- Read Chapter 6 and Chapter 10 up to, but not including, Section 10.2.

Write a MATLAB code to perform the following handwritten digit recognition computations.

Step 01 Download the handwritten digit database

"USPS.mat"

from CANVAS 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.

[**Hint:** 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 Read the description of this step in Chapter 10.01 of the textbook and/or Professor Saito's Lecture 21. Compute the mean digits in the `train_patterns` and 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 $k - 1$ $(1 \leq k \leq 10)$ using the following MATLAB command:

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

Step 03 Read the description of this step in Chapter 10.01 of the textbook and/or Professor Saito's Lecture 21. 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 with one line of MATLAB code:

```
>> 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 in the PDF file containing your report.

[**Hint:** First gather the classification results corresponding tothe $k-1$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 Read the description of this step in Chapter 10.02 of the textbook and/or Professor Saito's Lecture 21. Now conduct an SVD-based classification computation.

(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 vectors, 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 ca do this with the following code:

```
>> [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 kth 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 f**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 IN A WELL WRITTEN REPORT!

(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*. These are most likely the first machine learning algorithms to be widely used in the 'real world'.)

(b) Give an explanation of what you are doing in **Step 02**, and why you are doing it. You will find some helpful comments concerning **Step 02** in Chapter 10.01 of the textbook. Include some thoughts to support your comments.

(c) Comment on the intermediate results at the end of **Step 03** and at the end of **Step 04**. How effective is each algorithm; i.e, for that particular algorithm what percentage of *each* digit is identified correctly? Which digit is the most difficult to identify correctly? Which digit is the easiest to identify correctly? You can obtain all of this information from the confusion matrices you produced in **Step 03** and **Step 04**. Include some thoughts to support your comments. In particular, in YOUR OWN WORDS explain the theory that is behind the algorithm in **(a)**–**(d)**. (This is discussed in detail in Chapter 10.2 of the textbook.)

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

Step 06 Submit a *well documented* MATLAB program named

"Digit_Recognition_youremailname.m"

This program should perform all of the tasks in **Step 01** to **Step 04** above without any user input. It is sufficient to have your program print the various images and tables on the computer screen. In particular, your program does not have to have produce a PDF file containing the images of the digits produced in **Step 01(b)** and **Step 02**.

Again, here is a description of what is meant by a *well documented* MATLAB program.

DO NOT submit only the MATLAB source code without comments. Furthermore, DO NOT include the bare minimum of explanation for each subsection of your code. Please consider using an active mind when including comments in your program. In particular, as technicians and highly educated individuals, it is worth your time to describe what you are doing IN OUR OWN WORDS for each individual segment of the code; i.e., each portion of the code that performs a separate task, even if it is '*only*' inputting a file. For example, 'What is the format of the file: binary, text, MATLAB data structures? What is contained in the file? How is it stored? Relate the algorithm(s) back to the theory we have been studying in lecture and in the homework assignments. When you read your own code, you should be able to easily identify what you have learned from this writing the program, and how this relates to the themes presented in lectures and in the textbook.