CS 15-851: Algorithms for Big Data Spring 2025

Lecture 4 (Part 1) — 02/06/2025

Prof. David Woodruff *Scribe: Renfei Zhou*

1 Missing Proofs from Last Lecture

There are several claims that we did not prove in the last lecture. We present them here.

Claim 1. For matrices A and B, $\|A+B\|_F^2 = \|A\|_F^2 + \|B\|_F^2 + 2\operatorname{Tr}(A^TB)$.

Proof. Let A_i, B_i be the i-th column of A and B, respectively. We have

$$\|A+B\|_F^2 = \sum_i \|A_i+B_i\|_2^2 = \sum_i \Big(\|A_i\|_2^2 + \|B_i\|_2^2 + 2\langle A_i, B_i\rangle\Big) = \|A\|_F^2 + \|B\|_F^2 + 2\operatorname{Tr}(A^TB). \quad \blacksquare$$

Claim 2. For matrices A and B, $\operatorname{Tr}(AB) \le \|A\|_F\|B\|_F$.

Proof. Let $A^{(i)}$ be the i-th row of A and let B_i be the i-th column of B. We have

$$\operatorname{Tr}(AB) = \sum_i \langle A^{(i)}, B_i\rangle \le \sum_i \|A^{(i)}\|_2\|B_i\|_2 \le \left(\sum_i \|A^{(i)}\|_2^2\right)^{1/2}\left(\sum_i \|B_i\|_2^2\right)^{1/2} = \|A\|_F\|B\|_F,$$

where the two inequalities follow from the Cauchy-Schwarz inequality. $\blacksquare$

Claim 3. Let S be the CountSketch matrix with $k = \Theta(1/\varepsilon^2)$ rows and let B^* be an arbitrarily fixed matrix. Then $\|SB^*\|_F^2 = (1\pm\varepsilon)\|B^*\|_F^2$ with constant probability.

Proof. Assume B^* is an $n\times d$ matrix. First, we show $\mathbb{E}[\|SB^*\|_F^2] = \|B^*\|_F^2$ by analyzing each column on both sides independently: We have $\mathbb{E}[\|SB_i^*\|_2^2] = \|B_i^*\|_2^2$ due to the analysis of CountSketch, and taking a summation over all d columns of B^* gives $\mathbb{E}[\|SB^*\|_F^2] = \|B^*\|_F^2$.

Next, we analyze the variance of $\|SB^*\|_F^2$. We use $S^{(i)}$ to denote the i-th row of S, and use B_j^* to denote the j-th column of B^*. We use $h:[n]\to[k]$ to represent the hash function used for CountSketch, so that $h^{-1}(i)$ is the set of columns with non-zero entries in the i-th row of S.

$$\begin{aligned}\operatorname{Var}\Big(\|SB^*\|_F^2\Big) &= \mathbb{E}\Big[\|SB^*\|_F^4\Big] - \|B^*\|_F^4 \\ &= \sum_{(i_1,j_1)}\sum_{(i_2,j_2)} \mathbb{E}\Big[(S^{(i_1)}B_{j_1}^*)^2(S^{(i_2)}B_{j_2}^*)^2\Big] - \|B^*\|_F^4. \end{aligned} \tag{1}$$

We partition all terms in the summation into two groups according to whether $i_1 = i_2$ or not. The first group sums to

$$\begin{aligned}\sum_{i=1}^k \sum_{j_1,j_2\in[d]} \mathbb{E}\Big[(S^{(i)}B_{j_1}^*)^2(S^{(i)}B_{j_2}^*)^2\Big] &= \sum_{i=1}^k \sum_{j_1,j_2\in[d]} \mathbb{E}\left[\left(\sum_{t_1\in h^{-1}(i)}\sigma(t_1)B_{t_1,j_1}^*\right)^2\left(\sum_{t_2\in h^{-1}(i)}\sigma(t_2)B_{t_2,j_2}^*\right)^2\right] \\ &= \sum_{i=1}^k \sum_{j_1,j_2\in[d]} \mathbb{E}\left[\sum_{t_1,t_1',t_2,t_2'\in h^{-1}(i)} \sigma(t_1)\sigma(t_1')\sigma(t_2)\sigma(t_2')\cdot B_{t_1,j_1}^*B_{t_1',j_1}^*B_{t_2,j_2}^*B_{t_2',j_2}^*\right]. \end{aligned} \tag{2}$$

For every term in the large summation indexed by (t_1, t_1', t_2, t_2'), a necessary condition for it to have non-zero contribution to the expectation is that $\{t_1, t_1', t_2, t_2'\}$ form two identical pairs (i.e., no value occurs an odd number of times in $\{t_1, t_1', t_2, t_2'\}$), in which case the leading coefficient $\sigma(t_1)\sigma(t_1')\sigma(t_2)\sigma(t_2')$ always equals 1. There are three cases in which this can happen.

- **Case 1:** $t_1 = t_1' \neq t_2 = t_2'$. These terms sum up to

$$\sum_{i=1}^{k} \sum_{j_1,j_2\in[d]} \mathbb{E}\left[\sum_{t_1\neq t_2\in h^{-1}(i)} (B^*_{t_1,j_1})^2 (B^*_{t_2,j_2})^2\right] \leq \frac{1}{k} \sum_{j_1,j_2\in[d]} \sum_{t_1,t_2\in[n]} (B^*_{t_1,j_1})^2 (B^*_{t_2,j_2})^2 = \frac{1}{k}\|B^*\|_F^4,$$

  where the inequality holds because we only added non-negative terms with $t_1 = t_2$ to the summation.

- **Case 2:** $t_1 = t_2 \neq t_1' = t_2'$. These terms sum up to

$$\begin{aligned}\sum_{i=1}^{k} \sum_{j_1,j_2\in[d]} \mathbb{E}\left[\sum_{t\neq t'\in h^{-1}(i)} B^*_{t,j_1} B^*_{t',j_1} B^*_{t,j_2} B^*_{t',j_2}\right] &\leq \frac{1}{k} \sum_{j_1,j_2\in[d]} \sum_{t,t'\in[n]} B^*_{t,j_1} B^*_{t',j_1} B^*_{t,j_2} B^*_{t',j_2} \\ &= \frac{1}{k} \sum_{j_1,j_2\in[d]} \langle B^*_{j_1}, B^*_{j_2}\rangle^2 \leq \frac{1}{k} \sum_{j_1,j_2\in[d]} \|B^*_{j_1}\|_2^2 \cdot \|B^*_{j_2}\|_2^2 = \frac{1}{k}\|B^*\|_F^4.\end{aligned}$$

  Again, at the first inequality, we added several terms with $t = t'$ which are all non-negative.

- **Case 3:** $t_1 = t_2' \neq t_1' = t_2$. It is the same as Case 2 by renaming the variables (swapping t_2 and t_2').

- **Case 4:** $t_1 = t_2 = t_1' = t_2'$. These terms sum up to

$$\sum_{i=1}^{k} \sum_{j_1,j_2\in[d]} \mathbb{E}\left[\sum_{t\in h^{-1}(i)} (B^*_{t,j_1})^2 (B^*_{t,j_2})^2\right] = \sum_{j_1,j_2\in[d]} \sum_{t\in[n]} (B^*_{t,j_1})^2 (B^*_{t,j_2})^2 = \sum_{t\in[n]} \|(B^*)^{(t)}\|_2^4.$$

The terms in the original summation (1) with $i_1 \neq i_2$ sum up to

$$\begin{aligned}\sum_{i_1\neq i_2\in[k]} \sum_{j_1,j_2\in[d]} \left(\sum_{t_1\in h^{-1}(i_1)} (B^*_{t_1,j_1})^2\right)\left(\sum_{t_2\in h^{-1}(i_2)} (B^*_{t_2,j_2})^2\right) &= \sum_{i_1\neq i_2\in[k]} \sum_{j_1,j_2\in[d]} \sum_{t_1\neq t_2\in[n]} \frac{1}{k^2} (B^*_{t_1,j_1})^2 (B^*_{t_2,j_2})^2 \\ &\leq \sum_{t_1\neq t_2\in[n]} \|(B^*)^{(t_1)}\|_2^2 \cdot \|(B^*)^{(t_2)}\|_2^2 \\ &= \|B^*\|_F^4 - \sum_{t\in[n]} \|(B^*)^{(t)}\|_2^4.\end{aligned}$$

Adding all cases together and substituting back into (1), we know

$$\mathrm{Var}(\|SB^*\|_F^2) \leq \frac{3}{k}\|B^*\|_F^4.$$

By Chebyshev's inequality, we get

$$\Pr[|\|SB^*\|_F^2 - \|B^*\|_F^2| \geq \varepsilon\|B^*\|_F^2] \leq \frac{(3/k)\|B^*\|_F^4}{\varepsilon^2\|B^*\|_F^4} \leq 0.1$$

by setting $k = 30/\varepsilon^2$. ■

2 Low Rank Approximation

Let A be an $n \times d$ matrix, which can be viewed as a set of n points in $\mathbb{R}^d$. In many real-life scenarios, A can be well approximated by a low-rank matrix, but A itself has a high rank due to noise. The goal of low-rank approximation is to find a rank-k matrix A' that approximates A, so that A' takes less space to store, less time to process, and the data in A' are more interpretable than the original matrix A.

If we do not care about the time complexity for finding the low-rank approximation, then SVD gives us the best possible solution. Formally, assume $A = U\Sigma V^T$ is the SVD of A, by keeping only the top-k singular values in Σ and zeroing out other entries (denote the obtained matrix by Σ_k), we get a rank-k matrix $A_k = U\Sigma_k V^T$. This matrix is called the *truncated SVD* of A and it is the best rank-k approximation to A if the distance is measured using the Frobenius norm $\|A - A_k\|_F$.

However, computing the SVD of A will be time consuming, so we want faster algorithms to find the low-rank approximation. Formally, our goal is to find a rank-k matrix A' where $\|A - A'\|_F \leq (1+\varepsilon)\|A - A_k\|_F$, i.e., its distance to A is within $1+\varepsilon$ times the optimal distance. We will introduce an algorithm based on affine embeddings (using CountSketch) that can find A' in time $\operatorname{nnz}(A) + (n+d)\operatorname{poly}(k/\varepsilon)$.

2.1 Warm-Up

In this subsection, we first introduce an algorithm that is slightly slower than the ideal time complexity, but illustrates the main idea of projection.

Overview. Let S be the CountSketch matrix with $r \coloneqq \operatorname{poly}(k/\varepsilon) \ll n$ rows. Then, each row in SA is a random linear combination of the rows in A. $\operatorname{rowspan}(SA)$ is a r-dimensional subspace of $\mathbb{R}^d$. The main idea of the algorithm is to project all n points (rows) in A onto the subspace $\operatorname{rowspan}(SA)$, and then use SVD to find a low-rank matrix A' whose rows lie in $\operatorname{rowspan}(SA)$ that well approximates the n *projected* points.

We begin by showing that there exists a good solution A' that lies in $\operatorname{rowspan}(SA)$ by introducing a thought experiment. We consider a hypothetical regression problem:

$$\min_X \|A_k X - A\|_F. \tag{3}$$

Note that the optimal solution to this regression problem is simply $X = I$, because $A_k X$ has rank $\leq k$, and A_k itself is already the best rank-k approximation of A.

Recall that S is an affine embedding, so $\|SA_k X - SA\|_F = (1 \pm \varepsilon)\|A_k X - A\|_F$ for all matrices X, so solving the sketched regression problem

$$\min_X \|SA_k X - SA\|_F \tag{4}$$

will give a $(1+\varepsilon)$-approximate solution to the original regression problem (3). By normal equations, we know the optimal solution to (4) is $X = (SA_k)^- SA$, thus

$$\|A_k (SA_k)^- SA - A\|_F \leq (1+\varepsilon)\|A_k - A\|_F.$$

It shows that $A_k (SA_k)^- SA$ is an approximation to A such that

- it has rank at most k;
- its rows lie in rowspan(SA); and
- it is $(1+\varepsilon)$-optimal as a rank-k approximation of A.

It indicates that if we only search for low-rank matrices in rowspan(SA), we can still find an $(1+\varepsilon)$-optimal solution to the original low-rank approximation problem. Based on this fact, the algorithm works as follows.

Algorithm. Recall that our goal is to solve

$$\min_{X:\text{rank-}k} \|XSA - A\|_F^2. \tag{5}$$

We rewrite the objective using normal equations:

$$\|XSA - A\|_F^2 = \|XSA - A(SA)^- SA\|_F^2 + \|A(SA)^- SA - A\|_F^2, \tag{6}$$

where the second term on the right-hand side does not depend on X, so we only need to optimize for the first term.

We can compute SA's thin SVD $SA = U\Sigma V^T$, where U and Σ are $r \times r$ matrices and V^T is a $r \times d$ matrix, in $O(d \operatorname{poly} r)$ time. Then we rewrite the objective (first term in (6)) as

$$\min_{X:\text{rank-}k} \|XSA - A(SA)^- SA\|_F^2 = \min_{X:\text{rank-}k} \|XU\Sigma - A(SA)^- U\Sigma\|_F^2 = \min_{Y:\text{rank-}k} \|Y - A(SA)^- U\Sigma\|_F^2.$$

So far, the problem has been reduced to computing the rank-k approximation of $A(SA)^- U\Sigma$, which is a $n \times r$ matrix, and thus computing its SVD only takes $O(n \operatorname{poly} r)$ time. After getting Y, we can output $XSA = YV^T$ as a rank-k approximation to the input matrix A in a factorized form.

We remark that computing the matrix $A(SA)^- U\Sigma$ is the time bottleneck of the algorithm, which takes $\operatorname{nnz}(A) \cdot r + d \operatorname{poly} r$ time, which is unacceptable. This challenge is similar to directly projecting all n points into the subspace rowspan(SA), which requires us to compute $A(SA)^- SA = A(SA)^- U\Sigma V^T$ (the only difference is the factor V^T on the right). The merit of this algorithm is that all SVD steps are already efficient, as they are applied only to $r \times d$ and $n \times r$ matrices, which takes $O((n+d) \operatorname{poly} r)$ time.

2.2 Efficient Algorithm

The only bottleneck of the previous algorithm is to compute the projection of n points into the subspace rowspan(SA). Although the precise projections are hard to compute, we can view projections as regression problems, and use an affine embedding to get approximate projections.

We start with the the optimization problem (5), and apply an affine embedding R that satisfies

$$\|XSAR - AR\|_F^2 = (1 \pm \varepsilon)\|XSA - A\|_F^2$$

for all matrices X. Since SA is an $r \times d$ matrix, the number of columns in the affine embedding R will be $r' \coloneqq \operatorname{poly}(r/\varepsilon) = \operatorname{poly}(k/\varepsilon)$. After applying R, (5) reduces to solving

$$\min_{X:\text{rank-}k} \|XSAR - AR\|_F^2.$$

Using normal equations, this equals

$$\|AR(SAR)^{-}(SAR) - AR\|_F^2 + \min_{X:\text{rank-}k} \|XSAR - AR(SAR)^{-}(SAR)\|_F^2.$$

The second term can be rewritten as

$$\min_{Y:\text{rank-}k} \|Y - AR(SAR)^{-}(SAR)\|_F^2. \tag{7}$$

We remark that the optimal solution Y for (7) can always be represented as $Y = XSAR$, i.e., it lies in the row space of SAR, since otherwise projecting Y into rowspan(SAR) gives a better solution to (7).

We can compute the optimal Y by computing the SVD of $AR(SAR)^{-}(SAR)$, which takes $n \operatorname{poly} r' = n \operatorname{poly}(k/\varepsilon)$ time. Note that computing the matrix $AR(SAR)^{-}(SAR)$ itself is also efficient, taking $\operatorname{nnz}(A) + n \operatorname{poly}(k/\varepsilon)$ time. Finally, we output $Y(SAR)^{-}SA = XSA$ as the low-rank approximation to the input matrix A.