MAT 167 Programming Project (due Friday, May 27)

First of all,

- Read Chapters 6 and 11.

Using MATLAB, do the following handwritten digit recognition computations.

Step 01 Download the handwritten digit database

"USPS.mat"

from SmartSite and load this file into your MATLAB session.

(a) This file contains four arrays

- `train_patterns`
- `test_patterns`

of size 256×4649 and

- `train_labels`
- `test_labels`

of size 10×4649. You may find it helpful to think of these arrays as matrices. The arrays `train_patterns` and `test_patterns` contain a raster scan of the 16×16 gray level pixel intensities that have been normalized to lie within the range $[-1, 1]$. The arrays `train_labels` and `test_labels` contain the true information about the digit images. That is, if the jth handwritten digit image in `train_patterns` truly represents the digit i, then the $(i+1, j)$th entry of `train_labels` is $+1$, and all the other entries of the jth column of `train_labels` are -1.

(b) Now, display the first 16 images in `train_patterns` using `subplot(4,4,k)` and `imagesc` functions in MATLAB. Print out the figure and include it in your Programming Project LaTeX and PDF files.

<u>Hint</u>: You need to `reshape` each column into a matrix of size 16×16 followed by transposing it in order to display it correctly.

Step 02 Compute the mean digits in the `train_patterns`, put them in a matrix called `train_aves` of size 256×10, and display these 10 mean digit images using `subplot(2,5,k)` and `imagesc`. Print out the figure as a PDF file and include it in your LaTeX and PDF documents.

Hint: You can gather (or pool) all the images in `train_patterns` corresponding to digit $k-1$ $(1 \le k \le 10)$ using the following MATLAB command:

```
>> train_patterns(:, train_labels(k,:)==1);
```

Step 03 Now conduct the simplest classification computations as follows:

(a) First, prepare a matrix called `test_classif` of size 10×4649 and fill this matrix by computing the Euclidean distance (or its square) between each image in the `test_patterns` and each mean digit image in `train_aves`.

Hint: the following line computes the squared Euclidean distances between all of the test digit images and the kth mean digit of the training dataset by one line:

```
>> sum((test_patterns-repmat(train_aves(:,k),[1 4649])).^2);
```

(b)) Compute the classification results by finding the position index of the minimum of each column of `test_classif`. Put the results in a vector `test_classif_res` of size 1×4649.

Hint: You can find the position index giving the minimum of the jth column of `test_classif` by

```
>> [tmp, ind] = min(test_classif(:,j));
```

Then, the variable `ind` contains the position index, an integer between 1 and 10, of the smallest entry of `test_classif(:,j)`.

(c)) Finally, compute the confusion matrix `test_confusion` of size 10×10, print out this matrix, and submit your results.

Hint: First gather the classification results corresponding to the $k-1$st digit by

```
>> tmp=test\_classif\_res(test_labels(k,:)==1);
```

This `tmp` array contains the results of your classification of the test digits whose true digit is $k-1$ for $1 \leq k \leq 10$. In other words, if your classification results were perfect, all the entries of `tmp` would be k. But in reality, this simplest classification algorithm makes mistakes, so `tmp` contains values other than k. You need to count how many entries have the value j in `tmp`, for $j = 1:10$. This will give you the kth row of the `test_confusion` matrix.

Step 04 Finally, conduct the SVD-based classification computations.

(a) Pool all of the images corresponding to the kth digit `train_patterns`, compute the rank 17 SVD of that set of images, i.e., the first 17 singular values and vector), and put the left singular vectors (or the matrix U) of the kth digit into the array `train_u` of size $256 \times 17 \times 10$. For $k = 1:10$, you can do the following:

```
>> [train_u(:,:,k),tmp,tmp2] = svds(train_patterns(:,train_labels(k,:)==1),17);
```

You do not need the singular values and right singular vectors in this computation.

(b) Compute the expansion coefficients of each test digit image with respect to the 17 singular vectors of each train digit image set. In other words, you need to compute 17×10 numbers for each test digit image. Put the results in the 3D array `test_svd17` of size $17 \times 4649 \times 10$. This can be done with the commands

```
>> for k=1:10
     test_svd17(:,:,k) = train_u(:,:,k)' * test_patterns;
   end
```

(c) Next, compute the error between each original test digit image and its rank 17 approximation using the kth digit images in the training data set. The idea of this classification is that a test digit image should belong to the class of the k^*th digit if the corresponding rank 17 approximation is the best approximation (i.e., the smallest error) among 10 such approximations. Prepare a matrix `test_svd17res` of size 10×4649, and put those approximation errors into this matrix.

Hint: The rank 17 approximation of test digits using the 17 left singular vectors of the kth digit training images can be computed by `train_u(:,:,k)*test_svd17(:,:,k);`

(d) Finally, compute the confusion matrix using this SVD-based classification method by following the same strategy as in **Step 03(b)** and **Step 03(c)** above. Name this confusion matrix `test_svd17_confusion`. Include this matrix in your report and submit your results.

Step 05 **Analyze your results!**

(a) For **Step 01** explain your understanding of the data structure in which the images of the digits are stored. In particular, include a brief explanation of the difference between the training data and the test data. (This is a simple example of *machine learning*. It is most likely the first machine learning algorithm - or algorithms - to be widely used in the 'real world'.)

(b) Comment on the intermediate results at the end of each **Step 02**, **Step 03** and **Step 04**. How effective is this particular algorithm? Include some thoughts to support your comments.

(c) Summarize all of your results in a separate section at the end. Compare the results from **Step 02**, **Step 03**, and **Step 04**. Which of the three algorithms yields the best result? The second best result? Why?