Problem Set 2 Solutions

October 18, 2019

1 Problem 1

In lecture #3, we saw that any matrix \( S \in \mathbb{R}^{r \times n} \) which satisfies the following three properties is good enough for an affine embedding with probability 2/3.

1. For a fixed matrix \( B \), we have \( \|SB\|_F = (1 \pm \epsilon)\|B\|_F \) with probability \( 99/100 \).

2. For matrices \( C, D \), we have approximate matrix product:
\[
\text{Pr}[\|CS^TSD - CD\|_F \geq \epsilon\|C\|_F\|D\|_F] > 99/100
\]

3. For a fixed matrix \( A \in \mathbb{R}^{n \times d} \), \( S \) is an \( \epsilon \)-subspace embedding for \( A \) with probability \( 99/100 \): for all \( x \in \mathbb{R}^d \):
\[
\|SAx\|^2_2 = (1 \pm \epsilon)\|Ax\|^2_2
\]

By the problem statement, 1), holds with \( \Omega(1/\epsilon^2) \) rows of \( s \). From Lecture 2, properties 2 and therefore 3 hold with \( \Omega(d^2/\epsilon^2) \) rows of \( s \). So \( s = \Omega(d^2/\epsilon^2) \).

1.1 1.2

First generate a count-sketch \( S \in \mathbb{R}^{r \times n} \), where \( r = \text{poly}(k/\epsilon) \). Observe that we can compute \( SA \) in \( 	ext{nnz}(A) \) time, and now \( SA \in \mathbb{R}^{r \times n} \), so we can compute \((SA)B\) in \( \text{nnz}(B) \cdot r \) time, and finally we can compute \((SAB)C\) in \( \text{nnz}(C) \cdot r \) time. Thus we compute \( SABC \) in at most \( (\text{nnz}(A) + \text{nnz}(B) + \text{nnz}(C))\text{poly}(k/\epsilon) \) time. As seen in class, we now know that there exists a good (meaning \( (1 + \epsilon) \) approx.) rank-\( k \) approximation in the row span of \( SABC \). The remaining steps are nearly identical to the steps taken in class for LRA. So we now want to approximately project the rows of \( ABC \) onto the row span of \( SABC \). Note that the optimal projection is given by \( XSABC \) where \( X \) is the optimizer to \( \min_{\text{rank}-k X} \|XSABC - ABC\|_F \), thus our observation can be summarized by:
\[
\min_{\text{rank}-k X} \|XSABC - ABC\|_F \leq (1 + \epsilon)\|ABC\|_F - ABC\|_F
\]

This projection is too costly to compute, so instead we generate an affine embedding (say count-sketch) \( R \in \mathbb{R}^{d \times r} \) for \( SABC \). Note that we only need \( r = \text{poly}(k/\epsilon) \), because \( S \) only has \( \text{poly}(k/\epsilon) \) rows. We then compute \( (SABC)R \) in \( \text{poly}(k/\epsilon) \) time, and also compute and \( A\overline{BCR} \) and \( (SABC)R^\top \). Note that since \( R \) is count-sketch, by the same argument as above, we can compute \( A\overline{BCR} \) in \( \text{nnz}(A) + \text{nnz}(B) + \text{nnz}(C) \) time. Now we just solve
\[
\min_{\text{rank}-k Y} \|A\overline{BCR}SABC - SABC - Y\|_F^2
\]
But now \( A\overline{BCR}SABC \) is size \( n \times \text{poly}(k/\epsilon) \), so we can solve for the optimizer \( Y \) above in \( O(n\text{poly}(k/\epsilon)) \) time. We can then output \( Y(SABC)R \) in factored form.
2 Problem 2

Fix some $\epsilon_0 = \Theta(1)$. Let $S \in \mathbb{R}^{r \times n}$ be a count-sketch with $r = \Theta(d^2/\epsilon_0^2) = \Theta(d^2)$ columns. Then $S$ is a $(1 + \epsilon_0)$ SE for $A \in \mathbb{R}^{r \times d}$ with probability $99/100$. Let $S_i$ be the $i$-th row of $S$, and let $\Omega_i = \{j \in [n] \mid S_{i,j} \neq 0\}$, i.e. $\Omega_i$ is the set of non-zero entries in the $i$-th row of $S$. Then the $i$-th row of $SA$ is just $S_iA_{\Omega_i}$ where $A_{\Omega_i}$ is the $|\Omega_i| \times d$ submatrix of $A$ with rows in $\Omega_i$. By assumption, we can compute $S_iA_{\Omega_i}$ in time $T(\tfrac{|\Omega_i|d}{nd})$, thus we can compute $SA$ in time

$$\sum_{i=1}^{r} T(\tfrac{|\Omega_i|d}{nd}) = T$$

which follows because the $\Omega_i$’s partition $[n]$. We then compute the QR decomposition $SA = QR^{-1}$ in time $\text{poly}(d)$, and compute $R \in \mathbb{R}^{d \times d}$ in time $\text{poly}(d)$. As seen in class: $\kappa(A) \leq (1 + \epsilon_0)/(1 - \epsilon_0)$, since $(1 - \epsilon_0)\|ARx\|_2 \leq \|SARX\|_2 = 1$, and $(1 + \epsilon_0)\|ARx\|_2 \geq \|SARX\|_2 = 1$. So now we know that $AR$ is $O(1)$-well conditioned. Given $SA$, we can compute $SAR$ and $Sb$ in $n + \text{poly}(d)$ time, and solve

$$x_0 = \arg \min_x \|SARx - Sb\|_2$$

in $\text{poly}(d)$ time. Then we know that $x_0$ is a $(1 + \epsilon_0)$ optimal solution to $\min_x \|ARx - b\|_2$. We now apply gradient descent, setting $x_i \leftarrow x_{i-1} + RT(\sigma - ARx_{i-1})$. As argued in lecture 4 (see slides for proof), if $x^* = \arg \min_x \|ARx - b\|_2$, after each step we have $\|AR(x_{i+1} - x^*)\|_2 = O(\epsilon_0)\|AR(x_i - x^*)\|_2$. Moreover, $\|ARx_i - b\|_2 = \|AR(x_i - x^*)\|_2^2 + \|ARx^* - b\|_2$. Now note that $\|ARx^* - b\|_2 = \min_x \|Ax - b\|_2$, since $AR$ and $A$ have the same column span. Thus after $t$ steps, if $\epsilon_0 < 1/2$, we have

$$\|ARx_i - b\|_2 \leq 2^{-t} \|AR(x_0 - x^*)\|_2^2 + OPT$$

Now since $x_0$ was a constant factor solution, we have $\|ARx - b\| \leq O(1) \cdot OPT$, thus setting $t = O(\log(1/\epsilon))$, we have $\|ARx_i - b\|_2 \leq \epsilon OPT + OPT$ as desired. For the runtime of this portion, note that compute $ARx_{i-1}$ can be done by first computing $Rx_{i-1}$ in $\text{poly}(d)$. Then $A(Rx_{i-1})$ and $A^T(b - ARx_i)$ can be computed in $O(T)$ time as seen above, and the last matrix vector product is $\text{poly}(d)$ time.

3 Problem 3

3.1 3.1

Let $V^T \in \mathbb{R}^{d \times d}$ be a basis for the row span of $A$ with orthonormal rows. Since $A$ is full rank, it follows that $V^T$ is full rank, and thus all the singular values of $V^T$ are 1, so $V^T$ must also have orthonormal columns. Thus $\ell_i(A) = 1$ for $i = 1, 2, \ldots, d$

3.2 3.2

Let $U$ be an orthonormal basis for the row span of $A$, and $W$ an orthonormal basis for the row span of $B'$. If the rank of $A$ and $B$ are the same, then the result is clear. Otherwise, we have $\ell_i(A) = a_i^T(AA^T)^{-1}a_i$, and $\ell_i(B) = b_i^T(BB^T)^{-1}b_i$, where $a_i, b_i$ are the $i$-th column of $A, B$. Note by construction that $a_i = b_i$ for $i = 1, 2, \ldots, d$. Now note that for any $x \in \mathbb{R}^n$, $\|x^T B\|_2^2 = \|x^T A\|_2^2 + \langle x, a_{d+1} \rangle \geq \|x^T A\|_2^2$. Thus $x^T BB^T x \geq x^T AA^T x$ for all $x \in \mathbb{R}^n$, so $BB^T \succ AA^T$ where $\succ$ is the PSD ordering, and therefore $(BB^T)^{-} \prec (AA^T)^{-}$. So $x^T(BB^T)^{-} x \leq x^T(AA^T)^{-} x$. Thus $\ell_i(A) = a_i^T(AA^T)^{-}a_i \geq a_i^T(BB^T)^{-}a_i \ell_i(B)$ for $i \in [d]$ as needed.
\section*{3.3}

As the problem suggests, we consider \( D = [A, \sqrt{\lambda}I_n] \) and sample according to \( \ell_i(D) = \ell_i(D) \) for \( i = 1, 2, \ldots, d, \) and \( \ell_i(D) = 1 \) for \( i > d. \) First note that \( \ell_i(D) = \tau_i(A) \) for \( i = 1, 2, \ldots, d. \) To see this, first note via block-matrix multiplication that \( D D^T = [A, \sqrt{\lambda}I]\lambda [A, \sqrt{\lambda}I]^T = AA^T + \lambda I. \) Using the alternative definition of leverage scores, we have \( \ell_i(D) = a_i^T (DD)^{-1} a_i = a_i^T (AA^T + \lambda I)^{-1} a_i = \tau_i(A). \) Secondly, note that \( \ell_i(D) \leq 1 \) for all \( i, \) since leverage scores are upper bounded by \( 1, \) from which it follows that \( \ell_i(\ell_i(D)) \) for \( i > d. \)

In class, we saw that for a \( n \times d \) matrix \( A \) with \( d > n, \) if we sample \( O(n \log(n)/\epsilon^2) \) columns \( C = DS \) of \( D \) according to probabilities \( \ell_i > \beta \ell_i(D), \) we will obtain \( (1 - \epsilon) CC^T \preceq DD^T \preceq (1 + \epsilon) CC^T \) with probability \( 9/10, \) giving

\[
(1 - \epsilon) CC^T \preceq AA^T + \lambda I_n \leq (1 + \epsilon) CC^T
\]

Observe that our distribution \( \ell_i \) satisfies this property with \( \beta = 1. \) So we sample according to \( \ell_i, \) and obtain the above result with probability \( 9/10. \) We can write \( C = DS = [AS_1, \sqrt{\lambda}IS_2], \) where \( S_1, S_2 \) are column sampling matrices. Now note that the probability that a given sample is from a column in \( A \) is at most \( \frac{\sum_{i < d} \ell_i(D)}{\sum_{i \leq d} \ell_i(D)} = \frac{\sum_{i < d} \tau_i(A)}{\sum_{i \leq d} \ell_i(D)} \leq \frac{2k}{n} \) (here we use the fact that sum of all the leverage scores is at most \( n \)). Thus, taking \( O(n \log(n)/\epsilon^2) \) samples, the expected number of columns of \( AS_1 \) is \( O(k \log(n)/\epsilon^2), \) and is \( O(k \log(n)/\epsilon^2) \) with probability \( 1 - 1/n \) by a Chernoff bound.

Now note \( CC^T = (AS_1)(AS_1)^T + \lambda(I_n S_2)(I_n S_2)^T, \) so it will suffice to show that

\[
\| (I_n S_2)(I_n S_2)^T - I_n \|_2 \leq \epsilon \tag{1}
\]

But note that 1) the leverage scores for \( I_n \) are uniform and 2) \( S_2 \) is uniformly sampling \( O(n \log(n)/\epsilon^2) \) columns. Thus the same matrix Chernoff bound for leverage score sampling applies, and we obtain right away. Given this, we have \( \lambda(I_n S_2)(I_n S_2)^T (1 - \epsilon) \preceq I_n \preceq \lambda(I_n S_2)(I_n S_2)^T (1 + \epsilon), \) so

\[
(1 - \epsilon)(AS_1)(AS_1)^2 - \epsilon I_n \preceq AA^T \preceq (1 + \epsilon)(AS_1)(AS_1)^2 + \epsilon I_n
\]
as needed.