO and LRU are:*

1. *k-competitive against OPT.*
2. *2-competitive against OPT when OPT is given $k/2$ memory.*

[1]The result is covered in CS224 Fall 2014, `http://people.seas.harvard.edu/~minilek/cs224/lec/lec10.pdf`

Why does this justify the Assumption 2 of the cache-oblivious model? Well, as long as $T(N, M, B) = \Theta(T(N, M/2, B))$, where $T(\cdot, \cdot, \cdot)$ is the cost given by the analysis of our cache-oblivious algorithm, then Theorem 1 implies that using FIFO or LRU instead of the assumed OPT results in no (asymptotic) loss in performance.

2 Some Cache-oblivious Algorithms

Now we are going to look at some cache-oblivious algorithms[2].

2.1 Array traversal/Reversal

For traversal, as mentioned in the last lecture, the DAM algorithm is actually cache oblivious: we just scan the array in blocks of size B at a time. I/O cost is still at most $O(1 + N/B)$.

For reversal, we can traverse the array backwards and forwards and swap along the way. So by traversal cost, cost for reversal is $O(1 + N/B)$.

2.2 Square Matrix Multiplication

Here our DAM algorithm from last time does not carry over to the cache-oblivious model, since we explicitly broke up the matrix into sub-matrices of size $\sqrt{\frac{M}{2}}$ by $\sqrt{\frac{M}{2}}$. But we are still able to do something simple. Note that we can choose how things layout in the memory. We recursively construct our layout. We first split our matrices into four blocks such that:

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{pmatrix},$$

reducing multiplication of $N \times N$ matrices to eight multiplications and four additions of $N/2 \times N/2$ matrices. Moreover, we will store our matrices A and B on disk as follows.

A_{11}	A_{12}	A_{21}	A_{22}

B_{11}	B_{12}	B_{21}	B_{22}

Then we apply this construction recursively (until the sub-matrix we want to store is 1×1) for each A_{ij} and B_{ij}. For example, A_{11} will be further decomposed (within the decomposition of A above) as follows.

$$\begin{pmatrix} (A_{11})_{11} & (A_{11})_{12} \\ (A_{11})_{21} & (A_{11})_{22} \end{pmatrix}$$

$(A_{11})_{11}$	$(A_{11})_{12}$	$(A_{11})_{21}$	$(A_{11})_{22}$

This gives us a recursive algorithm for matrix multiplication.

Let's analyze number $T(N)$ of I/Os. We have 8 recursive multiplications, and the additions just require scans over $O(N^2)$ entries. Thus recurrence is given by $T(N) = 8\,T(\frac{N}{2}) + O(1 + \frac{N^2}{B})$. The base case is $T(\sqrt{M}) = O(\frac{M}{B})$ for $N \leq \sqrt{M}$, since we can read an entire $\sqrt{M} \times \sqrt{M}$ matrix into memory (due to the recursive data layout!). Solving this gives $T(N) = O(\frac{N^2}{B} + \frac{N^3}{B\sqrt{M}})$. For $N \geq M$, $\frac{N^3}{B\sqrt{M}}$ dominates $T(N)$, and we get $T(N) = O(\frac{N^3}{B\sqrt{M}})$.

[2]This section borrows largely from notes of CS229r Fall 2013 Lecture 22 scribed by Arpon Raksit.

Remark 2. *This technique of recursively laying out data to get locality, and then using M and B to get a good base case of our analysis, will be quite useful in many situations.*

2.3 Linked Lists

We want to support the following three operations:

- Insert(x, p): insert element x after p;
- Delete(p): delete p;
- Traverse(p, k): traverse k elements starting at p.

Shooting for: $O(1)$ each insertion and deletion, and $O(1+k/B)$ to traverse k elements (amortized).

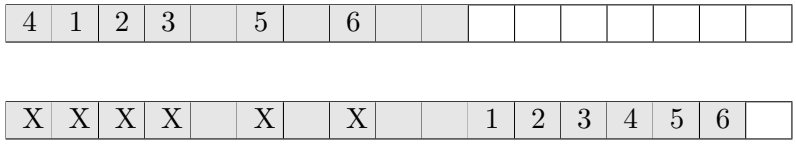
Data structure: maintain an array where each element has pointers to the next and previous locations that contain an element of the list. But it will be self-organizing.

Insertion(x, p): append element x to end of array. Adjust pointers accordingly. It costs $O(1)$ I/Os.

Deletion(p): mark the array location specified by p as deleted. It costs $O(1)$ I/Os.

But now elements might be far apart in the array, so on traversal queries we're going to fix up the data structure (this is the self-organising part).

Traverse(p, k): we traverse as usual using the pointers. But in addition, afterwards we delete the k elements we traversed from their locations and append them to the end of the array.



Rebuild: finally, after every $\frac{N}{2}$ operations (traverse counts as k operations), rewrite entire data structure to a new contiguous array. Free the old one. This increases amortized complexity of each operation by $O(\frac{1}{B})$.

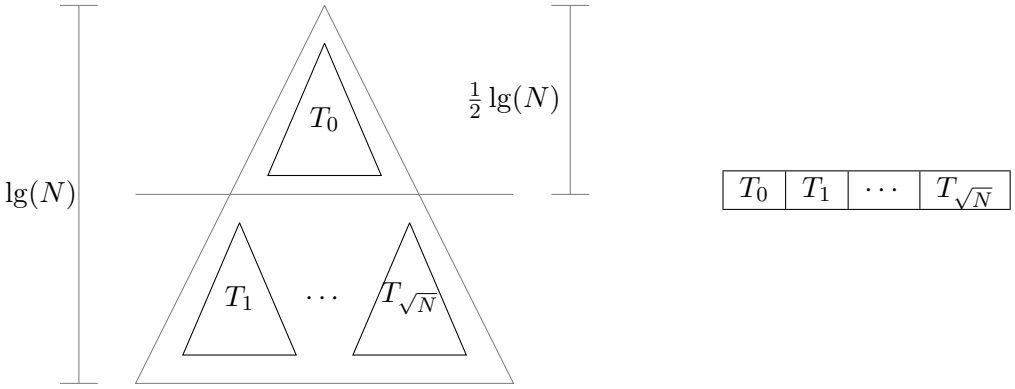
Now, what does traversal cost? When we do a traversal, we touch r contiguous runs of elements. Thus the number of I/Os in the traversal is $O(r + k/B)$——one I/O for each run, and the cost of a scan of the k elements. But there must have been r updates to cause the gaps before each run. We amortize the $O(r)$ over these r updates, so that traversal costs $O(k/B)$ amortized. (To be more precise: any sequence of a insertions, b deletions, and a total of k items traversed costs $O(a + b + 1 + k/B)$ total I/Os.) And since after a traversal we consolidate all of the runs, the r updates we charged here won't be charged again.

> "One money for me means one I/O." – Jelani Nelson

That's amortized linked lists. See [BCD+02] for worst-case data structure.

2.4 Static B-tree

We're going to build a B-tree (sort of) without knowing B. The data structure will only support queries, that is, no insertions [FLPR99]. For dynamic B-tree, refer to [BDFC05]. We will use another recursive layout strategy, except with binary trees. It looks as follows (conceptual layout on left, disk layout on right). Keep in mind this picture is recursive again.



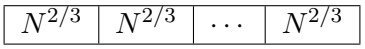
We query as usual on a binary search tree. To analyse the I/O cost, consider the first scale of recursion when the subtrees/triangles have at most B elements. Reading in any such triangle is $O(1)$ I/Os. But of course there are at least $\sqrt{B}$ elements in the tree, so in the end traversal from root to leaf costs $O(2 \cdot \log(N)/\log(\sqrt{B})) = O(\log_B N)$ I/Os.

2.5 Lazy Funnel Sort

Original funnel sort is due to [FLPR99], simplified by [BFJ02]. Yet another recusrive layout strategy, but a lot funkier. Assume we have the following data structure.

Definition 3. *A* K-funnel *is an object which uses* $O(K^2)$ *space and can merge* K *sorted lists of total size* K^3 *with* $O((K^3/B)\log_{M/B}(K^3/B) + K)$ *I/Os.*

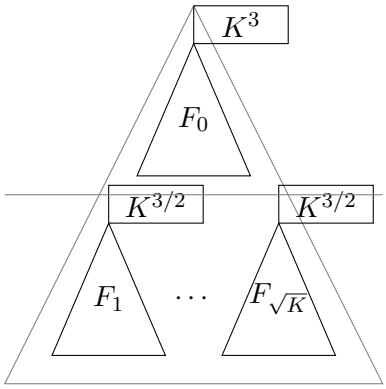
Lazy funnel sort splits the input into blocks of size $N^{2/3}$, recuseively sorts each block, and merges blocks using the K-funnel, with $K = N^{1/3}$.



When analysing this, we will make the following *tall cache assumption*. Unfortunately this assumption is required to get the desired $O((N/B)\log_{M/B}(N/B))$ I/Os for sorting [BFJ02].

Assumption 4 (Tall cache)**.** *Assume* $M = \Omega(B^2)$*. But note that this can be relaxed to* $M = \Omega(B^{1+\gamma})$ *for any* $\gamma > 0$*.*

We're running low on time so let's just see what the K funnel is. It's another recursive, built out of $\sqrt{K}$-funnels. The funnels are essentially binary trees, except with buffers (the rectangles, with labelled sizes) attached.



F_0	buffers	F_1	$\cdots$	$F_{\sqrt{K}}$

At each level the $\sqrt{K}$ buffers use $O(K^2)$ space. Then the total space used is given by the recurrence $S(K) = (1 + \sqrt{K})S(\sqrt{K}) + S(K^2)$. Solving this gives $S(K) \leq O(K^2)$.

How do you use a K-funnel to merge? Every edge has some buffer on it (which all start off empty). The root node tries to merge the contents of the buffers of the two edges to its children. If they are empty, the root recursively asks its children to fill their buffers, then proceeds to merge them. The recursion can go all the way down to the leaves, which are either connected to the original K lists to be merged, or are connected to the output buffers of other funnels created at the same level of recursion (in which case you recursively ask them to fill their output buffers before merging).

Theorem 5. *As described, lazy funnel sort costs $O((N/B)\log_{M/B}(N/B))$ I/Os (under the tall cache assumption).*

Proof sketch. The recurrence above proved funnel property (1). For funnel property (2), look at the coarsest scale of recursion where we have J funnels, with $J \ll \sqrt{M}$. The details are in [Dem02]. □

References

[BCD+02] Michael A Bender, Richard Cole, Erik D Demaine, Martin Farach-Colton, and Jack Zito. Two simplified algorithms for maintaining order in a list. In *Algorithms?ESA 2002*, pages 152–164. Springer, 2002.

[BDFC05] Michael A Bender, Erik D Demaine, and Martin Farach-Colton. Cache-oblivious b-trees. *SIAM Journal on Computing*, 35(2):341–358, 2005.

[BFJ02] Gerth Stølting Brodal, Rolf Fagerberg, and Riko Jacob. Cache oblivious search trees via binary trees of small height. In *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 39–48. Society for Industrial and Applied Mathematics, 2002.

[Dem02] Erik D Demaine. Cache-oblivious algorithms and data structures. *Lecture Notes from the EEF Summer School on Massive Data Sets*, 8(4):1–249, 2002.

[FLPR99] Matteo Frigo, Charles E Leiserson, Harald Prokop, and Sridhar Ramachandran. Cache-oblivious algorithms. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*, pages 285–297. IEEE, 1999.

[ST85] Daniel D Sleator and Robert E Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.

CS 229r: Algorithms for Big Data Fall 2015

Lecture 26 — December 1, 2015

Prof. Jelani Nelson *Scribe: Zezhou Liu*

1 Overview

Final project presentations on Thursday. Jelani has sent out emails about the logistics: 8 minute presentation with 1-2 minutes for questions. Groups can split up the work however they want (one person can make the slides and the other presents, etc.)

This time, we want to cover algorithms in a distributed environment, maybe 1M machines. How to do this efficiently? Work starts as early as ('76 Csanky)[1], with a survey by ('88, Eppstein, Galil) on PRAM.

PRAM: ignore communication (shared memory across all processors) and synchronization. The goal is to understand time vs. number of processors. However, shared memory is unrealistic, and somewhere in late 80s/early 90s, PRAM wasn't studied much anymore because you had to consider communication and syncrhonization.

Bulk Synch Parallel (BSP) [BSP]: You should explicitly worry about communication and synchronization. BSP today: Apache, Hama, Google, Pragel.

Today: MapReduce [DG08]. This system was built and used at Google to deal with massively parallel jobs. Hadoop is an open-source version. This system is used by Google, Facebook, Amazon, ...

How do we work with these systems and build efficient algorithms on these models?

2 MapReduce

In this model, data items are each $\langle key, value\rangle$ pairs. Computation is broken up into rounds:

Round:

- Map: Each item is processed by some *map* function, and emits a set of new $\langle key, value\rangle$ pairs.
- Shuffle: This step is oblivious to the programmer. All items emitted in the map phase are grouped by key, and items with the same key are sent to the same reducer.
- Reducer: Receives $\langle k, v_1, v_2, ..., v_3\rangle$ and emits new set of items.

Goals:

- Few number of rounds
- Want to use $<< n$ mem per reducer.

- Want $<< n^2$ total mem used.
- Small total work (# machines * max(work per machine))
- Small parallel work (What max(time per machine)?)

3 MapReduce Problems

Example: Sorting.

Theorem 1. *TeraSort (O'Malley, '08) [2]. Sorts arbitrary and comparable elements (although the original was on bounded integers and used tricks to speed that up).*

Input elements are $\langle i; A[i] \rangle$, where A is an unsorted array A[1..n].

Say we want to use p-machines in parallel. So we want to divide it into the smallest n/p elements, next smallest n/p elements, etc. so we can send the jth n/p elements to machine j to be sorted.

Two-round algorithm:

Round 1: Sample $T = log(p)/\varepsilon^2)$

def map1 $\langle i, A[i] \rangle$:

    emit $\langle i\%p, A[i] \rangle$

    w.p T/n:

        for j = 0...p:

            emit $\langle j, A[i] \rangle$

First partition the values, and then for some random samples, send to some reducer.

def reduce1 $\langle j; X \rangle$:

    B are the (i, A[i]), and S is set of sampled elements

    S $\leftarrow$ sort(S)

    for each (i, x) $\in$ B:

        find some r $\in 0, ...p-1$ that x should map to in relation to S, and emit $\langle r; (i, x) \rangle$

Round 2:

def map2: identity

def reduce2 $\langle j, B \rangle$:

    Sort B by 2nd element (recall B is (i, x))

    write output to j.out

At this point, each machine has their own sorted list, so we just need to concatenate the sorted lists from each machine.

Analysis: First, I have to choose epsilon. There are two things I want to balance: 1) Each reducer is getting set of size n/p, but also getting some additional sampled elements $log(p)/\varepsilon^2$. So want $log(p)/\varepsilon^2 \leq n/p$. 2) Chernoff bound says: if we sample $T = log(1/delta) * C/\varepsilon^2$ elements and look at αth smallest element in sample, its rank in actual sorted order of A is $\alpha n + -\varepsilon n$ w.p. $1 - \delta$. Union bound says that with high probability, each reducer in the second round gets $\leq n/p + \varepsilon n$ items to sort. (set $\varepsilon = 1/p \rightarrow T = log(p)\ p^2$)

Going back, we want $p^2 lg(p) \leq n/p$, so $p = n^{1/3}/lg(n) \rightarrow$ with high probability each of the p machines needs to sort $\leq O(n/p) = O(n^{2/3} lg(n))$.

4 Min Spanning Tree (MST)

In the case where it is not too sparse, with $m = \#edges = n^{1+c}$ for some $c > 0$. We want memory per machine to be $<< m(m^{1-\delta})$. Algorithm by (Karloff, Suri, Vassilivitskii, SODA '10)[3].

For each pair $(i, j) \in [k]^2$:

- Let $G_{ij} = (V_i \cup V_j, E_{ij})$ be the induced graph on $V_i \cup V_j$
- Compute using any sequential algorithm $M_{ij} = MSF(G_{ij})$
- Send $H = \bigcup_{i,j} M_{ij}$ to a single reducer and output $M = MST(H)$.

MapReduce:

Round1:

def map1 $\langle (u, v); NULL \rangle$:

emit $\langle h(u), h(v); (u, v) \rangle$

if h(u) == h(v) = i:

for each j = 1, ..., k

emit $\langle (i, j); (u, v) \rangle$

def reduce1 $\langle (i, j), E_{i,j} \rangle$:

let $e_1, ..., e_t$ be MSF of E_{ij}

for a = 1..t:

emit$\langle \$, (e_a, (i, j)) \rangle$

This is emitting all the edges of $M_{i,j}$.

def map2: identity

def reduce2: Take all previous edges and call MST of this set of edges.

Analysis: (memory) - mem per machine round 2: $\leq k^2 \cdot O(n/k) = O(kn) = O(n^{1+c/2}) << m = n^{1+c}$ - mem per machine round 1: $\max_i, j\ |E_{i,j}|$. The expected size of $E_{i,j} = \sum_{p \in E} \mathbb{P}$(endpoints of e in $Vi \cup Vj) = 2m/k$

For Chernoff, we have that $\mathbb{E}_{i,j} \leq \sum_{v \in V} \mathbb{1}_{h(u) \in \{i,j\}} \cdot deg(v)$. You can do better than this, as shown in KSV '10: partition V according to degrees. $V_t = \{v : 2^{t-1} \leq deg(v) < 2^t\}$. This means with high probability $\max_{i,j} |E_{i,j}| = O(n^{1+c/2})$

Downside: Total memory needed $\sim k^2 \cdot n^{1+c/2} = n^{1+3c/2} = n^{2+c/2} = m^{2-\varepsilon}$ (Lattanzi et all, SPAA '11) with $O(n^c)$ machines, $O(n^{1-\varepsilon})$ mem per machine, and ceil(c/ε) rounds.

5 Triangle Counting

For Triangle Counting, the input is an undirected graph, and we want to output $|\{u < v < w \in V | (u,v),(v,w),(w,u) \in E\}|$. With no parallelism, there is a simple $O(n^3)$ algorithm using 3 nested for-loops by looping over all (u,v,w). You can even get $m^{3/2}$. MapReduce can implement this with $\sqrt(m)$ machines.

$x \leftarrow 0$
  for $v \in V$
    for $u \in \Gamma(v)$ s.t. $\deg(u) \geq \deg(v)$
      for $w \in \Gamma(v)$ s.t. $\deg(w) \geq \deg(v)$
        if $(u,w) \in E$
          $x \leftarrow x + 1$
return $x/2$

We can implement the above algorithm in MapReduce, with the the following properties:

- No reducer gets more than $O(\sqrt{m})$ items
- The total work done is $O(m^{\frac{3}{2}})$
- The number of rounds is 2

5.1 MapReduce algorithm

Round 1:
def map1: The input is $< (v,u); \varnothing >$.

    if $u \succ v$: emit $\langle v,u \rangle$

def reduce1: The input is $< v; S \subseteq T(v) >$.

    for $< u,w > \in S$:

        emit $< u,w >$ if is an edge.

Round 2:

def map2:

    if input is $< v,(u,w) >$:

        emit $< (u,w); v >$

    if input is $< (u,w), \varnothing) >$:

emit $< (u, w), \$ >$

def reduce2 $< (u, w); S >$:

if $\$ \in S$: then

for each $v \in S$ s.t. $v \neq \$$:

emit $< v, 1 >$

References

[1] Csanky, L., Fast parallel matrix inversion algorithms, *SIAM J. Computing* 5, 1976,

[BSP] Leslie G. Valiant. 1990. A bridging model for parallel computation. *Commun. ACM 33*, 8 (August 1990), 103-111. DOI=10.1145/79173.79181 http://doi.acm.org/10.1145/79173.79181

[DG08] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[2] O'Malley, O. TeraByte sort on Apache Hadoop, 2008;

[3] Howard J. Karloff, Siddharth Suri, Sergei Vassilvitskii. A Model of Computation for MapReduce. *SODA 2010*: 938-948.

[4] Silvio Lattanzi, Benjamin Moseley, Siddharth Suri, Sergei Vassilvitskii. Filtering: a method for solving graph problems in MapReduce. *SPAA 2011*: 85-94.

[5] Siddharth Suri, Sergei Vassilvitskii. Counting triangles and the curse of the last reducer *WWW 2011*: 607-614.