CS 15-851: Algorithms for Big Data	Spring 2025
Lecture 4 Part 2 — 02/06	
Prof. David Woodruff	*Scribe: Esther Cao*

1 High Precision Regression

We aim to find x' with $|Ax' - b|_2 \leq (1+\epsilon)\min_x |Ax - b|_2$, with high probability. So far, our regression algorithms have had running time nnz(A) + poly(d/ϵ). In high precision settings when ϵ is small, this runtime may be too expensive.

Goal: define an algorithm with a running time of $poly(d) \cdot \log(1/\epsilon)$.

We achieve this with an algorithm that blends sketching with gradient descent.

Definition. The condition of A, $\kappa(A)$, is defined as the ratio between the largest and smallest singular values of A:

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

A is "perfectly conditioned" iff $\kappa(A) = 1$.

The condition number provides a heuristic for how spread out the singular values are. Many algorithms depend on the condition number, and have faster performance when κ is low. The role of sketching in this algorithm is to reduce $\kappa(A)$ to $O(1)$.

1.1 Small QR Decomposition

- Let S be a $(1+\epsilon_0)$ subspace embedding for A. Note that ϵ_0 is not the final ϵ value of the runtime bound; we can think of it as a constant, such as $\epsilon_0 = 1/2$.
- Compute SA. For a CountSketch matrix S, this takes $nnz(A)$ time.
- Compute $SA = QR^{-1}$, the QR-factorization of SA, where Q has orthonormal columns, and R^{-1} is an arbitrary matrix. Note that the QR-factorization can be computed with the thin SVD, letting $Q = U$, and $R^{-1} = \Sigma' V$ where Σ' is the matrix Σ with 0-rows omitted.
- Since Q is orthonormal, the above satisfies $\kappa(SAR) = \kappa(Q) = \sigma_{max}(Q)/\sigma_{min}(Q) = 1$.

Lemma 1. $\kappa(AR) = \frac{1+\epsilon_0}{1-\epsilon_0}$

Proof. Note that $\kappa(SAR) = \kappa(Q) = 1$ since Q is orthonormal.

This can be proven by looking at both sides of the subspace embedding guarantee.

For all unit vectors x:

$$(1-\epsilon_0)|ARx|_2 \leq |SARx|_2 = 1$$

$$(1+\epsilon_0)|ARx|_2 \geq |SARx|_2 = 1$$

The first inequality follows from S is a subspace embedding, and the second equality follows from $SAR = Q$ is orthonormal and thus $SARx$ has length 1.

Therefore we have that for all unit x, $\frac{1}{1+\epsilon_0} \leq |ARx|_2 \leq \frac{1}{1-\epsilon_0}$. Plugging into the definition of κ gives $\kappa(AR) \leq (\frac{1}{1-\epsilon_0})/(\frac{1}{1+\epsilon_0}) = \frac{1+\epsilon_0}{1-\epsilon_0}$. ■

The matrix R is often known as a preconditioner, and can be found by sketching.

Also, note that we don't need to compute the matrix AR (which would take much more than nnz(A) time), since AR would only be used in the context of right-multiplying by a vector which could be calculated efficiently with $A(R(x))$.

1.2 Finding a constant-factor solution

So far, we have found a matrix R such that AR has a small condition number: $\kappa(AR) = \frac{1+\epsilon_0}{1-\epsilon_0}$. Finding R took time $nnz(A) + poly(d)$.

Let S be a $(1+\epsilon_0)$-subspace embedding for AR. (AR has the same column span as A).

Note that $\min_x |Ax - b|_2$ and $\min_x |ARx - b|_2$ are equivalent problems, by a change of variable $y = Rx$.

First, we can find an initial solution x_0 by solving the sketched regression problem $x_0 = argmin_x |SARx - Sb|_2$. The time needed to compute x_0 is $poly(d)$:

1. Compute $SA \in \mathbb{R}^{poly(d)\times d}$ in nnz(A) time
2. compute $(SA)R$, in poly(d) time.
3. Compute Sb in nnz(b)$\leq n$ time
4. This is now a small regression problem, which can be computed by the pseudoinverse in poly(d) time.

Next, we can iteratively improve on the solution with gradient descent. Define

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

We prove the convergence of this value towards the optimal solution x^*.

Claim 1. $|AR(x_{m+1} - x^*)|_2 \leq O(\epsilon_0)^{m+1} |AR(x_0 - x^*)|_2$

$$\begin{aligned}
AR(x_{m+1} - x^*) &= AR(x_m + R^T A^T (b - ARx_m) - x^*) && \text{(By definition of } x_{m+1}\text{)} \\
&= (AR - ARR^T A^T AR)(x_m - x^*) && (R^T A^T b = R^T A^T ARx^* \text{ by Normal equations)} \\
&= (U\Sigma V^T - (U\Sigma V^T)(V\Sigma U^T)(U\Sigma V^T))(x_m - x^*) \\
& && \text{(Substituting SVD of } AR = U\Sigma V^T\text{)} \\
&= U(\Sigma - \Sigma^3)V^T(x_m - x^*) && \text{(Simplifying)}
\end{aligned}$$

We know that AR is well-conditioned (so the singular values of AR are all approximately the same), thus all the diagonal entries of Σ are $1 \pm \epsilon_0$, and all the diagonal entries of Σ^3 are $(1 \pm \epsilon_0)^3 = 1 \pm O(\epsilon_0)$, therefore $|\Sigma - \Sigma^3|_2 = O(\epsilon_0)$.

$$\begin{aligned}
|AR(x_{m+1} - x^*)|_2 &= |U(\Sigma - \Sigma^3)V^T(x_m - x^*)|_2 && \text{(By previous result)} \\
&= |(\Sigma - \Sigma^3)V^T(x_m - x^*)|_2 && \text{(U is orthonormal)} \\
&= O(\epsilon_0)|V^T(x_m - x^*)|_2 && (|\Sigma - \Sigma^3|_2 = O(\epsilon_0)) \\
&\le O(\epsilon_0)/(1 - \epsilon_0)|\Sigma V^T(x_m - x^*)|_2 && \text{(Entries of Σ_0 are in $1 \pm \epsilon_0$)} \\
&= O(\epsilon_0)/(1 - \epsilon_0)|U\Sigma V^T(x_m - x^*)|_2 && \text{(U is orthonormal)} \\
&= O(\epsilon_0)|AR(x_m - x^*)|_2 && \text{(SVD, $AR = U\Sigma V^T$)} \\
&= O(\epsilon_0)^{m+1}|AR(x_0 - x^*)|_2 && \text{(Inductive hypothesis)}
\end{aligned}$$

Therefore, the norm $|AR(x_m - x^*)|_2$ shrinks by $O(\epsilon_0)$ in each iteration.

If we look at the iterate x_m after $m = O(lg(1/\epsilon))$ steps, then $|AR(x_m - x^*)|_2^2 \le O(\epsilon)|AR(x_0 - x^*)|_2^2$.

Finally, we return to comparing ARx

Claim 2. $|ARx_0 - ARx^*|_2^2 \le \epsilon \cdot O(1)|ARx^* - b|_2^2$

Proof:

$$\begin{aligned}
|ARx_0 - ARx^*|_2^2 &\le |ARx_0 - b|_2^2 + |ARx^* - b|_2^2 && \text{(Pythagorean theorem)} \\
&\le \epsilon \cdot O(1)|ARx^* - b|_2^2
\end{aligned}$$

The second inequality follows from:

$$|ARx_0 - b|_2^2 \le (1 + \epsilon_0)|ARx^* - b|_2^2 \qquad \text{(S is subspace embedding)}$$

Claim 3. $|ARx_m - b|_2^2 \le (1 + \epsilon)OPT$

Finally, we can bound $|ARx_m - b|_2$ by the following:

$$\begin{aligned}
|ARx_m - b|_2^2 &= |AR(x_m - x^*)|_2^2 + |ARx^* - b|_2^2 && \text{(Pythagorean theorem)} \\
&\le O(\epsilon)|AR(x_0 - x^*)|_2^2 + |ARx^* - b|_2^2 && \text{(Claim 1, where $m = \lg(1/\epsilon)$)} \\
&\le O(\epsilon)|ARx^* - b|_2^2 + |ARx^* - b|_2^2 && \text{(Claim 2)} \\
&\le (1 + O(\epsilon))OPT && (OPT = |ARx^* - b|_2^2)
\end{aligned}$$

Claim 4. This algorithm has runtime with logarithmic dependence on $1/\epsilon$.

1. $nnz(A) + poly(d)$: compute $R \in \mathbb{R}^{d \times d}$ so that $\kappa(AR) \le (1 + \epsilon_0)/(1 - \epsilon_0)$
2. $nnz(A) + poly(d)$: compute $1 + \epsilon_0$-approximation of the initial value x_0
3. $lg(1/\epsilon) \cdot (d^2 + nnz(A))$: iterative gradient descent for $\log 1/\epsilon$ iterations.

This sums to a total runtime of $nnz(A)lg(1/\epsilon) + poly(d) \cdot lg(1/\epsilon)$, which meets the original goal.

Note: classical gradient descent can be slow because of two problems: it depends on the condition number, and also depends on the initial starting point. Sketching solves both problems, as it enforces the condition number to be 1, and also sets a good starting point x_0.

2 Leverage score sampling

Leverage score sampling provides another subspace embedding, based on sampling the rows of A. This method has the property that if A has sparse rows, then SA has sparse rows where S is the subspace embedding. It aims to sample the rows of A based on importance.

This method will be covered in the next lecture.