Assignment 4: CS 663

Due: 26th October before 11:55 pm

Remember the honor code while submitting this (and every other) assignment. You may discuss broad ideas with other students or ask me for any difficulties, but the code you implement and the answers you write must be your own. We will adopt a zero-tolerance policy against any violation.

Submission instructions: Follow the instructions for the submission format and the naming convention of your files from the submission guidelines file in the homework folder. However, please do *not* submit the face image databases in your zip file that you will upload on moodle. Please see `assignment4_SVD_FaceRecognition.rar`. Upload the file on moodle *before* 11:55 pm on 26th October. We will not penalize submission of the files till 10 am on 27th October. **No late assignments will be accepted after this time.** Please preserve a copy of all your work until the end of the semester.

1. In this part, you will implement a mini face recognition system. Download the ORL face database from the homework folder. It contains 40 sub-folders, one for each of the 40 subjects/persons. For each person, there are ten images in the appropriate folder named 1.pgm to 10.pgm. The images are of size 92 by 110 each. Each image is in the pgm format. You can view/read the images in this format, either through MATLAB or through image viewers like IrfanView on Windows, or xv/display/gimp on Unix. Though the face images are in different poses, expressions and facial accessories, they are all roughly aligned (the eyes are in roughly similar locations in all images). For the first part of the assignment, you will work with the images of the first 32 people (numbers from 1 to 32). For each person, you will include the first six images in the training set (that is the first 6 images that appear in a directory listing as produced by the `dir` function of MATLAB) and the remaining four images in the testing set. You should implement the recognition system by using the `eig` or `eigs` function of MATLAB on an appropriate data matrix. Record the recognition rate using squared difference between the eigencoefficients while testing on all the images in the test set, for $k \in \{1, 2, 3, 5, 10, 15, 20, 30, 50, 75, 100, 150, 170\}$. Plot the rates in your report in the form of a graph. Now modify the required few lines of the code but using the `svd` function of MATLAB (on the $\boldsymbol{L}$ matrix as defined in class) instead of `eig` or `eigs`.

   Repeat the same experiment (using just the `eig` or `eigs` routine) on the Yale Face database from the homework folder. This database contains about 64 images each of 38 individuals (*labeled from 1 to 39, with number 14 missing; some folders have slightly less than 64 images*). Each image is in pgm format and has size 192 by 168. The images are taken under different lighting conditions but in the same pose. Take the first 40 images of every person for training and test on the remaining 24 images (that is the first 40 images that appear in a directory listing as produced by the `dir` function of MATLAB). Plot in your report the recognition rates for $k \in \{1, 2, 3, 5, 10, 15, 20, 30, 50, 60, 65, 75, 100, 200, 300, 500, 1000\}$ based on (a) the squared difference between all the eigencoefficients and (b) the squared difference between all *except* the three eigencoefficients corresponding to the eigenvectors with the three largest eigenvalues. Display in your report the reconstruction of any one face image from the ORL database using $k \in \{2, 10, 20, 50, 75, 100, 125, 150, 175\}$ values. Plot the 25 eigenvectors (eigenfaces) corresponding to the 25 largest eigenvalues using the subplot or subimage commands in MATLAB. [35 points]

2. What will happen if you test your system on images of people which were not part of the training set? (i.e. the last 8 people from the ORL database). What mechanism will you use to report the fact that there is no matching identity? Work this out carefully and explain briefly in your report. Write code to test whatever you propose on all the 32 remaining images (i.e. 8 people times 4 images per person), as also the entire test set containing 6 images each of the first 32 people. How many false positives/negatives did you get? [15 points]

3. Consider a set of N vectors $\mathcal{X} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_N}\}$ each in $\mathbb{R}^d$, with average vector $\boldsymbol{\bar{x}}$. We have seen in class that the direction $\boldsymbol{e}$ such that $\sum_{i=1}^{N} \|\boldsymbol{x_i} - \boldsymbol{\bar{x}} - (\boldsymbol{e} \cdot (\boldsymbol{x_i} - \boldsymbol{\bar{x}}))\boldsymbol{e}\|^2$ is minimized, is obtained by maximizing $\boldsymbol{e}^t\boldsymbol{C}\boldsymbol{e}$, where $\boldsymbol{C}$ is the covariance matrix of the vectors in $\mathcal{X}$. This vector $\boldsymbol{e}$ is the eigenvector of matrix $\boldsymbol{C}$ with the highest eigenvalue. Prove that the direction $\boldsymbol{f}$ perpendicular to $\boldsymbol{e}$ for which $\boldsymbol{f}^t\boldsymbol{C}\boldsymbol{f}$ is maximized, is the eigenvector of $\boldsymbol{C}$ with the second highest eigenvalue. For simplicity, assume that all non-zero eigenvalues of $\boldsymbol{C}$ are distinct and that rank$(\boldsymbol{C}) > 2$. Extend the derivation to handle the case of a unit vector $\boldsymbol{g}$ which is perpendicular to both $\boldsymbol{e}$ and $\boldsymbol{f}$ which maximizes $\boldsymbol{g}^t\boldsymbol{C}\boldsymbol{g}$. [10 points]

4. The aim of this exercise is to help you understand the mathematics behind PCA more deeply. Do as directed: [5+5+5+5=20 points]

   (a) Prove that the covariance matrix in PCA is symmetric and positive semi-definite.

   (b) Prove that the eigenvectors of a symmetric matrix are orthonormal.

   (c) Consider a dataset of some N vectors in d dimensions given by ~~$\{\boldsymbol{x_i}\}_{i=1}^{d}$~~ $\{\boldsymbol{x_i}\}_{i=1}^{N}$ with mean vector $\boldsymbol{\bar{x}}$. Note that each $\boldsymbol{x_i} \in \mathbb{R}^d$ and also $\boldsymbol{\bar{x}} \in \mathbb{R}^d$. Suppose that only k eigenvalues of the corresponding covariance matrix are large and the remaining are very small in value. Let $\boldsymbol{\tilde{x}_i}$ be an approximation to $\boldsymbol{x_i}$ of the form $\boldsymbol{\tilde{x}_i} = \boldsymbol{\bar{x}} + \sum_{l=1}^{k} \boldsymbol{V_l}\alpha_{il}$ where $\boldsymbol{V_l}$ stands for the lth eigenvector (it has d elements) and α_{il} (it is a scalar) stands for the lth eigencoefficient of $\boldsymbol{x_i}$. Argue why the error $\frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{\tilde{x}_i} - \boldsymbol{x_i}\|_2^2$ will be small. What will be the value of this error in terms of the eigenvalues of the covariance matrix?

   (d) Consider two uncorrelated zero-mean random variables (X_1, X_2). Let X_1 belong to a Gaussian distribution with variance 100 and X_2 belong to a Gaussian distribution with variance 1. What are the principal components of (X_1, X_2)? If the variance of X_1 and X_2 were equal, what are the principal components?

5. The aim of this exercise is to help you understand the mathematics of SVD more deeply. Do as directed: [5+5+10=20 points]

   (a) Argue that the non-zero singular values of a matrix $\boldsymbol{A}$ are the square-roots of the eigenvalues of $\boldsymbol{A}\boldsymbol{A}^T$ or $\boldsymbol{A}^T\boldsymbol{A}$. (Make arguments for both)

   (b) Show that the squared Frobenius norm of a matrix is equal to the sum of squares of its singular values.

   (c) A students tries to obtain the SVD of a $m \times n$ matrix $\boldsymbol{A}$ using eigendecomposition. For this, the student computes $\boldsymbol{A}^T\boldsymbol{A}$ and assigns the eigenvectors of $\boldsymbol{A}^T\boldsymbol{A}$ (computed using the `eig` routine in MATLAB) to be the matrix $\boldsymbol{V}$ consisting of the right singular vectors of $\boldsymbol{A}$. Then the student also computes $\boldsymbol{A}\boldsymbol{A}^T$ and assigns the eigenvectors of $\boldsymbol{A}\boldsymbol{A}^T$ (computed using the `eig` routine in MATLAB) to be the matrix $\boldsymbol{U}$ consisting of the left singular vectors of $\boldsymbol{A}$. Finally, the student assigns the non-negative square-roots of the eigenvalues (computed using the `eig` routine in MATLAB) of either $\boldsymbol{A}^T\boldsymbol{A}$ or $\boldsymbol{A}\boldsymbol{A}^T$ to be the diagonal matrix $\boldsymbol{S}$ consisting of the singular values of $\boldsymbol{A}$. He/she tries to check his/her code and is surprised to find that $\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T$ is not equal to $\boldsymbol{A}$. Why could this be happening? What processing (s)he do to the eigenvectors computed in order rectify this error? (Note: please try this on your own in MATLAB.)