Large Scale Matrix Computation, Optimization and Learning (EE270) HW #1

Winter 2021 Due: **Friday, January 29, 23:59pm**

- After a genuine attempt to solve the homework problems by yourself, you are free to collaborate with your fellow students to find solutions to the homework problems. Regardless of whether you collaborate with other students, you are required to type up or write your own solutions. Copying homework solutions from another student or from existing solutions is a serious violation of the honor code. Please take advantage of the professor's and TA's office hours. We are here to help you learn, and it never hurts to ask!
- **The assignments should be submitted via Gradescope including your code attached as pdf**

1. **Linear Algebra (15 pts)**

   Are the following statements **true** or **false**? If true, prove it; if false, show a counterexample.

   (a) The inverse of a symmetric matrix is itself symmetric.

   (b) All 2×2 orthogonal matrices have the following form

   $$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \text{ or } \begin{bmatrix} \cos\theta & \sin\theta \\ \sin\theta & -\cos\theta \end{bmatrix}.$$

   (c) Let $A = \begin{bmatrix} -8 & -1 & -6 \\ -3 & -5 & -7 \\ -4 & -9 & -2 \end{bmatrix}$. A can be written as $A = CC^T$ for some matrix C.

2. **Divide and Conquer Matrix Multiplication (15 pts)**

   If A is a matrix, then $A^2 = AA$ is the square of A.

   (a) Show that five multiplications are sufficient to compute the square of a 2×2 matrix $A = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix}$, where a_1, a_2, a_3, a_4 are scalars.

   (b) Generalize the formula in part (a) to a 2×2 block matrix $A = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix}$ where A_1, A_2, A_3, A_4 are arbitrary matrices.

   (c) Instead of using the classical matrix multiplication (three-loop) algorithm for computing A^2, we may apply the block formula you derived in (b) to reduce $2n \times 2n$ problems to several $n \times n$ computations, which can be tackled with classical matrix multiplication. Compare the total number of arithmetic operations. Generate $2n \times 2n$ random A matrices and plot the wall-clock time of the classical matrix multiplication algorithm and the algorithm using the formula in (b) to compute A^2 for $n = 4, ..., 10000$ (or as large as your system memory allows). You can use standard packages for matrix multiplication, e.g., numpy.matmul.

(d) Show that if you have an algorithm for squaring an $n \times n$ matrix in $O(n^c)$ time, then you can use it to multiply any two arbitrary $n \times n$ matrices in $O(n^c)$ time. [Hint: Consider multiplying two matrices A and B. Can you define a matrix whose square contains AB?]

3. **Probability (30 pts)**

(a) Random variables X and Y have a joint distribution $p(x, y)$. Prove the following results. You can assume continuous distributions for simplicity.

i. $\mathbb{E}[X] = \mathbb{E}_Y[\mathbb{E}_X[X|Y]]$

ii. $\mathbb{E}[I[X \in \mathcal{C}]] = P(X \in \mathcal{C})$, where $I[X \in \mathcal{C}]$ is the indicator function[1] of an arbitrary set $\mathcal{C}$.

iii. $\text{var[X]} = \mathbb{E}_Y[\text{var}_X[X|Y]] + \text{var}_Y[\mathbb{E}_X[X|Y]]$

iv. If X and Y are independent, then $\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$.

v. If X and Y take values in $\{0, 1\}$ and $\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$, then X and Y are independent.

(b) Show that the approximate randomized counting algorithm described in Lemma 1 of Lecture 2 slides (page 14) is unbiased:

$$\mathbb{E}\tilde{n} = n\,. \tag{1}$$

(c) Prove the variance formula in Lemma 2 of Lecture 2 slides (page 38) for Approximate Matrix Multiplication $AB \approx CR$

$$\mathbf{Var}\left[(CR)_{ij}\right] = \frac{1}{m}\sum_{k=1}^{d}\frac{A_{ik}^2 B_{kj}^2}{p_k} - \frac{1}{m}(AB)_{ij}^2\,. \tag{2}$$

where $\{p_k\}_{k=1}^d$ are sampling probabilities.

4. **Positive (Semi-)Definite Matrices (15 pts)**

Let A be a real, symmetric $d \times d$ matrix. We say A is *positive semi-definite* (PSD) if, for all $\mathbf{x} \in \mathbb{R}^d$, $\mathbf{x}^\top A\mathbf{x} \geq 0$. We say A is *positive definite* (PD) if, for all $\mathbf{x} \neq 0$, $\mathbf{x}^\top A\mathbf{x} > 0$. We write $A \succeq 0$ when A is PSD, and $A \succ 0$ when A is PD.

The *spectral theorem* says that every real symmetric matrix A can be expressed $A = U\Lambda U^\top$, where U is a $d \times d$ matrix such that $UU^\top = U^\top U = I$ (called an orthogonal matrix), and $\Lambda = diag(\lambda_1, ..., \lambda_d)$. Multiplying on the right by U we see that $AU = U\Lambda$. If we let u_i denote the i^{th} column of U, we have $Au_i = \lambda_i u_i$ for each i. This expression reveals that the λ_i are eigenvalues of A, and the corresponding columns u_i are eigenvectors associated to λ_i.

(a) A is PSD iff $\lambda_i \geq 0$ for each i.

(b) A is PD iff $\lambda_i > 0$ for each i.

Hint: Use the following representation

$$U\Lambda U^T = \sum_{i=1}^{d}\lambda_i u_i {u_i}^T\,.$$

[1] $I[X \in \mathcal{C}] := 1$ if $X \in \mathcal{C}$ and 0 otherwise

5. **Norms (20 pts)**

(a) For $p = 1, 2, \infty$, verify that the functions $\|\cdot\|_p$ are norms. Then, for a vector $x \in \mathbb{R}^n$, show that

$$\|x\|_\infty \leq \|x\|_2 \leq \|x\|_1 \leq \sqrt{n}\|x\|_2 \leq n\|x\|_\infty$$

and for each inequlaity, provide an example demonstrating that the inequality can be tight.

(b) For vectors x,$y \in \mathbb{R}^n$, show that $|x^T y| \leq \|x\|_2\|y\|_2$ with equality if and only if x and y are linearly dependent. More generally, show that $x^T y \leq \|x\|_1\|y\|_\infty$. Note that this implies that $\|x\|_2^2 \leq \|x\|_1\|x\|_\infty$; and that these are special cases of Hölder's inequality.

(c) For $A \in \mathbb{R}^{m\times n}$, show that $\text{Trace}(A^T A) = \sum_{ij} A_{ij}^2$, and show that $\sqrt{\sum_{ij} A_{ij}^2}$ is a norm on $m \times n$ matrices. This is the Frobenius norm, denoted $\|\cdot\|_F$. Show that, in addition to satisfying the definining properties of a norm, the Frobenius norm is a submultiplicative norm, in that

$$\|AB\|_F \leq \|A\|_F\|B\|_F$$

whenever the dimensions are such that the product AB is defined.

(d) Recall the definiton of the spectral norm of an $m \times n$ matrix A : $\|A\|_2 = \sqrt{\lambda_{max}(A^T A)} = \sigma_{max}(A)$, where $\lambda_{max}(A^T A)$ is the largest eigenvalue pf $A^T A$ and σ_{max} is the largest singular value of A. Show that the Frobenius norm and the spectral norm are unitarily invariant: if U and V are unitary (orthogonal in the real case) matrices, then $\|U^T AV\|_\xi = \|A\|_\xi$, for $\xi = 2, F$.

6. **Approximate Matrix Multiplication (20 pts)**

Here, we will consider the empirical performance of random sampling and random projection algorithms for approximating the product of two matrices. You may use Matlab, or C, or R, or any other software package you prefer to do your implementations. Please be sure to describe what you used in sufficient detail that someone else could reproduce your results. Let A be an $n \times d$ matrix, with $n \gg d$, and consider approximating the product $A^T A$. First, generate the matrices A from one of three different classes of distributions introduced below.

- Generate a matrix A from multivariate normal $N(1_d, \Sigma)$, where the (i, j)th element of $\Sigma_{ij} = 2 \times 0.5^{|i-j|}$.(Refer to as GA data.)
- Generate a matrix A from multivariate t-distribution with 3 degree of freedom and covariance matrix Σ as before. (Refer to as T3 data.)
- Generate a matrix A from multivariate t-distribution with 1 degree of freedom and covariance matrix Σ as before. (Refer to as T1 data.)

To start, consider matrices of size $n \times d$ equal to 500×50. (So, you should have three matrices, one matrix A generated in each of the above ways.)

(a) For each matrix, approximate the product $(A^T A)$ with the random sampling algorithm we discussed in class, i.e., by sampling with respect to a probability distribution that depends on the norm squared of the rows of the input matrix. Plot the probability distribution. Does it look uniform or nonuniform? Plot the performance of the spectral and Frobenius norm error as a function of the number of samples.

(b) For each matrix, approximate the product $(A^T A)$ with the random sampling algorithm we discussed in class, except that the uniform distribution, rather than the norm-squared distribution, should be used to construct the random sample. Plot the performance of the spectral and Frobenius norm error as a function of the number of samples. For which matrices are the results similar and for which are they different than when the norm-squared distribution is used ?

(c) Now you will implement the matrix approximation technique on the MNIST dataset for handwritten digit classification. Details about MNIST dataset can be found at `http://yann.lecun.com/exdb/mnist/`. We provide the dataset in **.mat** file so that you can easily import it into Matlab by using **load('mnist_matrix.mat')**. To import the dataset in Python you can use:
import scipy.io
data = scipy.io.loadmat('mnist_matrix.mat')
In **.mat** file you will find one matrix, $A \in \mathbb{R}^{60000 \times 784}$. For this matrix, approximate the product $(A^T A)$ with the random sampling algorithm we discussed in class, i.e., by sampling with respect to a probability distribution that depends on the norm squared of the rows of the input matrix. Plot the probability distribution. Dos it look uniform or nonuniform? Plot the performance of the spectral and Frobenius norm error as a function of the number of samples.