Computational Linear Algebra – Programming Assignment 3

(points: 50, due on December 10)

1 Important points

- All the commented codes have to be crisp and submitted online with a report. Submit the codes and report in a zip folder with folder name '*STUDENTNAME_pga3*.zip'.
- Inbuilt commands should not be used unless specified.
- The programs have to be coded in MATLAB or python.
- The matrices are given in the attached text file as well.

2 Questions

2.1 Power iterations (5+3+3)

Consider the matrix $A = \begin{bmatrix} 12 & 13 & 14 & 15 & 17 & 20 \\ 13 & 14 & 15 & 16 & 18 & 21 \\ 14 & 15 & 16 & 17 & 20 & 23 \\ 15 & 16 & 17 & 19 & 22 & 26 \\ 17 & 18 & 20 & 22 & 25 & 30 \\ 20 & 21 & 23 & 26 & 30 & 37 \end{bmatrix}$

1) Write a program using power method to find third largest eigenvalue and corresponding eigenvector of A. Can your program be used for any matrix? If yes, justify. If no, specify the set of matrices for which the program is applicable.

2) Use the same program in (1) to find the third smallest eigenvalue and the corresponding eigenvector of A. Compare the values with those obtained using the inbuilt commands.

3) Given the transition matrix of a discrete markov chain, compute its stationary distributions using power method. You can construct a transition matrix from a random matrix using the following code snippet,

n is the user input;
$A = randn(n, n)$;
$e = ones(n, 1)$;
$A \leftarrow A + 3ee^\top$;
$D \leftarrow$ diagonal matrix formed by $A * e$;
$A \leftarrow D^{-1}A$;

2.2 Application of schur factorization

Solve the sylvester equation $AX + XB = C$, where $A, B, C \in \mathbb{R}^{5\times 5}$ are given as,

$$A = \begin{bmatrix} 9 & 2 & 3 & 0 & 1 \\ 2 & 7 & -1 & 2 & -1 \\ 3 & -1 & 5 & -1 & 3 \\ 0 & 2 & -1 & 5 & 1 \\ 1 & -1 & 3 & 1 & 9 \end{bmatrix} \quad B = \begin{bmatrix} 13 & 1 & 5 & 4 & 3 \\ 1 & 5 & 0 & -1 & 0 \\ 5 & 0 & 6 & 3 & 3 \\ 4 & -1 & 3 & 5 & 4 \\ 3 & 0 & 3 & 4 & 9 \end{bmatrix} \quad C = \begin{bmatrix} 21.6 & 1.3 & 14.5 & 6.1 & 6.6 \\ 1.4 & 6.3 & -2.2 & 0.9 & -1.9 \\ 15.6 & -1.3 & 17.7 & 4.2 & 7.2 \\ 3.1 & 1.7 & -1.3 & 5.6 & 6.7 \\ 8.8 & -2.3 & 8,3 & 9.4 & 24.8 \end{bmatrix}$$

Note that the matrices A, B and C are selected such that solution X exists.

2.2.1 Kronecker product (3+2)

Convert the equation into linear system using kronecker product and solve the linear sytem using QR decomposition by modified Gram schmidt. You can use the program implemented in Assignment 1. Mention the computational complexity of implementing this linear system.

2.2.2 Schur factorization (3+3+5+3+2)

1) Modify the QR algorithm in Assignment 2 by adding the simple step of converting to heisenberg form before the iterations.

2) Compute Schur factorization of matrices $A = U\Lambda_A U^\top$ and $B = V\Lambda_B V^\top$ using the algorithm implemented in (1). Fix the maximum number of iterations to be 30.

3) Perform the following steps to convert into a simpler linear system and solve it.

- Substitute for A and B in the equation $AX + XB = C$.
- Multiply the entire equation by $U^\top$ from the left and V from the right.
- Substitute $U^\top XV$ as Y and solve the simpler linear system to obtain Y and thereby X.

4) Is the solution obtained by kronecker product exact? Calculate and plot the error between this solution and the one obtained using kronecker product by varying number of iterations from 5 to 70 in steps of 5.

5) Mention the computational complexity of the entire algorithm in terms of number of iterations.

2.3 Application of eigendecompositon (3+3+4+3+3+2)

Face classification into neutral and smiling: We investigate binary classification of face images. Download the images available at: `https://tinyurl.com/tctgs4w`

The database consists of the faces of 200 people and each person has two frontal images (one with a neutral expression and the other with a smiling facial expression). Thus in total, there are 400 full frontal face images. We will use 180 images each from the smiling and neutral sets as training data, and 20 images from each set as the test data. Our goal is to classify each image in the test set as smiling or neutral. Convert each 162×193 image into a (column) vector and store the vectors in a 31266×200 matrix $X1$ (for smiling images) and a 31266×200 matrix $X2$ (for the neutral images). In order to remove the bias, compute the mean of all columns in $[X1]$ (denoted by μ_1) and $[X2]$ (denoted by μ_2) and subtract it from the data as

$X1[j] \leftarrow X1[j] - \mu_1$,

$X2[j] \leftarrow X2[j] - \mu_2$.

for every column j. Note that μ_1 and μ_2 are vectors of size 31266. Let the submatrices formed by the first 180 columns of $X1$ and $X2$ be called $X1_t$ and $X2_t$ respectively. Similarly, the submatrices formed by the last 20 columns of $X1$ and $X2$ are $Y1$ and $Y2$ respectively.

1) Fix $k = 10$ and find the top k eigenvectors of $X1_t X1_t^\top$ and store them as $U1$, without calculating the huge matrix $X1_t X1_t^\top$. Note that the matrix is of size 31266×31266. Similarly find top k eigenvectors of $X2_t X2_t^\top$ and store them as $U2$. (inbuilt commands can be used)

2) Knowing that columns in $X1_t$ and $X2_t$ represent images, we can think $X1_t$ as 180 data points of size 31266 and hence $X1_t X1_t^\top$ is the sample covariance of the data. With this information in hand, explain the effect of operations in (1) and what $U1$ and $U2$ represent.

3) For each face vector a in the test set (column in $Y1$ or $Y2$), compute the relative error of estimating a by each of the two training sets as

$$e_1(a) = \frac{\|a - \mathrm{P}_{R(U1)}(a)\|}{\|a\|},$$

and

$$e_2(a) = \frac{\|a - \mathrm{P}_{R(U2)}(a)\|}{\|a\|},$$

where $\mathrm{P}_V(a)$ represents projection of a onto the space V. Classify a as smiling if $e_1(a) < e_2(a)$, and as neutral otherwise. Calculate the total number of misclassifications (classifying smiling as neutral and vice versa).

4) Repeat part (3) for $k = 20, 30, 40, 50$. Plot the number of misclassifications against the number of principal components.

5) Explain what exactly this classifier does and why the plot in (4) is justified.

6) What is the total computational complexity of the classifier?