Case Study I: Clustering and Classification *(DUE: WEDNESDAY, OCT 13 @ 4:00PM)*

Textbook Reading: Chapter 4.4 and 4.5

A. Introduction

In this case study you will use the skills and methods you have learned in linear algebra and MATLAB to classify handwritten digits.

The main idea of classification is to be able to determine the "meaning" of an 'unlabeled' input. For this case study, what this means is to determine the numerical "meaning" of the handwritten digits based on their vector representation. There are many ways to accomplish classification; here we will do it using clustering (see also Chapter 4.4 and 4.5). You will be using MATLAB to build a method, using clustering, that takes an image of a handwritten digit and determines what number, from 0 to 9, is depicted.

B. Additional Instructions: Clustering

Your design should be implemented in MATLAB using the skeleton file provided on Canvas. This file provides code that reads in 28x28 images from the MNIST database (see textbook reading above). Each image is read in as a 784-dimensional vector. Each vector has entries whose values are between 0 and 255, indicating levels of gray.

You should implement the k-means algorithm as discussed in class and presented in the textbook in order to cluster these vectors into appropriate numerical groups. ***Do not simply use the 'kmeans' function built-in to MATLAB. Rather, implement the algorithm from the ground up based on the concepts of norm and distance, guided by the skeleton file.***

A working algorithm will produce k centroids. Any given image will be closest to one of these k centroids.

Example:

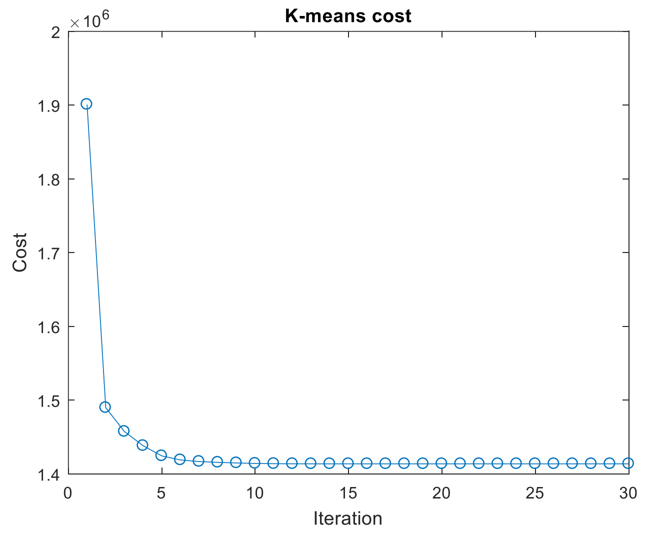

Figure 1. The k-means cost function, as a function of iterations

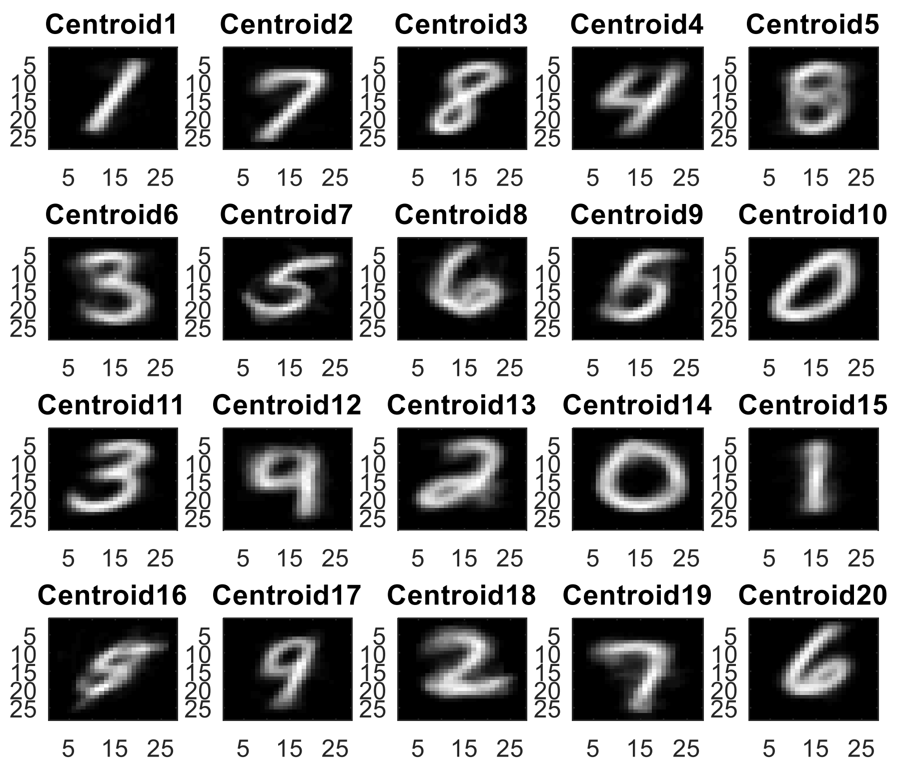

Figure 2. In this example, *k* was set to 20. The resulting centroids are depicted above.

C. Additional Instructions: Classification

In order to perform classification, you must assign each centroid to a numerical meaning. You can then test the performance of your classifier as follows

1. Take an input image and use a nearest neighbor criteria to assign it to a centroid.
2. Check whether the meaning of that centroid matches that of the test image.

A correct classification occurs when the test image is consistent with the numerical interpretation of the centroid to which it is closest.

Example:

In Figure 2, we depict 20 centroids resulting from running the k-means algorithm on the set of 1000 training images. For the purposes of classification, we must ascribe numerical meaning to each of these centroids. For example, by visual adjudication, we choose (across, then down): 1,7,8,4,8,3,5,6,5,0,3,9,2,0,1,5,9,2,7,6

Once we have labeled the centroids in this way, we can start to classify images from the test set.

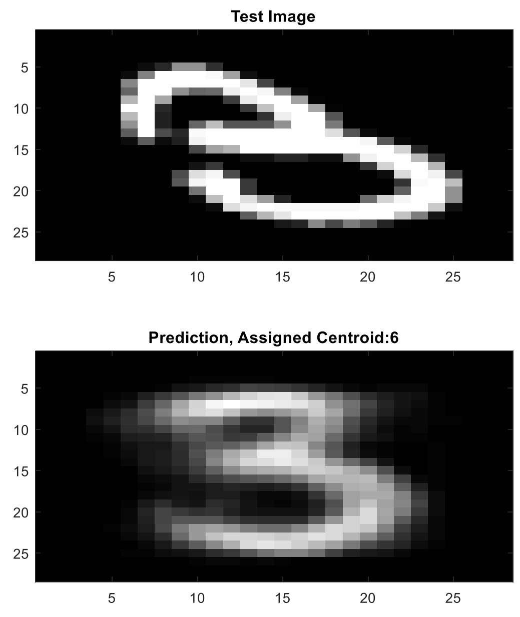

Figure 3. Here, a test image is assigned to Centroid 6 (see centroids above in Figure 2).

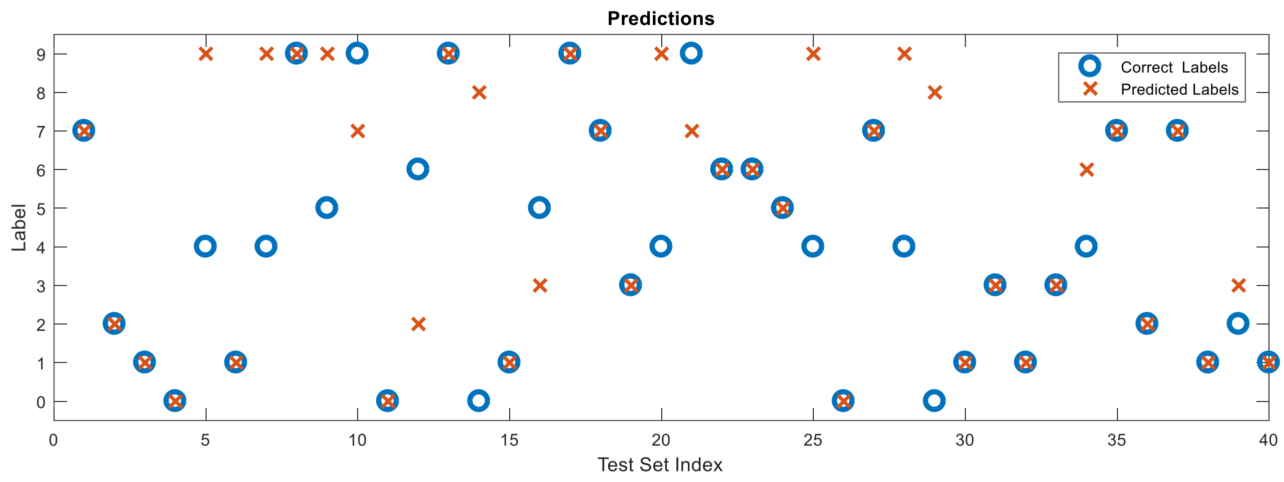

Figure 4. Correct and Predicted Labels for a 40 image test set. When the X and O overlay, that test image has been classified correctly. Here, 26/40 are correctly classified. **Your test set contains 200 images.**

D. Training and Testing

An important aspect of classification is the separation between training and testing sets. To test an algorithm, you should use inputs/data that were not used in the training. For this case study, this means that you should establish your clusters using one set of data, then test the "performance" of these clusters using a different set of data.

We have provided a test set of 200 images (as documented in the skeleton file). We have also provided a 200 dimensional array that contains the correct labels (numerical meaning) for each of the test images (this is simply the last column of the test array).

E. Classifying the test set and finding outliers

The script 'cs1_mnist_evaluate_test_set.m' is intended to evaluate the performance of your classifier on the test set. You should build up this file in order to do the following:

1. Produce a figure similar to Figure 4. (Note that the file contains code that does portions of this already).
2. **Find outliers**. There are a small number of digits in the test set that are outliers. That is, there is something about the way these digits are encoded as vectors that makes them different than all others. This does not mean they have a different semantic meaning; i.e., they are still digits and still have labels.
   - ***a.*** ***Design*** a strategy to find outliers. Try and leverage your k-means infrastructure to facilitate finding these outliers.
   - ***b.*** To indicate the outliers, use the vector 'outliers' that is introduced in the script. This ith entry of this vector should contain a flag of '1' if the ith test digit is an outlier.
   - ***c.*** Produce a plot, similar to Figure 5, below, indicating which digits are outliers.
   - ***d.*** In your writeup, describe your design.
   - ***e.*** If you believe you have identified the distinguishing feature of the outliers, you may try and counteract this within your k-means classifier (so that the classifier is able to correctly classify the outliers). Feel free to experiment with this in the 'cs1_mnist_evaluate_test_set.m'. The code you turn in will be used to evaluate performance in the competition phase.

F. Competition scoring

The 'cs1_mnist_evaluate_test_set.m' code written by each group will be used to classify a set of competition images. Performance will be ranked based upon the accuracy of assigning the correct digit label to each competition image, labeling the competition image as an outlier, or both.

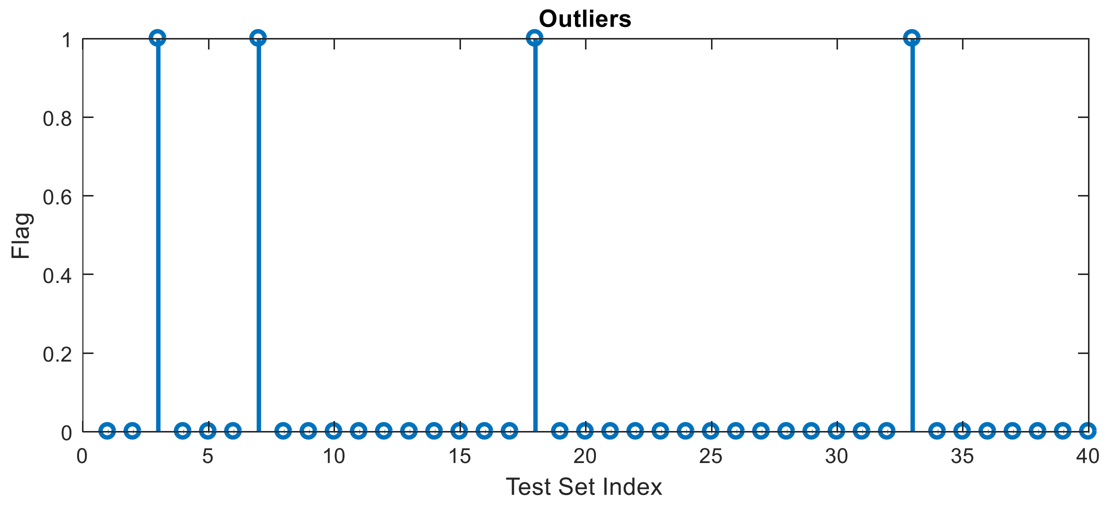

Figure 5. Example of plot depicting outliers for a 40 image test set. Here, images 3, 7, 18 and 33 are outliers. **Your test set contains 200 images.** You must design a strategy to find and flag the outliers, then make a plot like this.

The final score will be computed as:

$$Score = (\#correct) + 50R_{TP} + 50R_{TN} - 0.5(\#centroids)$$

The components of this score are:

(*#correct*): The number of digits correctly classified

R_{TP}: The true positive rate of outliers, i.e., the number of correctly identified outliers divided by the number of actual outliers.

R_{TP}: The true negative rate of outliers, i.e., the number of correctly identified non-outliers divided by the number of actual non-outliers.

(*#centroids*): The number of centroids used in your classifier

G. Tips

1. The skeleton file reads the images into an array that is n x 785. The 785th column of this array is where the cluster label is stored. For example, if you are clustering into 5 clusters, then this column would store either 1,2,3,4 or 5 based on cluster/centroid to which that row is assigned.
2. The provided skeleton file shows how to reshape a 784 vector into a 28 x 28 array that can be then plotted as an image (as above).
3. Loops are going to be very useful throughout this case study.
4. The skeleton file for clustering and classification are separate. This is deliberate. Each time you run the clustering file, you will generate new centroids. Having the classification file separate will allow you to "save" your centroids from one run and compare them to another. This will also facilitate the competition phase of the case study.
5. In MATLAB, functions should either appear at the very end of your script, or in an entirely separate file.
6. You are free to choose k and the number of iterations of the algorithm as you please!

H. What to Turn In

1. Your built-up skeleton file, which should perform k-means on the training set and produce a plot of k centroids (similar to Figure 2) and a plot of how the k-means cost reduces with iterations (similar to Figure 1). Use the matlab `publish` command to prepare a pdf for submission (similar to what you have done on MATLAB homeworks), in addition to your code.

2. Your built-up file 'cs1_mnist_evaluate_test_set.m', which should output the 200-dimensional vectors "predictions" and "outliers". 'predictions' contains the predicted labels for each test digit. 'vectors' contains the flags (i.e., an entry of 1) indicating which digits are outliers. Use the matlab `publish` command to prepare a pdf for submission (similar to what you have done on MATLAB homeworks), in addition to your code.

3. Any other code that you used.

4. A MAT file named ‘classifierdata.mat’ that contains two arrays:
   a. k x 785 array named ‘centroids’ containing the k centroids you wish to use for the competition phase.
   b. k x 1 array named ‘centroid_labels’ containing the labels (numerical meanings) of the above k centroids.

5. A 1-2 page report illustrating the results of classifying your own handwritten digits. **Use the Word template provided with the case study materials.** This should contain figures similar to Figure 5, above. Present the results of classifying individual differences in a format of your choosing. Make sure that your report clearly states/presents:
   a. the rationale for the design
   b. quantitative evaluation of the performance of the design
   c. conclusions about the design based on the results presented in the report.

6. A signed honor code document, as distributed with the case study materials.

I. Rubric

1. Correctness of MATLAB code (40%)
   a. Success rate of classification on the test data and competition data
   b. Success rate of outlier detection
2. Presentation (20%)
   a. Plots and Figures should be clear.
      i. Make sure axis numbers are large and legible.
      ii. Make sure lines are legible.
   b. Written report should be well-organized and clearly written
3. Programming style (20%)
4. Study design (20%)
   a. Motivation and justification for your outlier detection strategy