MAT 167: Homework Assignment #7 (due Wednesday, May 30)

First of all, do the following:

- Read Chapters 6 9.

Problem 1. Using MATLAB, do the following PCA experiments.

(a) Download the face database from the following link:
http://www.math.ucdavis.edu/~saito/courses/167.s12/faces.mat . Then, load this file into your MATLAB session (by now, you should be familiar with loading a data file into MATLAB). This file contains a 3D array called `faces` of size $128 \times 128 \times 143$. Before doing any PCA, SVD, etc., let's change this 3D array into a 2D array (i.e., a matrix) as follows:

```
>> X = reshape(faces, [128^2 143]);
```

This creates a data matrix X. Now, each column vector of X represents a face image of 128^2. Let's compute the average face via:

```
>> aveface = mean(X, 2);
```

I already explained what 2 in the input arguments means in the class, but you can always check the meaning by running "help mean" in your MATLAB session. Once you compute the average face, display it on your screen using the following commands:

```
>> imagesc(reshape(aveface, [128 128])); axis image; axis off
>> colormap(gray);
```

Then, submit the plot of this average face.

(b) Now, subtract this average face from each column of X by

```
>> X0 = X - repmat(aveface, [1 143]);
```

This $X0$ represents $\widetilde{X}$ in my lecture notes. Now, check the mean of $X0$ is really a zero vector or not by computing the absolute error between the mean of $X0$ and the zero vector of length $128^2 = 16384$ using 2-norm. Report the result.

(c) Finally, compute the PCA using the reduced SVD as follows:

```
>> [U, S, V] = svd(X0, 0);
```

As I explained in my lecture, adding 0 when you use `svd` in MATLAB means the reduced SVD. Do not forget to add this 0; otherwise, you may need to kill your MATLAB session because it is too expensive to compute the full SVD of this matrix size. Now, display the first 4 principal axis vectors `U(:,1)`, `U(:,2)`, `U(:,3)`, `U(:,4)` as images as follows:

```
>> for k=1:4
    subplot(2,2,k); imagesc(reshape(U(:,k), [128 128])); axis image; axis off;
   end
```

Then, print this figure, and submit it.

(d) As I discussed in the class, we can approximate each face in this database using the top k PCA components as:

$$\tilde{\boldsymbol{x}}_j \approx \bar{\boldsymbol{x}} + U(:,1:k)U(:,1:k)^{\mathsf{T}}\tilde{\boldsymbol{x}}_j \quad j = 1,\dots,143.$$

Do the k-term approximation experiments of No.2 face with $k = 5, 55, 105$, as follows:

```
>> % check the min and max value of No.2 face.
>> minval = min(X(:,2)); maxval = max(X(:,2));
>> subplot(2,2,1); imagesc(reshape(X(:,2), [128 128]), [minval maxval]);
>> axis image; axis off;
>> % Note that it is important to compute U(:,1:5)'*X0(:,2) first, i.e.,
>> % use the parentheses there.  Without these parentheses, MATLAB tries to
>> % compute U(:,1:5)*U(:,1:5)' first, which is a matrix of size 16384 x 16384,
>> % i.e., too big to handle on your computer!
>> X2approx5 = aveface+U(:,1:5)*(U(:,1:5)'*X0(:,2));
>> subplot(2,2,2); imagesc(reshape(X2approx5, [128 128]), [minval maxval]);
>> axis image; axis off;
>> X2approx55 = aveface+U(:,1:55)*(U(:,1:55)'*X0(:,2));
>> subplot(2,2,3); imagesc(reshape(X2approx55, [128 128]), [minval maxval]);
>> axis image; axis off;
>> X2approx105 = aveface+U(:,1:105)*(U(:,1:105)'*X0(:,2));
>> subplot(2,2,4); imagesc(reshape(X2approx105, [128 128]), [minval maxval]);
>> axis image; axis off;
```

Print this figure and submit it.

(e) Let's compute the approximation errors via

```
>> X2err5 = X0(:,2) - X2approx5;
>> X2err55 = X0(:,2) - X2approx55;
>> X2err105 = X0(:,2) - X2approx105;
>> subplot(1,3,1);imagesc(reshape(X2err5,[128 128])); axis image; axis off;
>> subplot(1,3,2);imagesc(reshape(X2err55,[128 128]),[min(X2err5),max(X2err5)]);
>> axis image; axis off;
>> subplot(1,3,3);imagesc(reshape(X2err105,[128 128]),[min(X2err5),max(X2err5)]);
>> axis image; axis off;
```

Then, print this figure and submit it. Also, compute the Frobenius norm of these errors, confirm that the error decreases for larger k compared to the smaller k, and report the result.

(f) Finally, repeat Part (d) and (e) with No.143 face instead of No.2 face.

Problem 2: Suppose the $m \times n$ matrix A has the form

$$A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix},$$

where A_1 is a nonsingular matrix of dimension $n \times n$ and A_2 is an arbitrary matrix of dimension $(m-n) \times n$. Prove that $\|A^\dagger\|_2 \le \|A_1^{-1}\|_2$.

Problem 3: Using MATLAB, do the following K-means experiments. (Note that if you have a personal copy of MATLAB on your laptop computer, MATLAB's `kmeans` algorithm may not be accessible because it is a part of the statistics toolbox, which your copy may not include. Run `help kmeans` in your MATLAB session. Then you can see whether your MATLAB has this `kmeans` function. If not, you need do this experiments in our computer room.)

(a) Download the breast cancer data from the following link:
http://www.math.ucdavis.edu/~saito/courses/167.s12/breastcancer.mat . Then, load this file into your MATLAB session. This file contains a matrix called `data` of size 683 $\times$ 11, which represents 9 measurements (various geometric features) taken from images of 683 cells in breasts. `data(j,1)` is the Id. number of the jth cell, `data(j,2:10)` contains those 9 measurements of the jth cell, and `data(j,11)` contains the diagnosis by the doctors, i.e., *benign if this value is 2; malignant if this value is 4*. Run the K-means algorithm `kmeans` in MATLAB with $K = 2$ by

```
>> idx = kmeans(data(:,2:10), 2);
```

The output `idx` contains the cluster number of each data point, and since you used $K = 2$, the cluster value of the jth cell, i.e., `idx(j)` is either 1 or 2. Create now the table of classification results like the one shown as Table 9.1 of the textbook, i.e.,

	M	B
M	222	17
B	9	435

In the above table, 222 malignant cells were correctly classified as "malignant'," but 17 malignant cells were erroneously classified as "benign." On the other hand, 435 benign cells were correctly classified as "benign," but 9 benign cells were erroneously classified as "malignant."

[Hint: `2*idx` gives you the array whose entries are either 2 or 4 so that that is easier to compare with the ground truth diagnosed by the doctors. However, one thing you need to pay attention to is that the value 2 in `2*idx` may or may not correspond to the "benign" class because K-means algorithm with K outputs just two clusters without any label information. It is up to you to decide which cluster should match the "benign" class and which cluster should match the "malignant" class. You should pick the matching/labeling that gives you lower misclassification rates.]

(b) Compute the *false-positive rate* and *false-negative rate* of your classification results. These are defined as:

$$\text{false-positive rate} := \frac{\text{Number of benign cells incorrectly classified as ``malignant''}}{\text{Total number of benign cells}}$$

$$\text{false-negative rate} := \frac{\text{Number of malignant cells incorrectly classified as ``benign''}}{\text{Total number of malignant cells}}$$

Hence if the table shown above is your result, then these rates are:

$$\text{false-positive rate} = \frac{9}{444} = 0.0203.$$
$$\text{false-negative rate} = \frac{17}{239} = 0.0711.$$

(c) Repeat the above classification experiments using K-means algorithm 9 more times. Then, generate the classification table of 2 $\times$ 2 consisting of the average rates of these 10 experiments (the first one in Part (a) and these 9 experiments in Part (c)). Then, compute the average false-positive and false-negative rates of these 10 experiments. Report these results.