CS 15-859: Algorithms for Big Data Fall 2019

Lecture 4 — Sept. 26 (Part 1)

Prof. David Woodruff *Scribe: Vaidehi Srinivas*

1 High Precision Regression

1.1 Goals

As always, we want to solve regression, so we want to output x' such that, with high probability,

$$|Ax' - b|_2 \leq (1+\varepsilon) \min_x |Ax - b|_2.$$

But this time we want it fast. We are looking for run time

$$O\left(\text{poly}(d) \cdot \log\left(\frac{1}{\varepsilon}\right)\right).$$

The sketching algorithms that we have seen so far help us solve regression, but they have a polynomial dependence on $\frac{d}{\varepsilon}$, so we are looking for something faster. We will get this better run time by combining sketching with an iterative algorithm.

Another issue that we run into, is the run times of many algorithms[1] depend on the condition number κ, where

$$\kappa(A) = \frac{\sup_{|x|_2=1} |Ax|_2}{\inf_{|x|_2=1} |Ax|_2}$$

We call a matrix with small κ "well-conditioned," and a matrix with $\kappa = 1$ "perfectly conditioned." (Note that κ can never be less than 1.)

We will also see how to use sketching to reduce $\kappa(A)$ to $O(1)$.

1.2 Small QR Decomposition

1. Let S be a $(1 \pm \varepsilon_0)$-subspace embedding for a matrix A.

   Think of S as a CountSketch matrix. Also, ε_0 is a constant, like $\frac{1}{2}$ or something. This is not the actual ε that we are going for, this is just a starting point.

2. We compute SA and the QR-factorization of SA.

   That is, we set

   $$SA = QR^{-1}$$

   for some Q with orthonormal columns. We can do this by taking the SVD of SA: $SA = U\Sigma V^T$, and setting $Q = U$ and $R^{-1} = \Sigma V^T$. Computing SA takes $\text{nnz}(A)$ time, and taking the SVD takes $\text{poly}(d)$ time, since SA has dimensions polynomial in d.

[1] like gradient descent

Fun Fact 1. $\kappa(AR) \leq \frac{1+\varepsilon_0}{1-\varepsilon_0}$

Proof. Since $SA = QR^{-1}$, $SAR = Q$. We know that Q has orthonormal columns, so $|Qx|_2 = |x|_2$ for all vectors x. This means that $Q = SAR$ is perfectly conditioned.

We know that S is a $(1 \pm \varepsilon_0)$-subspace embedding for A. This means for all unit vectors x:

$$(1 - \varepsilon_0)|ARx|_2 \leq |SARx|_2 = 1$$

$$(1 + \varepsilon_0)|ARx|_2 \geq |SARx|_2 = 1$$

where $|SARx|_2 = 1$ since SAR is perfectly conditioned.

Putting it together:

$$\kappa(AR) = \frac{\sup_{|x|_2=1} |ARx|_2}{\inf_{|x|_2=1} |ARx|_2} \leq \frac{1+\varepsilon_0}{1-\varepsilon_0}.$$

■

What did we do? We just found a cute matrix, R, that makes the condition number of A become close to 1, and condition number close to 1 means that our algorithms run fast!

Why did we use an approximation? If we wanted to multiply A by something to get it's condition number down to 1 so badly, we could have just taken the SVD of A directly:

$$A = U_A \Sigma_A V_A$$

and then taken

$$A(\Sigma_A V_A)^{-1} = U.$$

This would have deterministically given us a perfectly conditioned matrix. We didn't do this because $A \in \mathbb{R}^{n \times d}$ is big, so taking the SVD of A would take poly(nd) time, which is slow.

1.3 Finding a Constant Factor Solution

1. Let S be a $(1 \pm \varepsilon_0)$-subspace embedding for AR.

   This is a freshly drawn subspace embedding, not the same one as in the last section.

2. Solve $x_0 = \arg\min_x |SARx - Sb|_2$.

   It takes $\text{nnz}(A) + \text{poly}(d)$ time to compute R and x_0 since ε_0 is a constant. Note that since ε_0 is a constant, this is not a great solution, but we are happy because we can do it fast.

3. Iterate according to this update rule

   $$x_{m+1} \leftarrow x_m + R^T A^T (b - ARx_m)$$

   for $M \in O\left(\log\left(\frac{1}{\varepsilon}\right)\right)$ iterations.

   If we are smart about this, we don't ever have to calculate a matrix-on-matrix product to do this. So, we can do the $O\left(\left(\log\left(\frac{1}{\varepsilon}\right)\right)\right.$ iterations in $O(\text{nnz}(A) + d^2)$ per iteration.

   rips off mask to reveal gradient descent

Time bound: All together, this gives us a time bound of

$$O\left((\text{nnz}(A) + \text{poly}(d)) \cdot \log\left(\frac{1}{\varepsilon}\right)\right)$$

which is pretty good, since it has a log dependendence on $\frac{1}{\varepsilon}$ instead of a polynomial dependence.

Fun Fact 2. This algorithm gives us x_M such that

$$|ARx_M - b|_2^2 \in O(\varepsilon)|ARx^* - b|_2^2$$

where x^* is the optimal solution.

Proof. Let $x^* = \arg\min_x |ARx - b|_2$.

Intuition: Why is it that the x_ms are getting closer to the best solution?

Well, suppose that $ARx_m = b$. Then $R^T A^T(b - ARx_m) = 0$, so our update rule won't change x, and it continues to be a good solution.

Suppose $ARx_m = x^*$. That is, it's not necessarily b, but it's as close to b as we can get in the subspace AR. Then, we know that $b - ARx_m$ is orthogonal to $R^T A^T$. So, like above, $R^T A^T(b - ARx_m) = 0$ and the xs don't change and stay at the good solution.

The gory details: Let's start with x_{m+1}:

$$\begin{aligned} AR(x_{m+1} - x^*) &= AR(x_m + R^T A^T(b - ARx_m) - x^*) \\ &= (AR - ARR^T A^T AR)x_m + AR\left(R^T A^T b\right) - ARx^* \end{aligned}$$

Now, we know that

$$R^T A^T b = R^T A^T ARx^*$$

because of the normal equations. Basically, b is the optimal solution, and x^* is the closest solution in the subspace of AR. But since we are projecting both of them onto $R^T A^T$, the perpendicular difference will go away, and we will get the same thing.

Going back to what we were doing above:

$$\begin{aligned} AR(x_{m+1} - x^*) &= (AR - ARR^T A^T AR)x_m + AR\left(R^T A^T ARx^*\right) - ARx^* \\ &= (AR - ARR^T A^T AR)(x_m - x^*) \end{aligned}$$

Now, we'll do our second-favorite thing[2], and take the SVD of AR:

$$AR = U\Sigma V^T$$

This lets us collapse some stuff, since $V^T V = I$, $U^T U = I$, and $\Sigma = \Sigma^T$:

$$\begin{aligned} AR - ARR^T A^T AR &= U\Sigma V^T - U\Sigma V^T V\Sigma U^T U\Sigma V^T \\ &= U\Sigma V - U\Sigma^3 V^T \\ &= U\left(\Sigma - \Sigma^3\right)V^T \end{aligned}$$

[2] our favorite thing is multiplying by CountSketch

and plugging this into the above equation gives us

$$AR(x_{m+1} - x^*) = U\left(\Sigma - \Sigma^3\right)V^T(x_m - x^*).$$

Now, this is our error, so what we really want is its size.

$$\begin{aligned}
|AR(x_{m+1} - x^*)|_2 &= \left|U\left(\Sigma - \Sigma^3\right)V^T(x_m - x^*)\right|_2 \\
&= \left|\left(\Sigma - \Sigma^3\right)V^T(x_m - x^*)\right|_2 && U \text{ has orthonormal columns}
\end{aligned}$$

Now, remember that Σ comes from the SVD of AR, but we've shown that $\kappa(AR) \leq \frac{1+\varepsilon_0}{1-\varepsilon_0}$. This means that multiplying a vector by $\Sigma - \Sigma^3$ can increase the length of a vector by a factor of at most $O(\varepsilon_0)$.

$$\begin{aligned}
|AR(x_{m+1} - x^*)|_2 &\leq \left|\left(\Sigma - \Sigma^3\right)V^T(x_m - x^*)\right|_2 \\
&\leq O(\varepsilon_0)\left|V^T(x_m - x^*)\right|_2 \\
&\leq \frac{O(\varepsilon_0)}{1-\varepsilon_0}\left|\Sigma V^T(x_m - x^*)\right|_2 \\
&= O(\varepsilon_0)\left|U\Sigma V^T(x_m - x^*)\right|_2 && U \text{ orthonormal columns} \\
&= O(\varepsilon_0)\,|AR(x_m - x^*)|_2
\end{aligned}$$

From this we can get that:

$$\begin{aligned}
|AR(x_m - x^*)|_2 &\leq O(\varepsilon_0)\,|AR(x_{m-1} - x^*)|_2 \\
&\leq O(\varepsilon_0)^2\,|AR(x_{m-2} - x^*)|_2 \\
&\cdots \\
&\leq O(\varepsilon_0)^m\,|AR(x_0 - x^*)|_2 \\
|AR(x_m - x^*)|_2^2 &\leq O(\varepsilon_0)^{2m}\,|AR(x_0 - x^*)|_2^2
\end{aligned}$$

Finally, we use the Pythagorean Theorem to say

$$|ARx_m - b|_2^2 = |AR(x_m - x^*)|_2^2 + |ARx^* - b|_2^2$$

since these two components must be orthogonal (the first part is the component in the subspace of AR, and the second part is the leftovers).

Now we plug in what we have from above to say that

$$|ARx_m - b|_2^2 \leq O(\varepsilon_0)^{2m}\,|AR(x_0 - x^*)|_2^2 + |ARx^* - b|_2^2.$$

We chose the number of iterations $M \in O\left(\log\left(\frac{1}{\varepsilon}\right)\right)$, so we get that $O(\varepsilon_0)^{2M} \in O(\varepsilon)$, since ε_0 is a constant. So we have

$$|ARx_M - b|_2^2 \leq O(\varepsilon)\,|AR(x_0 - x^*)|_2^2 + |ARx^* - b|_2^2.$$

Since our original solution, x_0, was based on a $(1 \pm \varepsilon_0)$-subspace embedding, we know that

$$\begin{aligned}(1+\varepsilon_0)|ARx^* - b|_2^2 &\geq |ARx_0 - b|_2^2 \\ &= |ARx_0 - ARx^*|_2^2 + |ARx^* - b|_2^2 \\ \varepsilon_0|ARx^* - b|_2^2 &\geq |ARx_0 - ARx^*|_2^2\end{aligned}$$

The second line follows from the Pythagorean theorem, since x^* is the closest point in ARs subspace to b.

So we get

$$\begin{aligned}|ARx_M - b|_2^2 &\leq O(\varepsilon)\,|AR(x_0 - x^*)|_2^2 + |ARx^* - b|_2^2 \\ &\leq O(\varepsilon)\cdot\varepsilon_0\,|ARx^* - b|_2^2 + |ARx^* - b|_2^2 \\ &= O(\varepsilon)\,|ARx^* - b|_2^2\end{aligned}$$

which is what we want! ■

2 Leverage Score Sampling

2.1 The Plan

We want a subspace embedding based on sampling. Sampling is nice, because the sketch has the same sparsity as the original matrix.

We saw a naive-ish approach to sampling on the first HW, but that had a run-time dependence on κ^2. Intuitively, this is because we needed to sample enough rows to make sure that we got the important ones (the ones with large norms). Leverage Score Sampling helps us get around this problem.

One interesting difference between this and the approaches that we have seen before, is that the previous sketches that we have looked at did not depend on the matrix A that we are sketching. This sketch is going to be tailored to A.

2.2 What is a leverage score?

We start with the SVD of A:

$$A = U\Sigma V^T$$

We define the ith leverage score of A to be $|U_{i,*}|_2^2$ (where $U_{i,*}$ is the ith row of U). We write it as $\ell(i)$.

Fun Fact 3. $\sum_{v\in\mathsf{rows}(U)} \ell(i) = d$.

Proof. This is the sum of squared entries in the rows of U. Instead lets think of it as the sum of squared entries in the columns of U. Since U is orthonormal, each column has norm 1, so in total this is d. ■

Fun Fact 4. Leverage scores do not depend on the choice of orthonormal basis U for columns of A.

Proof. Let U and U' be two arbitrary orthonormal bases for columns of A.

Since both U and U' have the same column space, we have that

$$U = U'Z$$

for a change of basis matrix Z. That is, Z is a rotation matrix that preserves all lengths and inner products, and has orthonormal rows and columns.

This means

$$|e_i U|_2^2 = |e_i U' Z|_2^2 = |e_i U'|_2^2$$

so all the row norms of U and U' are the same. These are exactly the leverage scores. ■

Can we find these in real life? Well, if we could take the SVD of A, we could find U and calculate the leverage scores exactly, and life would be pretty good. But this is exactly the kind of large computation that we want to avoid.

Instead, our goal is to come up with a distribution $(q_1, q_2, \ldots, q_n)$ such that

$$q_i \geq \beta \cdot \frac{\ell(i)}{d}$$

where β is a parameter. (If β is 1, then this is an exact solution, and as β gets smaller, this approximation gets worse.)

2.3 The Sampling Matrix

We set our sampling matrix, S, to be

$$S = D \cdot \Omega^T.$$

$\Omega \in \mathbb{R}^{n \times k}$ is a sampling matrix and $D \in \mathbb{R}^{k \times k}$ is a rescaling matrix. For each column, j, between 1 and k, we independently and with replacement choose a row i with probability q_i. We set $\Omega_{i,j} = 1$ and $D_{i,j} = \frac{1}{\sqrt{q_i k}}$.

So $SU = D\Omega^T U$ will consist of rows of U scaled by a factor depending on their corresponding leverage score (their lengths, if we are sampling U directly).

2.4 Subspace Embedding

We want to show that for all unit vectors x

$$|SAx|_2^2 = (1 \pm \varepsilon)|Ax|_2^2.$$

We're going to do this similarly to how we analyzed the previous subsampling problems that we have looked at (See Subsampled Hadamard, and the first problem from homework 1).

We can write A in its SVD

$$A = U\Sigma V^T$$

and prove the subspace embedding property for U instead. This will prove it for A as well. That is, if we show that for all unit vectors y

$$|SUy|_2^2 = (1 \pm \varepsilon)|Uy|_2^2$$

then we can set $y = \Sigma V^T x$, and get

$$|SAx|_2^2 = |SU\Sigma V^T x|_2^2 = (1 \pm \varepsilon)|SU\Sigma V^T x|_2^2 = (1 \pm \varepsilon)|Ax|_2^2.$$

So now, we want to use Matrix Chernoff to show that with high probability

$$|U^T S^T SU - I|_2 \leq \varepsilon.$$

(Matrix Chernoff) Let $X_1, \ldots, X_n$ be independent copies of a symmetric random matrix $X \in \mathbb{R}^{n \times n}$ with $\mathbb{E}[X] = 0$, $|X|_2 \leq \gamma$, $|\mathbb{E}[X^T X]|_2 \leq \sigma^2$. Let $W = \frac{1}{k}\sum_{j \in [k]} X_j$. For any $\varepsilon > 0$:

$$\mathbb{P}[|W|_2 > \varepsilon] \leq 2d \cdot e^{-k\varepsilon^2/(\sigma^2 + \frac{\gamma\varepsilon}{3})}$$

So to use this, we need to define some convenient Xs, and show that they meet the conditions.

Let $\mathbf{i}(j)$ denote the row index of U sampled in the jth trial, so $U_{\mathbf{i}(j)}$ is the jth sampled row of U.

We define X_j to be

$$X_j = I_d - \frac{U_{\mathbf{i}(j)}^T U_{\mathbf{i}(j)}}{q_{\mathbf{i}(j)}}.$$

Now let's show that we meet the conditions.

- The Xs are independent copies of a symmetric random variable.

  This is true since we defined them to be independent copies from the same distribution. It's symmetric since I_d and $U_{\mathbf{i}(j)}^T U_{\mathbf{i}(j)}$ are symmetric.

- $\mathbb{E}[X] = 0$.

  First let's look at $\mathbb{E}\left[\frac{U_{\mathbf{i}(j)}^T U_{\mathbf{i}(j)}}{q_{\mathbf{i}(j)}}\right]$.

  $$\begin{aligned}\mathbb{E}\left[\frac{U_{\mathbf{i}(j)}^T U_{\mathbf{i}(j)}}{q_{\mathbf{i}(j)}}\right] &= \sum_i q_i \left(\frac{U_i^T U_i}{q_i}\right) \\ &= U_i^T U_i \\ &= U^T U \\ &= I_d \qquad\qquad U \text{ orthonormal columns}\end{aligned}$$

  Now, we can see

  $$\begin{aligned}\mathbb{E}[X] &= \mathbb{E}[I_d - \frac{U_{\mathbf{i}(j)}^T U_{\mathbf{i}(j)}}{q_{\mathbf{i}(j)}}] \\ &= I_d - \mathbb{E}\left[\frac{U_{\mathbf{i}(j)}^T U_{\mathbf{i}(j)}}{q_{\mathbf{i}(j)}}\right] \\ &= I_d - I_d = 0_d\end{aligned}$$

- $|X|_2 \leq \gamma$. ($\gamma = 1 + \frac{d}{\beta}$.)

$$\begin{aligned} |X|_2 &= \max_{\mathbf{i}(j)} \left| I_d - \frac{U_{\mathbf{i}(j)}^T U_{\mathbf{i}(j)}}{q_{\mathbf{i}(j)}} \right|_2 \\ &\leq \max_{\mathbf{i}(j)} |I_d|_2 + \frac{1}{q_{\mathbf{i}(j)}} \left| U_{\mathbf{i}(j)}^T U_{\mathbf{i}(j)} \right|_2 \qquad \triangle \text{ ineq.} \end{aligned}$$

  Now, let's look at $\left| U_{\mathbf{i}(j)}^T U_{\mathbf{i}(j)} \right|_2$. Remember that we can write

$$U_{\mathbf{i}(j)}^T U_{\mathbf{i}(j)} = \frac{U_{\mathbf{i}(j)}^T}{|U_{\mathbf{i}(j)}|_2} \cdot |U_{\mathbf{i}(j)}|_2^2 \cdot \frac{U_{\mathbf{i}(j)}}{|U_{\mathbf{i}(j)}|_2}$$

  and this is exactly the SVD of the original. This means that we can read our lonely singular value off of the middle "Σ", so

$$\left| U_{\mathbf{i}(j)}^T U_{\mathbf{i}(j)} \right|_2 = |U_{\mathbf{i}(j)}|_2^2.$$

  Plugging back into above, we get

$$\begin{aligned} |X|_2 &\leq 1 + \max_{\mathbf{i}(j)} \frac{1}{q_{\mathbf{i}(j)}} \cdot |U_{\mathbf{i}(j)}|_2^2 \\ &\leq 1 + \frac{d}{\beta} \end{aligned}$$

  the last line follows from the definition of the qs.

  This means that we have proved this for $\gamma = 1 + \frac{d}{\beta}$.

- $\mathbb{E}[X^T X] \leq \sigma^2$. ($\sigma^2 = \left(\frac{d}{\beta} - 1\right)$.)

$$\begin{aligned} \mathbb{E}[X^T X] &= \mathbb{E}\left[\left(I_d - \frac{U_{\mathbf{i}(j)}^T U_{\mathbf{i}(j)}}{q_{\mathbf{i}(j)}} \right)^T \left(I_d - \frac{U_{\mathbf{i}(j)}^T U_{\mathbf{i}(j)}}{q_{\mathbf{i}(j)}} \right) \right] \\ &= I_d - 2\,\mathbb{E}\left[\frac{U_{\mathbf{i}(j)}^T U_{\mathbf{i}(j)}}{q_{\mathbf{i}(j)}} \right] + \mathbb{E}\left[\frac{U_{\mathbf{i}(j)}^T U_{\mathbf{i}(j)} U_{\mathbf{i}(j)}^T U_{\mathbf{i}(j)}}{q_{\mathbf{i}(j)}^2} \right] \\ &= I_d - 2I_d + \mathbb{E}\left[\frac{U_{\mathbf{i}(j)}^T U_{\mathbf{i}(j)}}{q_{\mathbf{i}(j)}^2} \cdot |U_{\mathbf{i}(j)}|_2 \right] \\ &= -I_d + \sum_i q_i \cdot \frac{U_i^T U_i}{q_i^2} \cdot |U_i|_2 \\ &= -I_d + \sum_i U_i^T U_i \cdot \frac{|U_i|_2}{q_i} \\ &\leq -I_d + \frac{d}{\beta} \sum_i U_i^T U_i \qquad \text{(see note below)} \\ &= -I_d + \frac{d}{\beta} U^T U \\ &= \left(\frac{d}{\beta} - 1 \right) I_d \end{aligned}$$

Note: here $A \leq B$ means that $x^T Ax \leq x^T Bx$ for all x.

This means that

$$|\,\mathbb{E}[X^T X]|_2 \leq \left(\frac{d}{\beta} - 1\right)$$

so we can set $\sigma^2 2 = \left(\frac{d}{\beta} - 1\right)$.

Now, let's think about what

$$W = \frac{1}{k} \sum_{j \in [k]} X_j$$

actually means. Remember that the rows of SU are independently chosen rows of U chosen with respect to the probability distribution of the qs and scaled by $\frac{1}{\sqrt{q_i k}}$. This means that

$$\begin{aligned}
(SU)^T SU &= \sum_{j=1}^{k} \left(\frac{1}{\sqrt{q_{\mathbf{i}(j)} k}} \cdot U_{\mathbf{i}(j)}\right)^T \left(\frac{1}{\sqrt{q_i k}} \cdot U_{\mathbf{i}(j)}\right) \\
&= \sum_{j=1}^{k} \frac{1}{q_{\mathbf{i}(j)} k} \cdot U_{\mathbf{i}(j)}^T U_{\mathbf{i}(j)} \\
&= \frac{1}{k} \sum_{j=1}^{k} \frac{U_{\mathbf{i}(j)}^T U_{\mathbf{i}(j)}}{q_{\mathbf{i}(j)}}
\end{aligned}$$

So...

$$\begin{aligned}
W &= \frac{1}{k} \sum_{j \in [k]} X_j \\
&= \frac{1}{k} \sum_{j \in [k]} I_d - \frac{U_{\mathbf{i}(j)}^T U_{\mathbf{i}(j)}}{q_{\mathbf{i}(j)}} \\
&= I_d - \frac{1}{k} \sum_{j=1}^{k} \frac{U_{\mathbf{i}(j)}^T U_{\mathbf{i}(j)}}{q_{\mathbf{i}(j)}} \\
&= I_d - (SU)^T (SU)
\end{aligned}$$

Now, plugging this all into the Matrix Chernoff bound, we get:

$$\begin{aligned}
\mathbb{P}[|W|_2 > \varepsilon] &\leq 2d \cdot e^{-k\varepsilon^2/(\sigma^2 + \frac{\gamma\varepsilon}{3})} \\
\mathbb{P}[|I_d - U^T S^T SU|_2 > \varepsilon] &\leq 2d \cdot e^{-k\varepsilon^2 \Theta(\frac{\beta}{d})}
\end{aligned}$$

So if we set $k \in \Theta\left(\frac{d \log d}{\beta \varepsilon^2}\right)$, then we get that with probability $\geq 1 - e^{-\Theta(1)}$,

$$\begin{aligned}
\varepsilon &\geq |I_d - U^T S^T S U|_2 \\
&\geq |x^T \left(I_d - U^T S^T S U\right) x| && \text{for any unit } x \\
&= |x^T x - x^T U^T S^T S U x| \\
&= |1 - |SUx|_2^2| \\
|SUx|_2^2 &\in (1 \pm \varepsilon) \\
|SUx|_2^2 &\in (1 \pm \varepsilon)|Ux|_2^2
\end{aligned}$$

The last line follows since x is a unit vector and U has orthonormal columns, so $|Ux|_2^2 = 1$.