15-851 Algorithms for Big Data — Spring 2024

Problem Set 2

Due: Thursday, February 29, before class

Please see the following link for collaboration and other homework policies:
`http://www.cs.cmu.edu/afs/cs/user/dwoodruf/www/teaching/15851-spring24/grading.pdf`

Problem 1: Matrix Coherence (16 points)
The coherence $\mu(A)$ of an $n \times d$ matrix A is defined to be $\max_{i=1,\ldots,n} \ell_i(A)$, where $\ell_i(A)$ is the i-th leverage score of A.

1. Given an $n \times d$ matrix A with $n \geq d$, prove that

$$\mu(A) = \sup_x \frac{\|Ax\|_\infty^2}{\|Ax\|_2^2},$$

   where recall that for a vector $y \in \mathbb{R}^n$, we have $\|y\|_\infty = \max_{i=1}^n |y_i|$.

2. For $n \geq d$, argue that for any $n \times d$ matrix A with $\text{rank}(A) = d$, that $\mu(A) \geq d/n$. Give an example $n \times d$ matrix A where $\mu(A) = d/n$. Your example should work for any d and n with $n \geq d$ and n is an integer multiple of d.

3. Given an $n \times d$ matrix A and an $n \times d$ matrix B, let $C = [A, B]$, that is, we adjoin the columns of B to those of A so that C is an $n \times 2d$ matrix. Assume n is strictly larger than $2d$ and C is not equal to the all zeros matrix. Give an example pair of matrices A and B for which $\mu(C) = 1$ but $\mu(A) = d/n$ and $\mu(B) = d/n$.

4. You are again given A and B as in the previous part, but now you are promised that the columns of A and B are orthonormal, and that any column of A is perpendicular to any column of B. Let $C = [A, B]$ again, so that C is an $n \times 2d$ matrix with $n > 2d$ and C not equal to the all zeros matrix. Prove that

$$\max(\mu(A), \mu(B)) \leq \mu(C) \leq \mu(A) + \mu(B).$$

Problem 2: Low Rank Preconditioning (17 points)
In this problem you are given an $n \times n$ matrix A with $\text{rank}(A) = n$, and would like to compute an $n \times n$ matrix R so that $\kappa(AR) = O(1)$, where $\kappa(AR) = \frac{\sigma_1(AR)}{\sigma_n(AR)}$ is the condition number of AR, and $\sigma_i(AR)$ is the i-th singular value of AR. We will assume[1] that $\sigma_n(A) = 1$. As we saw in class, this is useful for ensuring the number of iterations of gradient descent for least squares regression is small. Here though, A is a square matrix and so the computation in class would be fairly slow. In this problem we will show how to use a low rank approximation to precondition square matrices provided they satisfy certain properties.

1. Suppose $\sigma_{k+1}(A) = O(\sigma_n(A))$, but the top k singular values of A may be arbitrarily large. Let A_k be the best rank-k approximation to A, i.e., the $n \times n$ matrix A for which $\|A - A_k\|_F$ is minimized. We can write $A_k = U_k \Sigma_k V_k^T$ where $U_k, V_k \in \mathbb{R}^{n \times k}$, and $\Sigma_k \in \mathbb{R}^{k \times k}$. Argue that the $n \times n$ matrix $R = I - V_k V_k^T + V_k \Sigma_k^{-1} V_k^T$ is a good preconditioner, that is, for all x we have
$$\|ARx\|_2 = \Theta(1)\|x\|_2.$$
HINT: It may help to write A in its SVD. Also note that the column spans of $A - A_k$ and A_k are orthogonal.

2. Computing A_k exactly is expensive, so let us see how we can get a faster algorithm. To simplify the problem, we will restrict to $k = 1$ in what follows. Suppose:
   - We have a unit vector z for which $\langle v_1, z \rangle \geq 1 - 1/(10\sigma_1(A)^2)$, where v_1 is the top right singular vector of A.
   - We have a number λ for which $\lambda = \sigma_1(A) \pm 1/10$.
   - $\sigma_2(A) = O(\sigma_n(A))$.

   Show that the $n \times n$ matrix $R = I - zz^T + z\frac{1}{\lambda_1}z^T$ is a good preconditioner, that is, for all x we have
$$\|ARx\|_2 = \Theta(1)\|x\|_2.$$
   HINT: use the properties of z and λ above and try to directly compute this quantity. It may help to write A in its SVD and write $x = \alpha z + \beta w$ where $\langle z, w \rangle = 0$.

   It is known how to find a unit vector v_1 and corresponding value λ efficiently, assuming $\sigma_1 > 2\sigma_2$. The running time is $O(\text{nnz}(A) \log n)$. The idea is to use the power method, which you may have seen in an earlier class. For those who are interested, we encourage you to read the solutions to problem 2 on problem set 2 here: `http://www.cs.cmu.edu/~dwoodruf/teaching/15859-fall22/index.html` to get an idea how to find such a z and v_1.

3. In class, each iteration of gradient descent involved multiplying by R and then by A, giving time $O(\text{nnz}(A) + n^2)$ in the case $n = d$. What is the per iteration time complexity of using the preconditioner you found in the previous part?

[1] It is not hard to reduce to this case but we will not discuss it here.

Problem 3: Tensor Low Rank Approximation (17 points)

Given q vectors $u_1 \in \mathbb{R}^{n_1}$, $u_2 \in \mathbb{R}^{n_2}$, $\cdots$, $u_q \in \mathbb{R}^{n_q}$, we use $u_1 \otimes u_2 \otimes \cdots \otimes u_q$ to denote an $n_1 \times n_2 \times \cdots \times n_q$ tensor such that, for each $(j_1, j_2, \cdots, j_q) \in [n_1] \times [n_2] \times \cdots \times [n_q]$,

$$(u_1 \otimes u_2 \otimes \cdots \otimes u_q)_{j_1, j_2, \cdots, j_q} = (u_1)_{j_1} (u_2)_{j_2} \cdots (u_q)_{j_q},$$

where $(u_i)_{j_i}$ denotes the j_i-th entry of vector u_i.

An $n \times n \times n$ tensor A is a 3-dimensional array where $A_{i,j,k} \in \mathbb{R}$. The *rank* of a tensor is the minimal integer k for which $A = \sum_{i=1}^{k} u^i \otimes v^i \otimes w^i$ for vectors u^i, v^i, w^i, each in $\mathbb{R}^n$, for each $i = 1, \ldots, k$. For a tensor A, we can define $\|A\|_F = \sqrt{\sum_{i,j,k} A_{i,j,k}^2}$ to be the natural

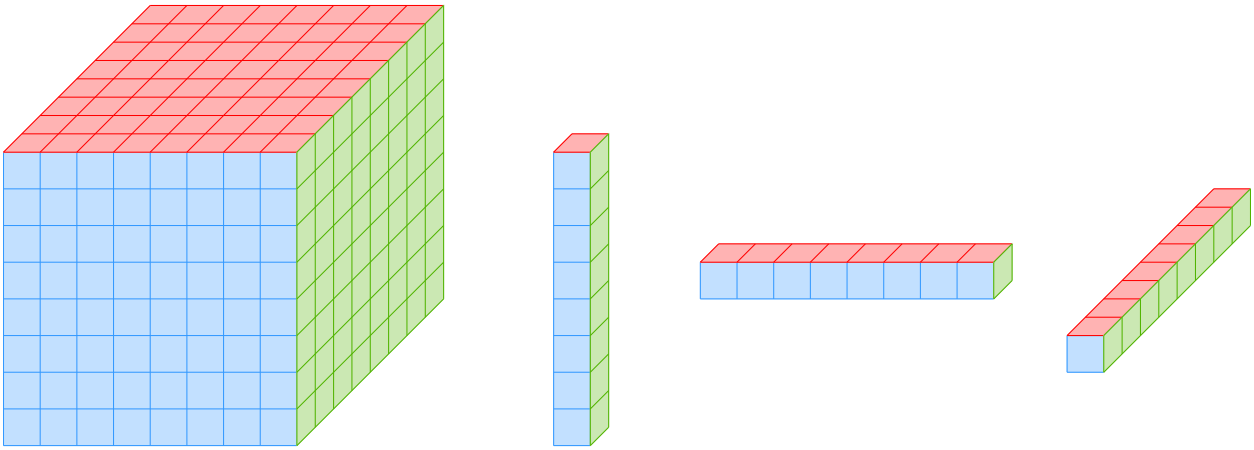

Figure 1: A 3rd order tensor contains n^2 columns, n^2 rows, and n^2 tubes.

analogue of the Frobenius norm for matrices. In this problem, we would like to try to find a rank-k tensor B for which

$$\|A - B\|_F^2 \leq (1 + \epsilon)\|A - A^*\|_F^2, \tag{1}$$

were A^* is a rank-k tensor which minimizes[2] $\|A - B\|_F^2$ over all rank-k tensors B. We can write $A^* = \sum_{i=1}^{k} u^{*,i} \otimes v^{*,i} \otimes w^{*,i}$. Define U^* to be the $n \times k$ matrix with columns $u^{*,i}$, for $i = 1, \ldots, k$. We similarly define V^* and W^*.

A useful concept is the notion of a *flattening* of a tensor: there are three flattenings denoted A^1, A^2, and A^3, each in $\mathbb{R}^{n \times n^2}$. For $(c, d) \in \{1, 2, \ldots, n\} \times \{1, 2, \ldots, n^2\}$, write $d = d_1 n + d_0$, where $d_0, d_1 \in \{1, 2, \ldots, n\}$. Define $A^1 \in \mathbb{R}^{n \times n^2}$ to be the matrix whose (c, d)-th entry is A_{c,d_0,d_1}. Define $A^2 \in \mathbb{R}^{n \times n^2}$ to be the matrix whose (c, d)-th entry is A_{d_0,c,d_1}. Finally, define $A^3 \in \mathbb{R}^{n \times n^3}$ to be the matrix whose (c, d)-th entry is $A_{d_0,d_1,c}$. See Figure 2 for an example.

[2]There is a subtle issue in showing such an A^* exists. You can take this as given for this problem.

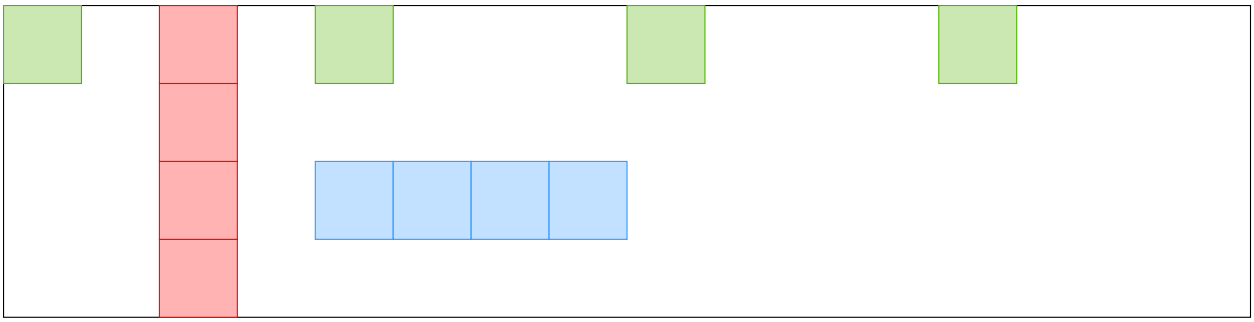

Figure 2: Flattening. We flatten a third order $4 \times 4 \times 4$ tensor along the 1st dimension to obtain a 4×16 matrix. The red blocks correspond to a column in the original third order tensor, the blue blocks correspond to a row in the original third order tensor, and the green blocks correspond to a tube in the original third order tensor.

With this notation, we have that the three flattenings of A_k^* are (1) $U^* \cdot ((V^*)^T \odot (W^*)^T)$, (2) $V^* \cdot ((U^*)^T \odot (W^*)^T)$, and (3) $W^* \cdot ((U^*)^T \odot (V^*)^T)$, where for two $k \times n$ matrices E and F, we define $E \odot F$ to be the $k \times n^2$ matrix obtained whose (c, d)-th entry is $E_{c,d_0} \times F_{c,d_1}$.

Let $(R^1)^T, (R^2)^T, (R^3)^T$ be three independent CountSketch matrices, each with $\text{poly}(k/\epsilon)$ columns. Prove that with probability at least $9/10$, there exists a rank-k tensor $B = \sum_{i=1}^k u^i \otimes v^i \otimes w^i$ which satisfies (1) and for which (1) $u^1, \ldots, u^k$ are in the column span of $A^1 R^1$, (2) $v^1, \ldots, v^k$ are in the column span of $A^2 R^2$, and (3) $w^1, \ldots, w^k$ are in the column span of $A^3 R^3$.

HINT: Try to generalize the argument we used in class for showing that for a matrix A, the row span of SA contains a $(1+\epsilon)$-approximate rank-k space to the matrix low rank approximation problem for A. You will want to fix the optimum A^* for this argument and consider a flattening of it. After doing this, you may need to apply the argument from class twice more, where each application may depend on the conclusion from a previous application and replace a certain part of A^* with what you get from the previous part.