Lab Session 7

MA423: Matrix Computations July-November 2024 S. Bora

Important instructions:

(i) **Switch to `format long e` for all experiments.**

(ii) **Submit a <u>single livescript program</u> that contains all comments, answers and codes necessary to produce the required outputs. Ensure that the answers are correctly numbered and the file does not include any irrelevant material. The livescript program should be saved as <u>MA423YourrollnumberLab7.mlx</u>**

Part I: Applications of SVD

1. Perform experiments in Exercises 4.2.19-4.2.21 of *Fundamentals of Matrix Computations*. You will find them on page 272-273 of the second edition and pages 271-272 of the third edition. Write a small description of your experiments.

   The aim of the experiment in Exercise 4.2.21 of *Fundamentals of Matrix Computations* is to show that the Rank Revealing QR Decomposition is less efficient than the SVD method when detecting numerical rank deficiency.

2. This is a demonstration of image compression techniques using SVD. The following commands will first load a built-in 320×200 matrix X that represents the pixel image of a clown, computes its SVD $X = U\Sigma V^T$ and then displays the image when X is approximated by its best rank k approximation $X_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$ for a chosen value of k.

   ```
   load clown.mat; [U, S, V] = svd(X); colormap('gray');
   image(U(:, 1:k)*S(1:k, 1:k)*V(:,1:k)')
   ```

   The storage required for A_k is $k(m+n) = 520k$ words whereas the storage required for the full image is $n \times m = 6400$ words in this case. Therefore, $\frac{520k}{6400}$ gives the compression ratio for the compressed image. Also the relative error in the representation is $\frac{\sigma_{k+1}}{\sigma_1}$. Run the above commands for various choices of k and make a table that records the relative errors and compression ratios for each choice.

Part II: Relations between Matrix Decompositions and the Square Root of a Matrix

1. (**You must read the stuff explained here. It will not be discussed in the lectures.**) This exercise will teach you various matrix decompositions and their relationships. The results are true for real and complex matrices. So, we present the results by considering only real matrices.

   Polar decomposition: A complex number z has a polar representation $z = rw$, where $r \geq 0$ and $|w| = 1$. In a sense, matrices are non-commutative analogs of complex numbers. [This topic is beyond the scope of this course.] So, quite naturally, a matrix $A \in \mathbb{R}^{n\times n}$ admits a polar decomposition
   $$A = RW,$$
   where $R \geq 0$, meaning R is self-adjoint and positive semidefinite, and W is unitary. Here R and W are called polar factors of A. Is polar decomposition of A unique?

   More generally, if $A \in \mathbb{R}^{m\times n}$ then we have
   $$A = \begin{cases} RW, & \text{if } m \leq n, \\ \\ WR, & \text{if } m \geq n, \end{cases}$$
   where $W \in \mathbb{R}^{m\times n}$ is such that W is an isometry if $m \geq n$ and W^T is an isometry otherwise and $R \in \mathbb{R}^{p\times p}, p = \min(m, n)$, is self-adjoint and positive semidefinite. Note that for a square matrix A, the polar decomposition of A can be written as $A = RW$ or $A = WR$ whichever we prefer (of course, the polar factors are not the same in both cases).

   Proof: Using SVD of A, we obtain a simple and elegant proof of polar decomposition. Here are the details. Suppose that $m \geq n$ and consider the condensed SVD $A = U\Sigma V^*$, where $U \in \mathbb{R}^{m\times n}$ and $V, \Sigma = \operatorname{diag}(\sigma_1, \cdots, \sigma_n) \in \mathbb{R}^{n\times n}$. Then
   $$A = U\Sigma V^* = UV^*(V\Sigma V^*) = WR,$$
   where $W = UV^* \in \mathbb{R}^{m\times n}$ is easily seen to be isometry and $R = V\Sigma V^*$ is obviously selfadjoint and positive semidefinite. When $m \leq n$, the proof is similar. $\square$

   Your Task: Based on the above proof write a matlab function
   ```
   [ W, R] = polard1(A)
   % [W, R] = polard1(A) computes polar factors W and R of the matrix A.
   ```
   You should test the output by evaluating W and R on several random matrices by checking whether `norm(W*R-A)` (or `norm(R*W-A)`) `norm(W'*W-I)`, (or `norm(W*W'-I)`) `norm(R-R')`, `max(imag(eig(R)))` are of the order of unit roundoff and `min(eig(R))` is non-negative.

   Next, suppose that $A \in \mathbb{R}^{n\times n}$. Then the polar decomposition $A = RW$ gives
   $$AA^* = RWW^*R = R^2.$$
   This shows that the polar factor R is such that $R^2 = AA^*$. Considering the polar decomposition $A = WR$ it follows that $R^2 = A^*A$.

2. **Square root of a matrix:** Let $A \in \mathbb{C}^{n\times n}$. If a matrix $R \in \mathbb{C}^{n\times n}$ satisfies $R^2 = A$ then R is called a square root of A. Square root of a complex number is a complicated concept. So, quite naturally, square root of a matrix is a highly complicated concept. A matrix can have no square roots; try $A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$. A matrix can have finitely many square roots (how many? well, it all depends on Jordan canonical form of the matrix); try $A = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$. A matrix can have infinitely many square roots; try $A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. And finally, there may be funny square roots; e.g. $\begin{bmatrix} a & 1+a^2 \\ -1 & -a \end{bmatrix}^2 = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$, $a \in \mathbb{C}$.

   Fortunately, there are easy cases. For example, if A is diagonalizable and all eigenvalues of A are real and positive then A has a unique square root R such that all eigenvalues of R are real and positive. By the assumption $A = XDX^{-1}$, where $D = \text{diag}(\lambda_1, \cdots, \lambda_n)$ and $\lambda_j > 0$. The matlab command `[X, D] = eig(A)` gives the above decomposition. Defining $R := X\text{diag}(\sqrt{\lambda_1}, \cdots, \sqrt{\lambda_n})X^{-1}$, it follows that R is the unique square root of A with positive eigenvalues. Additionally, R is also positive definite. (WHY?)

   This shows that if A is PD (positive definite) then A has a unique PD square root R. In such a case, R is denoted by $A^{1/2}$ or $\sqrt{A}$. This is also called the *principal square root* of A.

   Since A is PD, Cholesky decomposition of A can also be used to construct $A^{1/2}$. Let $A = G^*G$ be the Cholesky decomposition. Now compute the SVD $G = U\Sigma V^*$ and define $R = V\Sigma V^*$. Then

$$A = G^*G = (U\Sigma V^*)^*(U\Sigma V^*) = V\Sigma^2V^* = R^2.$$

   To compute Cholesky decomposition you may use the matlab command `R= chol(A)`) or use your own function.

   The methods we have outlined are expensive and are applicable to special matrices. If a general matrix has a square root, then it has a unique square root whose eigenvalues have non-negative real parts. This is called the principal square root of the matrix. The matlab command `sqrtm(A)` computes principal square root of a general matrix A (if it exists) by using a method known as squaring and scaling. See `help sqrtm`.

   Your task: Write two matlab functions, say, `R1 = mysqrt1(A)` and `R2 = mysqrt2(A)` implementing the first and the second method, respectively, to compute the principal square root of a PD matrix A. Also compute `R3 = sqrtm(A).` Test these methods on the Hilbert matrix for various values of n. Plot the values `norm( A-R1 * R1 )/norm(A)`, `norm( A-R2 * R2)/norm(A)` and `norm( A-R3 * R3)/norm(A)` (in a single plot) for $n = 5, 7, 10, 12$. Which method is reliable and better? What is your conclusion?

3. We have used SVD of a matrix to compute polar factors and square root of A. When A is nonsingular, we can follow the backward direction, that is, we can compute SVD of A from polar factors of A. So, how to compute polar decomposition of A without using SVD?

   If A is nonsingular and $A = RW$ is a polar decomposition of A, then $R^2 = AA^*$. Therefore R is the square root of the PD matrix AA^*. So, we get R from `R = mysqrt1(A * A').` Once R is known, we obtain W by setting $W = R^{-1}A$.

Your task: Write a matlab function implementing above method to compute polar decomposition of a nonsingular matrix.

```
[W, R] = polard2(A)
% [W, R] = polard2(A) computes polar factors of a nonsingular matrix A.
```

Run `polard2.m` for several randomly generating nonsingular matrices and test the correctness of your output by performing the same checks on W and R that you performed for `polard1.m`.

Generate 15 nonsingular (random) test matrices A_j of size 20 such that $\sigma_{\min}(A_j) = 10^{-j+6}$ for $j = 1 : 15$. You should know how to generate such matrices. Now compute $[W_j, R_j] = \text{polard1}(A_j)$ and $[X_j, T_j] = \text{polard2}(A_j)$ for $j = 1 : 15$. For $j = 1 : n$, compute $\text{norm}(W_j^* W_j - I)$ and $\text{norm}(X_j^* X_j - I)$ and plot the results (in a single plot). What is your conclusion? Which method is better and reliable?

4. Finally, we use polar decomposition to compute SVD of a nonsingular matrix A. Here are the details. Compute a polar decomposition $A = RW$ (using `polard2`) and then compute $R = VDV^*$, where V consists of orthonormal eigenvectors of R and D is the diagonal matrix containing eigenvalues of R in descending order. Then $A = RW = VDV^*W = VD(W^*V)^*$ and $A = UD\hat{V}^*$ with $U = V$ and $\hat{V} = W^*V$ is an SVD of A.

   Your task: Write a matlab function $[\texttt{U}, \texttt{S}, \texttt{V}] = \texttt{mysvd(A)}$ implementing above method to compute SVD of A. For the 15 test matrices generated to test polar decomposition, compute $\|V_j^* V_j - I\|_2$ and $\|U_j^* U_j - I\|_2$ when U_j and V_j are obtained by your function as well as the matlab function `svd(A)`. Plot the results and conclude which method is better and reliable.