CS 15-859: Algorithms for Big Data Fall 2019

Lecture 6 — 2019-10-10

Prof. David Woodruff *Scribe: Wan Shen Lim*

These scribe notes cover slides 103 to 116 inclusive.

1 Hints from Office Hours

1.1 Question 1: Matrix Multiplication Runtime

How long does it take to multiply two matrices?

$$\underset{n\times d}{\left[A\right]} \cdot \underset{d\times k}{\left[x\right]}$$

Naively, it requires $O(ndk)$ operations, but we can also do it in $O(k \cdot \text{nnz}(A))$ operations.

1.2 Question 3: Motivating Leverage Scores

Leverage scores tell us how "important" a row is. Consider the below matrix, $n = 6$ and $d = 2$.

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & -1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}$$

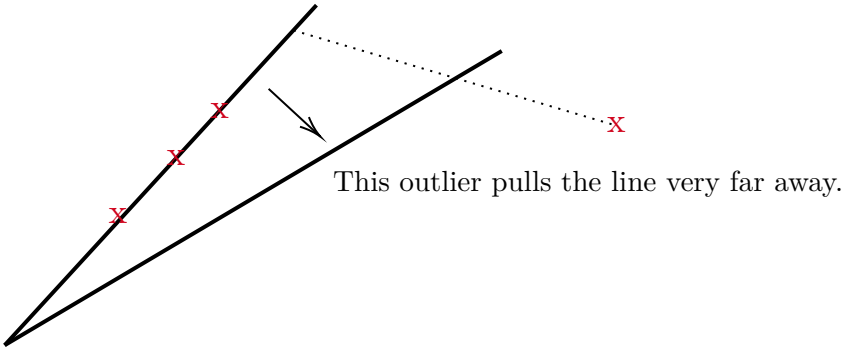
Notice that the squared row norm is the same for every row. Therefore, if we sample by the squared row norms, we are unlikely to pick the $\begin{bmatrix} 1 & -1 \end{bmatrix}$ row. But A has rank 2, and if we miss that row, the sampled matrix A' will only have rank 1. How can we favor sampling that row? Consider performing elementary row operations to get an orthonormal basis:

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & -1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \to \begin{bmatrix} 2 & 0 \\ 2 & 0 \\ 2 & 0 \\ 0 & -1 \\ 2 & 0 \\ 2 & 0 \end{bmatrix} \to \begin{bmatrix} \frac{1}{\sqrt{n-1}} & 0 \\ \frac{1}{\sqrt{n-1}} & 0 \\ \frac{1}{\sqrt{n-1}} & 0 \\ 0 & 1 \\ \frac{1}{\sqrt{n-1}} & 0 \\ \frac{1}{\sqrt{n-1}} & 0 \end{bmatrix} \underset{\text{writing leverage scores on the left}}{\to} \begin{matrix} \frac{1}{n-1} \\ \frac{1}{n-1} \\ \frac{1}{n-1} \\ 1 \\ \frac{1}{n-1} \\ \frac{1}{n-1} \end{matrix} \begin{pmatrix} \frac{1}{\sqrt{n-1}} & 0 \\ \frac{1}{\sqrt{n-1}} & 0 \\ \frac{1}{\sqrt{n-1}} & 0 \\ 0 & 1 \\ \frac{1}{\sqrt{n-1}} & 0 \\ \frac{1}{\sqrt{n-1}} & 0 \end{pmatrix}$$

Notice that the important row now has a huge leverage score!

2 ℓ_1-Regression

Note that ℓ_2-regression is sensitive to outliers as you are penalized by the square of the distance.



This motivates ℓ_1-regression, in which we find the x^* minimizing $|Ax - b|_1 = \sum_i |b_i - \langle A_{i*}, x\rangle|_1$.

We can reformulate ℓ_1-regression as the following linear programming problem,

$$\begin{aligned} &\text{minimize } (1, \cdots, 1) \cdot (\alpha^+ + \alpha^-) \\ &\text{subject to } Ax + \alpha^+ - \alpha^- = b \\ &\alpha^+, \alpha^- \geq 0 \end{aligned}$$

which can be solved in poly(nd) time by generic linear programming.

But poly(nd) is slow. We can sketch to go faster. Given $A_{n\times d}$ and b, this is where we're going:

1. Compute a poly(d)-approximation to our ℓ_1-regression problem.
   i.e. we will find x' such that $|Ax' - b|_1 \leq \text{poly}(d) \min_{x\in\mathbb{R}^d} |Ax - b|_1$
2. Compute a well-conditioned basis.
   i.e. this basis keeps lengths within poly(d) of the original, $\frac{|x|_1}{\text{poly}(d)} \leq |Ux|_1 \leq \text{poly}(d)|x|_1$
3. Sample rows from the well-conditioned basis and the residual of the poly(d) approximation.
   i.e. we will prove the problem equivalence of $\min_{x\in\mathbb{R}^d} |Ax - b|_1 = \min_{x\in\mathbb{R}^d} |Ux - b'|_1$ where $b' = b - Ax'$. It turns out sampling poly$(\frac{d}{\epsilon})$ rows proportional to their ℓ_1-norm is sufficient to obtain a $(1+\epsilon)$ approximation.
4. Now we only have poly$(\frac{d}{\epsilon})$ constraints, and generic linear programming is efficient!

2.1 Recap

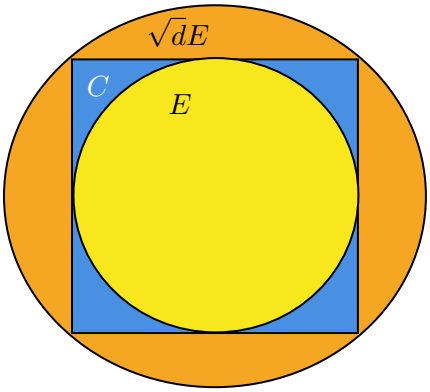
2.1.1 Well-Conditioned Basis

Intuition. Our goal: for any matrix $A_{n\times d}$, you can find a U preserving $|x|_1$ by up to a $\sqrt{d}$ factor.

Recall that for ℓ_2-regression, we were given a matrix $A_{n\times d}$ and were able to find a matrix $U_{n\times d}$ with orthonormal columns such that $A = UW$ and $|Ux|_2 = |x|_2$ for all x.

We want a similar result for ℓ_1-regression, i.e. given a matrix $A_{n\times d}$, we want to find a matrix $U_{n\times d}$ such that $A = UW$ and $|Ux|_1 \approx |x|_1$ for all x. It turns out we can preserve up to $\sqrt{d}$.

Lowner-John. Previously, we defined $|z|_{Q,1} = |Qz|_1$ and proved that it was a norm. We also considered the $|\cdot|_{Q,1}$ unit ball as $C = \left\{z \in R^d : |z|_{Q,1} \le 1\right\}$. We showed that C is a convex set which is symmetric about the origin. Pictured below is the ℓ_∞-ball in $\mathbb{R}^2$, i.e. a square.



Note that an ellipsoid is defined by the solutions to the equation $z^TFz = 1$ where F is a positive definite matrix. Since F is positive definite, there exists G such that $F = GG^T$.

The Lowner-John theorem says that given C, we can find an ellipsoid E such that $E \subseteq C \subseteq \sqrt{d}E$ where $E = \left\{z \in \mathbb{R}^d : z^TFz \le 1\right\}$, C is sandwiched between ellipsoids E and $\sqrt{d}E$.

Equivalently, $\sqrt{z^TFz} \le |z|_{Q,1} \le \sqrt{d}\sqrt{z^TFz}$ for all z.

Finding our basis. We claim that $U = QG^{-1}$ is a well-conditioned basis. Let $z = G^{-1}x$. Then

1. $|Ux|_1 = |QG^{-1}x|_1 = |Qz|_1 = |z|_{Q,1}$
2. $z^TFz = \left(\left(x^T\left(G^{-1}\right)^T\right)\left(G^TG\right)\left(G^{-1}x\right)\right) = x^Tx = |x|_2^2$

Now recall that by the Lowner-John theorem, for all z we have that

$$\sqrt{z^TFz} \le |z|_{Q,1} \le \sqrt{d}\sqrt{z^TFz}$$
$$|x|_2 \le |Ux|_1 \le \sqrt{d}|x|_2 \quad \text{by substitution}$$

A useful fact is that $\forall y \in \mathbb{R}^k, |y|_2 \le |y|_1 \le \sqrt{k}|y|_2$. Using that fact, we obtain that

$$\frac{|x|_1}{\sqrt{d}} \le |x|_2 \le |Ux|_1 \le \sqrt{d}|x|_2 \le \sqrt{d}|x|_1$$

and so we are done; we see that U preserves $|x|_1$ up to a $\sqrt{d}$ factor.

Note. Notice that we never talked about algorithms for computing the ellipsoid. It turns out that there are algorithms for finding an ellipsoid E in time poly(nd) or even $n \cdot$ poly(d), but we won't cover them because our algorithm will be faster anyway.

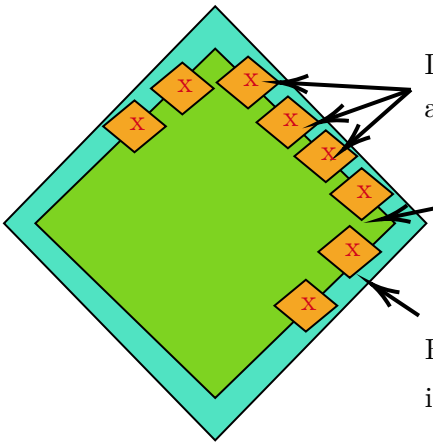
2.1.2 Net for ℓ_1-Ball

Intuition. A γ-net is a set of points "covering" the unit ball's surface so that no point is further than γ away. Recall γ-nets with the ℓ_2-norm. In bounding the number of points needed for the net construction, we only used the triangle inequality. Therefore the same argument holds for ℓ_1-norms.

We want to construct a γ-net for the unit ℓ_1-ball $B = \left\{x \in \mathbb{R}^d : |x|_1 = 1\right\}$. Recall the definition of a γ-net: subset N is a γ-net if for all $x \in B$, there exists $y \in N$ such that $|x - y|_1 \leq \gamma$.

Greedy algorithm is good enough. To construct N, a simple greedy algorithm suffices: while there exists a point $x \in B$ of distance larger than γ for every point in N, we include x in N.

Now consider drawing a ℓ_1-ball of radius $\frac{\gamma}{2}$ around every point in N. By the greedy construction, every such ball is disjoint (otherwise we wouldn't have picked the point). Moreover, every such ℓ_1-ball is contained within a large ℓ_1-ball of radius $1 + \frac{\gamma}{2}$ centered at 0^d.



Disjoint balls of radius $\frac{\gamma}{2}$ around points in the net

Original ℓ_1-ball

By triangle inequality, every point is contained within a $\left(1 + \frac{\gamma}{2}\right)$-ball

Useful fact: stretching by α in a d-dimensional space increases volume by α^d. Here, $\alpha = \frac{1+\frac{\gamma}{2}}{\frac{\gamma}{2}}$. In particular, the ratio of the volume of d-dimensional polytopes of radius $1 + \frac{\gamma}{2}$ compared with those of radius $\frac{\gamma}{2}$ is $\frac{\left(1+\frac{\gamma}{2}\right)^d}{\left(\frac{\gamma}{2}\right)^d} \geq |N|$. The last inequality follows because there can only be that many balls of radius $\frac{\gamma}{2}$.

Take $\frac{\left(1+\frac{\gamma}{2}\right)^d}{\left(\frac{\gamma}{2}\right)^d}$ points. In summary, we can create a γ-net by taking at most $\frac{\left(1+\frac{\gamma}{2}\right)^d}{\left(\frac{\gamma}{2}\right)^d}$ points.

2.1.3 Net for ℓ_1-Subspace

Intuition. We obtained a net for the ball, now we want to extend the net to cover the subspace. In doing so, we will lose a factor of d, but in practice this doesn't really matter as it just introduces a $\log d$ factor somewhere..

Just multiply the net by the basis. Let $A = UW$ where U is a well-conditioned basis, and N be a $\frac{\gamma}{d}$-net for the unit ℓ_1-ball B. We set $M = \{Ux : x \in N\}$, and by the above net for ℓ_1-ball argument we know that $|M| \leq \frac{\left(1+\frac{\gamma}{2}\right)^d}{\left(\frac{\gamma}{2}\right)^d}$.

But now we claim that for every x in B, there is already a $y \in M$ such that $|Ux - y|_1 \leq \gamma$.

1. Because N is a $\frac{\gamma}{d}$ net, we know there exists $x' \in N$ such that $|x - x'|_1 \leq \frac{\gamma}{d}$.
2. Therefore $|Ux - Ux'|_1 \leq |U|_1 |x - x'|_1$ by the triangle inequality.
3. Since U is a well-conditioned basis, we have that $|x|_1 \leq |Ux|_1 \leq dx$ for all x. Note that this is a loose bound. Now, $x - x'$ is just another point, so $|U|_1 \leq d|x - x'|_1$.
4. But then $d|x - x'|_1 \leq \gamma$ by the choice of x' in the first step.
5. And so by picking $y = Ux'$, we get that $|Ux - Ux'|_1 \leq \gamma$ as desired.

Take $\left(\frac{d}{\gamma}\right)^{O(d)}$ points. From above, we have that $|M| \leq \frac{\left(1+\frac{\gamma}{2}\right)^d}{\left(\frac{\gamma}{2}\right)^d} \frac{\left(\frac{2d}{\gamma}\right)^d}{\left(\frac{2d}{\gamma}\right)^d} = \frac{\left(\frac{2d}{\gamma}\right)^d + d^d}{d^d} \leq \left(\frac{d}{\gamma}\right)^{O(d)}$.

2.2 Algorithm for ℓ_1-regression

So we've shown so far that we can γ-net a ℓ_1-subspace. This will be useful later. Let's go back to discussing our algorithm for solving ℓ_1-regression. There are four parts, given $A_{n \times d}$ and b,

1. Compute poly(d) approximation.
   Find x' such that $|Ax' - b|_1 \leq \text{poly}(d) \min_x |Ax - b|_1$.
2. Compute well-conditioned basis.
   Find U with the same column-span as A such that for all x, $\frac{|x|_1}{\text{poly}(d)} \leq |Ux|_1 \leq \text{poly}(d)|x|_1$.
3. Sample rows from $U \circ b'$ where $b' = Ax' - b$.
   Recall that in ℓ_2-regression, leverage score sampling of the matrix $A \circ b = [A, b]$ gave a good approximation. In ℓ_1-regression, we do not have leverage scores (e.g. scores depend on basis).
   However, let $b' = Ax' - b$ and sample poly $\left(\frac{d}{\epsilon}\right)$ rows from $U \circ b' = [U, b']$ proportional to their ℓ_1-norm. This provides a $(1 + \epsilon)$ approximation for ℓ_1-regression. We do not prove this, but the proof is similar to the ℓ_2-regression case shown in previous weeks.
4. Solve ℓ_1-regression on the sample via linear programming and output that as your answer.
   Recall $A = UW$ for well-conditioned basis U. Let $y = Wz$, since W is invertible, $\min_y |Uy - b'|_1 = \min_z |Az - b'|_1 = \min_z |Az + Ax' - b|_1 = \min_x |Ax - b|_1$ by letting $x = z + x'$. So solving the sketch solves the original problem, i.e. linear programming with just poly $\left(\frac{d}{\epsilon}\right)$ constraints!

2.3 Sketching Theorem

Theorem. There exists a probability space over matrices $R_{(d \log d) \times n}$ such that for any matrix $A_{n \times d}$, with probability at least $\frac{99}{100}$ we have that

$$|Ax|_1 \leq |RAx|_1 \leq d \log d \cdot |Ax|_1 \quad \text{for all } x$$

Moreover, R is an embedding: it is linear, independent of A, and preserves the lengths of an infinite number of vectors.

2.3.1 Why we care, part 1: computing a $\mathrm{poly}(d)$ approximation

If we had such an R, we could compute RA and Rb. By solving for $x' = \arg\min_x |RAx - Rb|_1$, the Sketching Theorem then tells us that x' is a $d \log d$ approximation. Furthermore, we can solve for x' efficiently since RA and Rb have $d \log d$ rows.

2.3.2 Why we care, part 2: computing a well-conditioned basis

If we had such an R, we could compute RA and compute W such that RAW is orthonormal in the ℓ_2-sense. Then $U = AW$ is a well-conditioned basis! To see this, recall that $|\cdot|_1 \leq \sqrt{d \log d}|\cdot|_2$ and note that

$$\begin{aligned}
|AWx|_1 &\leq |RAWx|_1 && \text{since } R \text{ a subspace embedding} \\
&\leq \sqrt{d \log d}|RAWx|_2 && \text{going from } \ell_1 \text{ to } \ell_2 \\
&\leq \sqrt{d \log d}|x|_2 && \text{since } RAW \text{ has orthonormal columns} \\
&\leq \sqrt{d \log d}|x|_1 && \text{since } |\cdot|_2 \text{ is at most } |\cdot|_1
\end{aligned}$$

and also

$$\begin{aligned}
|AWx|_1 &\geq \frac{|RAWx|_1}{d \log d} && \text{since } R \text{ a subspace embedding} \\
&\geq \frac{|RAWx|_2}{d \log d} && \text{going from } \ell_1 \text{ to } \ell_2 \\
&\geq \frac{|x|_2}{d \log d} && \text{since } RAW \text{ has orthonormal columns} \\
&\geq \frac{|x|_1}{d^{\frac{3}{2}} \log d} && \text{since } |\cdot|_2 \text{ is at least } \frac{1}{\sqrt{d}}|\cdot|_1
\end{aligned}$$

And so ultimately we have that

$$\frac{|x|_1}{d^{\frac{3}{2}} \log d} \leq |AWx|_1 \leq \sqrt{d \log d}|x|_1$$

Which isn't quite symmetrically bounded, but is well-conditioned enough for our purposes.

2.3.3 Bringing the Sketching Theorem to life

A dense matrix R whose entries are i.i.d. Cauchy random variables and scaled by $\frac{1}{d \log d}$ works!

These scribe notes conclude before we fully see why this is the case. We'll show that we're not too small, the next scribe notes will show that we're not too large. But first, we go on a tangent about the Cauchy distribution.

2.3.4 Cauchy Distribution

Fun fact. The Cauchy distribution is David's favorite distribution.

The probability density function of the Cauchy distribution is given by

$$\text{pdf}(z) = \frac{1}{\pi\left(1+z^2\right)} \quad \text{for } z \in (-\infty, \infty)$$

The Cauchy distribution has fat tails, undefined expectation and infinite variance.

1. Fat-tailed: the pdf is $\frac{1}{\pi(1+z^2)}$, so as $z \to \infty$, the pdf $\to \frac{1}{z^2}$. Contrast this with the Gaussian distribution's behavior, $e^{-\frac{z^2}{2}}$ which rapidly goes to 0!
2. Undefined expectation: note that $\int_{-\infty}^{\infty} \frac{z}{\pi(1+z^2)} \mathrm{d}z \approx \int_{-\infty}^{\infty} \frac{1}{z} \mathrm{d}z$ which diverges.
3. Infinite variance: note that $\int_{-\infty}^{\infty} \frac{z^2}{\pi(1+z^2)} \mathrm{d}z = \infty$.

Recall that the sum of independent Gaussian random variables is Gaussian, i.e. $X \sim N(0, \sigma_1^2), Y \sim N(0, \sigma_2^2), X \perp Y \implies X + Y \sim N(0, \sigma_1^2 + \sigma_2^2)$. We call the Gaussian distribution 2-stable.

It turns out that the Cauchy distribution is 1-stable, meaning that if $z_1, z_2, \cdots, z_n$ are i.i.d Cauchy, then for $a \in \mathbb{R}^n$, we have that

$$a_1 \cdot z_1 + a_2 \cdot z_2 + \cdots + a_n \cdot z_n \sim |a|_1 \cdot z \quad \text{where } z \text{ is Cauchy}$$

Moreover, Cauchy random variables can be generated as the ratio of two standard normal random variables. This is useful for simulation.

2.3.5 Back to Sketching Theorem

We restate the Sketching Theorem.

Theorem. There exists a probability space over matrices $R_{(d\log d)\times n}$ such that for any matrix $A_{n\times d}$, with probability at least $\frac{99}{100}$ we have that

$$|Ax|_1 \le |RAx|_1 \le d\log d \cdot |Ax|_1 \quad \text{for all } x$$

We claimed that R whose entries are i.i.d. Cauchy random variables scaled by $\frac{1}{d\log d}$ works.

1-stability. To see this, note that

1. By 1-stability, for any row r of R we have that $\langle r, Ax\rangle = \frac{|Ax|_1 \cdot Z}{d\log d}$ for some Cauchy Z.
2. But then $RAx = \frac{1}{d\log d}\left(|Ax|_1 \cdot Z_1, |Ax|_1 \cdot Z_2, \cdots, |Ax|_1 \cdot Z_{d\log d}\right)$ for i.i.d. Cauchy $Z_1, \cdots, Z_{d\log d}$.
3. Therefore $|RAx|_1 = \frac{1}{d\log d}|Ax|_1 \sum_{j=1}^{d\log d} |Z_j|_1$. Note that $|Z_j|_1$ is half-Cauchy.

Chernoff. Let $\mathbf{I}$ be the indicator function and W_i be indicator random variables as follows: $W_i = \mathbf{I}(|Z_i|_1 \ge \frac{1}{10})$. The $\frac{1}{10}$ is arbitrary, all that matters is that we're bigger than a constant with a constant probability.

Then $\mathbb{E}[W_i] = \Omega(1) = \mathbf{Pr}[W_i = 1]$, so by linearity of expectation, for $W = \sum W_i$ we have that $\mathbb{E}[W] = \Omega(d\log d)$. Now we can apply Chernoff bounds: $\mathbf{Pr}[W \le \frac{1}{2}\mathbb{E}[W]] \le e^{-\Theta(\mathbb{E}[W])} = e^{-\Omega(d\log d)}$. The tiny Chernoff bound allows us to union bound over the net!

Union-Bounding By combining the 1-stability and Chernoff sections above, we have that $\sum_{j=1}^{d\log d} |Z_j|_1 = \Omega(d\log d)$ with probability $1 - e^{-\Omega(d\log d)}$.

We therefore obtain that $|RAx|_1 = \frac{1}{d\log d}|Ax|_1 \sum_{j=1}^{d\log d} |Z_j|_1 = |Ax|_1 \frac{\Omega(d\log d)}{d\log d}$ with high probability.

This tells us that we are not too small, but recall that $|Z_j|_1$ is heavy-tailed – we could turn out to be too large instead.

Scribe notes end here at slide 116 inclusive.