Homework 1, CS 6210, Fall 2020
Instructor: Austin R. Benson
Due Friday, September 18, 2020 at 10:19am ET on CMS (before lecture)

Policies

Collaboration. You are encouraged to discuss and collaborate on the homework, but you have to write up your own solutions and write your own code.
Programming language. You can use any programming language for the coding parts of the assignment. Code snippets that we provide and demos in class will use Julia.
Typesetting. Your write-up should be typeset with LaTeX. Handwritten homeworks are not accepted.
Submission. Submit your write-up as a single PDF on CMS: https://cmsx.cs.cornell.edu.

Problems

1. *SVD fun.*

   (a) Let $A \in \mathbb{R}^{n \times n}$. Show that there exists an orthogonal matrix $U \in \mathbb{R}^{n \times n}$ and symmetric positive semi-definite matrix $H \in \mathbb{R}^{n \times n}$ such that $A = UH$. (This is called the *polar decomposition* of A.)

   (b) Let $A \in \mathbb{R}^{n \times 1}$. Write down a thin (a.k.a. reduced or compact) SVD of A.

2. *Rank.*

   (a) Show that $\text{rank}(A) = \text{rank}(A^T)$ for any $A \in \mathbb{R}^{m \times n}$.

   (b) Let $A \in \mathbb{R}^{m \times n}$. Show that if $\text{rank}(A) = k$, then we can write $A = XY^T$, where $X \in \mathbb{R}^{m \times k}$, $Y \in \mathbb{R}^{n \times k}$, and $\text{rank}(X) = \text{rank}(Y) = k$.

   (c) Suppose we have a collection of n items, where item i has a latent score s_i, and not all scores are the same. We do not have access to the scores, but we do have an $n \times n$ matrix R of relative preferences, where $R_{ij} = s_j - s_i$ measures how much better item j is than item i. Show that $\text{rank}(R) = 2$ by writing a decomposition of the form in part (b).

   The factors X and Y in a low-rank decomposition of the form in (b) are not unique (why?). However, can you find a decomposition here that makes it easy to determine the latent scores s?

   (d) The matrix R from part (c) has the property that $R^T = -R$, which is called *skew symmetric*. Show that any skew symmetric matrix $S \neq 0$ satisfies $\text{rank}(S) \geq 2$.

3. *Norms.*

   (a) Show that $\|x\|_\infty \leq \|x\|_2 \leq \sqrt{n}\|x\|_\infty$ for any vector $x \in \mathbb{R}^n$.

   (b) Show that $\|A\|_\infty \leq \sqrt{n}\|A\|_2 \leq \sqrt{nm}\|A\|_\infty$ for any matrix $A \in \mathbb{R}^{m \times n}$.

   (c) Let $A \in \mathbb{R}^{n \times n}$. Show that $\rho(A) \leq \|A\|$ for any induced matrix norm $\|\cdot\|$, where $\rho(A)$ is the spectral radius of A (i.e., magnitude of the largest eigenvalue of A).

   (d) Let $A = xy^T$ for $x \in \mathbb{R}^m$ and $y \in \mathbb{R}^n$. Show that $\|A\|_2 = \|A\|_F = \|x\|_2 \cdot \|y\|_2$.

(e) The Kronecker product of matrix $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{p\times k}$, written as $A \otimes B$, is the matrix

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1n}B \\ a_{21}B & a_{22}B & \dots & a_{2n}B \\ \vdots & \vdots & & \vdots \\ a_{m1}B & a_{m2}B & \dots & a_{mn}B \end{bmatrix} \in \mathbb{R}^{mp\times nk}$$

Show that $\|A \otimes B\|_F = \|A\|_F \cdot \|B\|_F$. (If $A, B \in \mathbb{R}^{n\times n}$, this reduces the storage and computational complexity cost of computing $\|A \otimes B\|_F$ to $O(n^2)$, compared to $O(n^4)$ if we directly compute $A \otimes B$ first.)

(f) Let $A \in \mathbb{R}^{n\times n}$ be symmetric and $S \in \mathbb{R}^{n\times n}$ be skew symmetric (as defined in Question 2(d)). Show that $\|A + S\|_F^2 = \|A\|_F^2 + \|S\|_F^2$. *Hint:* It may be useful to interpret the Frobenius norm on matrices as the norm defined with respect to the *Frobenius inner product* over $\mathbb{R}^{m\times n}$: $\langle B, C\rangle = \sum_{i=1}^m \sum_{j=1}^n B_{ij}C_{ij}$.

4. *Sherman–Morrison update.*

Let $A \in \mathbb{R}^{n\times n}$ be nonsingular, and let x and y be vectors in $\mathbb{R}^n$ satisfying $1 + y^T A^{-1}x \neq 0$.

(a) Show that $A + xy^T$ is nonsingular, and that in particular,

$$(A + xy^T)^{-1} = A^{-1} - \frac{1}{1 + y^T A^{-1}x} A^{-1}xy^T A^{-1}.$$

(b) Suppose that we have done some pre-computations with A so that we can solve $Az = b$ in $O(n^2)$ time for unknown z. Design an algorithm that can solve $(A + xy^T)z = b$ in $O(n^2)$ time.

5. *Floating point error in matmul.*

For this question, you can assume that there is no underflow or overflow and that the input data is exactly representable in floating point.

(a) Let $x, y \in \mathbb{R}^n$. Show that $\text{fl}(\sum_{i=1}^n x_i y_i) = \sum_{i=1}^n x_i y_i(1 + \delta_i)$, where $|\delta_i| \leq n\epsilon$.

(b) Let $A \in \mathbb{R}^{m\times n}, B \in \mathbb{R}^{n\times p}$ and suppose that we compute the multiplication AB using the inner product algorithm discussed in class. Using part (a), show that $\text{fl}(AB) = AB + E$, where $|E| \leq n\epsilon \cdot |A||B|$ (here, the inequality is entrywise and $|M|$ denotes the matrix with entries equal to the absolute values of the entries of M).

(c) Does the result in part (b) change if we use the outer product matrix multiplication algorithm or any of the blocked algorithms discussed in class?

6. *Matmul implementation.*

For this question, we will examine the performance of algorithms for computing $C = AB$. Assume all matrices are square. Your submission should include a plot for both part (d) and part (e), along with explanations for parts (d)–(g). You can also include plots for (f) and (g) if they are helpful.

(a) Implement the inner product algorithm manually. Here's a Julia code snippet to start.

```
for i = 1:n
    for j = 1:n
        for k = 1:n
            @inbounds C[i, j] += A[i, k] * B[k, j]
        end
    end
end
```

(Is this code faster if the order of the first two loops is swapped?)

(b) Implement the row-by-row algorithm using mat-vecs. Here's a Julia code snippet to start.

```
for i = 1:n
    @inbounds C[i:i, :] += A[i:i, :] * B
end
```

(c) Implement the column-by-column algorithm using mat-vecs. Here's a Julia code snippet to start.

```
for j = 1:n
    @inbounds C[:, j] += A * B[:, j]
end
```

(d) Compare the performance of the algorithms from parts (a)–(c) and show that they all scale as $O(n^3)$. Also include the performance of the built-in matrix multiplication library function. Explain any performance differences. (In Julia, you can use the `@time` macro to measure running time.)

(e) Now assume that the matrix has a fixed dimension $n = Nb$, where b is a block size, and N is the number of blocks. Implement the blocked matrix multiplication algorithm. Here's a Julia code snippet to start.

```
for i = 1:N
    # I = ith row block indices
    for j = 1:N
        # J = jth row block indices
        # load local block of C
        @inbounds CIJ = C[I, J]
        for k = 1:N
            # K = inner dimension block
            @inbounds CIJ += A[I, K] * B[K, J]
        end
        @inbounds C[I, J] = CIJ
    end
end
```

We are slightly cheating in this code snippet by calling the built-in library function in the inner most loops, but that's OK for this question. (What happens when $b = 1$ or $b = n$?) If we were implementing the library function, we would at some point need to write down all of the operations.

Fix a matrix dimension n and measure the performance of the blocked algorithm as a function of the block size b. Compare this against the performance of algorithms in (a)–(c) and the library implementation. Explain the performance that you observe.

(f) Verify the result of Question 5(b), using 64-bit floats to find the "true" solution AB, and 32-bit floats to compute $\mathrm{fl}(AB)$. You can just use the built-in library matrix multiplication algorithms for the computations. Here's a Julia code snippet to start.

```
n = 100
A = rand(n, n)
B = rand(n, n)
```

```
A_fl = Float32.(A)
B_fl = Float32.(B)
C_true = A * B
C_fl = A_fl * B_fl
machine_precision = eps(Float32)
```

Explain what you observe. How do the bounds from 5(b) compare to what you find?

(g) Compare the outputs of your different matrix multiplication implementations for the same inputs. Do you notice any differences? Should there be differences? If you see differences, comment on their magnitude.