ECE 6250, Fall 2019

Homework #10

Due Wednesday November 20, at the beginning of class

As stated in the syllabus, unauthorized use of previous semester course materials is strictly prohibited in this course.

1. Using your class notes, prepare a 1-2 paragraph summary of what we talked about in class in the last week. I do not want just a bulleted list of topics, I want you to use complete sentences and establish context (Why is what we have learned relevant? How does it connect with other things you have learned here or in other classes?). The more insight you give, the better.

2. The variable `faces` in `faces.mat` contains $D = 2414$ examples of images of human faces. Each face is a 32×32 pixel image, stored as a vector of length 1024. You have been provided with a function `plotFaces.m` which will plot the first several images (columns) in a matrix that you provide it. For example, the command `plotFaces(faces,4,4)` will plot a 4×4 table with the first 16 images in the matrix `faces`.

   In this problem we will show that this data is relatively tightly clustered around a comparatively low-dimensional subspace of $\mathbb{R}^{1024}$.

   (a) Based on the discussion of PCA in class, write code which can compute the optimal affine approximation to the data in `faces`. Specifically, let $\boldsymbol{x}_n$ denote the n^{th} column in `faces`. We are looking for the optimal approximation of the form $\boldsymbol{x}_n \approx \boldsymbol{\mu} + \boldsymbol{Q}\boldsymbol{\theta}_n$. Plot the resulting approximation for the first 16 faces when $r = 16$ (using `plotFaces`).

   (b) Plot each of the basis vectors for the subspace identified in the previous part when $r = 16$ (again using `plotFaces`).

   (c) How large does r need to be to ensure that the relative error between the original dataset and our approximation is less than 5%? How does this relate to the singular values of the original matrix?

   (d) Comment on the qualitative differences between the original dataset and the approximation produced by PCA and how this changes as we vary r.

3. Let $\boldsymbol{A}$ be a tri-diagonal matrix:

$$\boldsymbol{A} = \begin{bmatrix} d_1 & c_1 & 0 & 0 & 0 & \cdots & 0 \\ f_1 & d_2 & c_2 & 0 & 0 & \cdots & 0 \\ 0 & f_2 & d_3 & c_3 & 0 & \cdots & 0 \\ 0 & 0 & f_3 & d_4 & c_4 & \cdots & 0 \\ \vdots & \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & & f_{N-2} & d_{N-1} & c_{N-1} \\ 0 & 0 & 0 & \cdots & & f_{N-1} & d_N \end{bmatrix}$$

(a) Argue that the LU factorization of $\boldsymbol{A}$ has the form

$$\boldsymbol{A} = \begin{bmatrix} * & 0 & 0 & 0 & \cdots & 0 \\ * & * & 0 & 0 & \cdots & 0 \\ 0 & * & * & 0 & \cdots & 0 \\ 0 & 0 & * & * & \cdots & 0 \\ \vdots & & & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & & * & * \end{bmatrix} \begin{bmatrix} * & * & 0 & 0 & 0 & \cdots & 0 \\ 0 & * & * & 0 & 0 & \cdots & 0 \\ 0 & 0 & * & * & 0 & \cdots & 0 \\ \vdots & & & & \ddots & \ddots & \\ 0 & 0 & \cdots & & & * & * \\ 0 & 0 & \cdots & & & 0 & * \end{bmatrix},$$

where $*$ signifies a non-zero term.

(b) Write down an algorithm that computes the LU factorization of $\boldsymbol{A}$, meaning the $\{\ell_i\}$, $\{u_i\}$, and $\{g_i\}$ below

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ \ell_1 & 1 & 0 & 0 & \cdots & 0 \\ 0 & \ell_2 & 1 & 0 & \cdots & 0 \\ 0 & 0 & \ell_3 & 1 & \cdots & 0 \\ \vdots & & & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & & \ell_{N-1} & 1 \end{bmatrix} \begin{bmatrix} u_1 & g_1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & u_2 & g_2 & 0 & 0 & \cdots & 0 \\ 0 & 0 & u_3 & g_3 & 0 & \cdots & 0 \\ \vdots & & & & \ddots & \ddots & \\ 0 & 0 & \cdots & & & u_{N-1} & g_{N-1} \\ 0 & 0 & \cdots & & & 0 & u_N \end{bmatrix},$$

I will get you started:

$u_1 = d_1$
for $k = 2, \ldots, N$
    $g_{k-1} =$
    $\ell_{k-1} =$
    $u_k =$
end

4. Write a MATLAB function that uses the power iteration procedure described in the class notes to find the largest eigenvalue of a matrix $\boldsymbol{A}$ and its associated eigenvector. Keep track of how many iterations it takes until the estimated eigenvalue does not change more than `eps` (i.e., machine precision) between iterations for the following matrices:

- A random symmetric 10×10 matrix.
- A random symmetric 100×100 matrix.
- A random symmetric 1000×1000 matrix.

You can form a random symmetric matrix using the `randn` command to generate a random $\boldsymbol{B}$ and then forming $\boldsymbol{A} = \boldsymbol{B} + \boldsymbol{B}^{\mathrm{T}}$. For each matrix size, repeat the above for many different choices of random matrices and produce a scatter plot that compares the number of iterations to $|\lambda_1|/|\lambda_2|$ (where λ_1 is the eigenvalue with largest magnitude and λ_2 is the eigenvalue with second largest magnitude). Comment on the following:

(a) Does the number of iterations seem to increase with the size of the matrix? Why should or should not that be the case?

(b) How do the eigenvalues found using your code compare with the largest (magnitude) eigenvalues found using the MATLAB `eig` command?

5. Suppose that you wish to solve for $\boldsymbol{x}$ given that $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ where

$$\boldsymbol{A} = \begin{bmatrix} 10000 & 10001 \\ 10001 & 10002 \\ 10002 & 10003 \\ 10003 & 10004 \\ 10004 & 10005 \end{bmatrix}, \qquad \boldsymbol{b} = \begin{bmatrix} 20001 \\ 20003 \\ 20005 \\ 20007 \\ 20009 \end{bmatrix}.$$

Note, the exact solution is $\boldsymbol{x} = \begin{bmatrix} 1 & 1 \end{bmatrix}^T$.

(a) Determine the condition number of $\boldsymbol{A}^T\boldsymbol{A}$.

(b) Compute the least-squares solution using the formula $\hat{\boldsymbol{x}} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b}$ explicitly.

(c) Compute the solution using the Cholesky factorization. You should show the matrix equations that you used to arrive at your solution along with any MATLAB code or commands (you can use the MATLAB `chol( )` command).

(d) Compute the solution using the QR decomposition. You should show the matrix equations that you used to arrive at your solution along with any MATLAB code or commands (you can use the MATLAB `qr( )` command).

(e) Compute the solution using the MATLAB backslash command "\". View the online documentation of the backslash command (also known as `mldivide()` and determine which method it is using to solve for $\hat{\boldsymbol{x}}$.

(f) Compare the answers and comment. If a method gives you an incorrect answer, see if you can identify exactly where things might be going wrong.

Include your code with your assignment submission.