CSE-575 Statistical Machine Learning **Name:** *Prabin Kumar Rath (prath4@asu.edu)*
Homework 3

1. The MNIST dataset has N_{train}=60000 images for training and N_{test}=10000 images for testing. Each image has 28x28 pixels and can be represented with a flattened vector of length D=784. Computing the K-NN accuracy for the entire test set for any specific K is in the order of $O(N_{train} * N_{test} * D)$. Hence, it becomes computationally expensive to use K-NN on the MNIST dataset with a brute force implementation. Hence, in this homework, I have used the following techniques to make the computation tractable for evaluating the K-NN algorithm.

   - Principal Component Analysis (PCA) has been used to reduce the dimension of the data from 784 to just 50 principal features.
   - Evaluation is done on a subset of N_{test}=100 test images which are chosen randomly from the original 10000 samples.

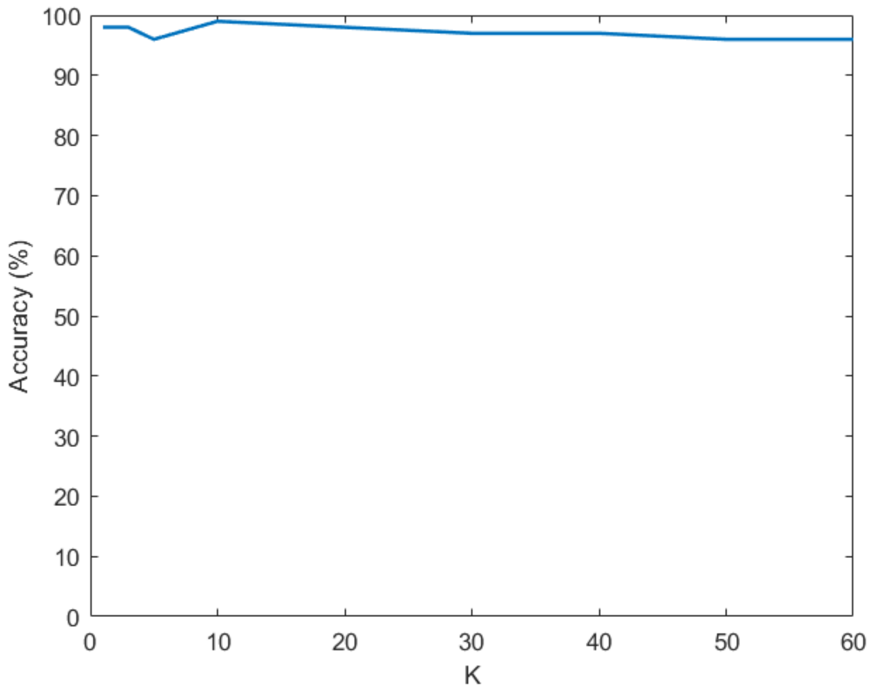

Figure 1: Accuracy of K-NN on the MNIST dataset using 50 principal components

After the experiments, the following are my observations from the MATLAB implementation of K-NN. Accuracy plot with respect to K is shown in Figure 1

- On an average the accuracy of prediction stays close to 98% with a maximum of 99% observed for K=10.
- Higher K is is not always preferable. It was observed that lower K values often resulted in better accuracy. K can be considered as a hyper parameter that needs to be tuned for difference datasets.
- K-NN is too expensive for real-time inference systems. Practical implementations would often require dimension reduction techniques.
- Only 50 principal components were able to capture enough information for predicting labels with high accuracy. Hence, it can be concluded that images in the MNIST dataset have redundant features which can be safely dropped using dimension reduction techniques.

Code file for this question is $q1\_knn.m$. I have also provided $q1\_knn.mlx$ live script file for easy evaluation. There is an $extra$ code file for this question that uses pre-computed euclidean distances for faster inference on the test set.

2. K-Means clustering with K=2 converged within 6 iterations. Cost plot with respect to iterations is shown in Figure 2

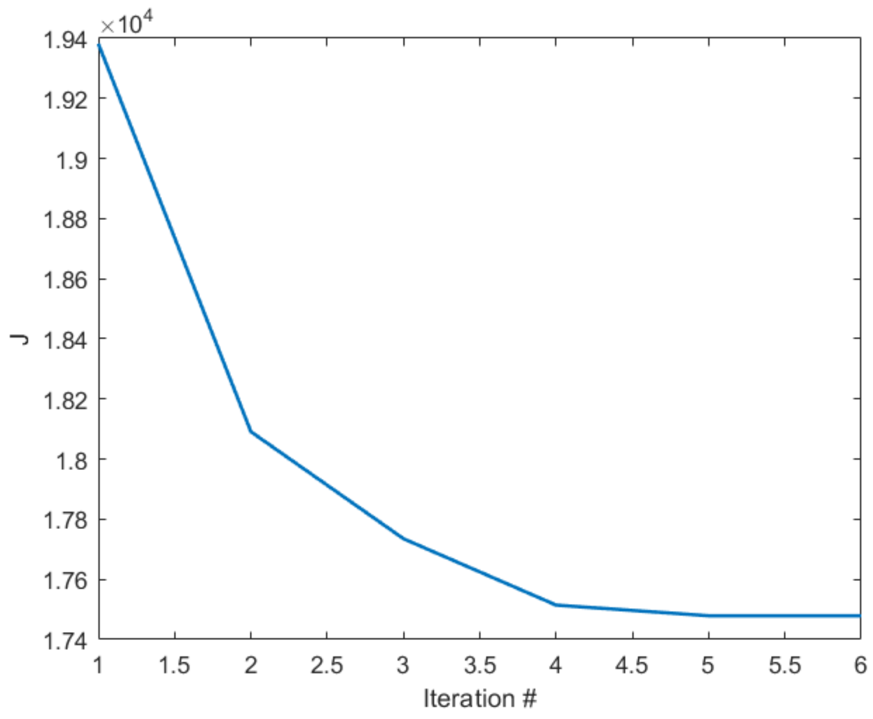

Figure 2: Cost of K-Means clustering algorithm for K=2

Result for 2-Means clustering is shown in Figure 3 using the first 2 features of the data.

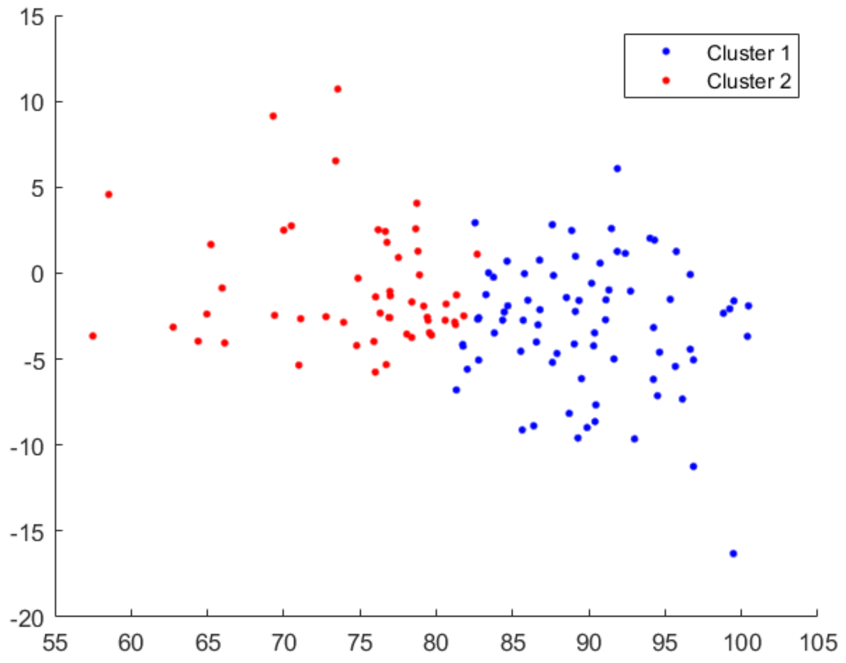

Figure 3: Results of 2-Means clustering. Blue represents Class 1 and Red represents Class 2.

After the experiments, the following are my observations from the MATLAB implementation of K-Means.

- For smaller datasets, K-Means algorithm converges very quickly.
- Cost of assignment depends on the dataset and will always reduce with successive iterations of the algorithm.

Code file for this question is $q2\_kmeans.m$. I have also provided $q2\_kmeans.mlx$ live script file for easy evaluation.

3. Gaussian Mixture Model (GMM) utilizes a continuous mixing coefficients for K different Gaussian distributions to cluster data points within a dataset. Results for K=2 is shown in Figure 4 using first 2 features of the data. The resultant gaussian mixture for the first two features is shown as a mesh plot in Figure 5.

   After the experiments, the following are my observations from the MATLAB implementation of GMM.

   - GMM is significantly faster then K-Means.
   - It is very sensitive to the initial assumptions for the class means and variances. Hence, it was required to pass the cluster estimates of K-Means as an initial guess for GMM.
   - Bad initial guess results in imbalanced cluster assignments.

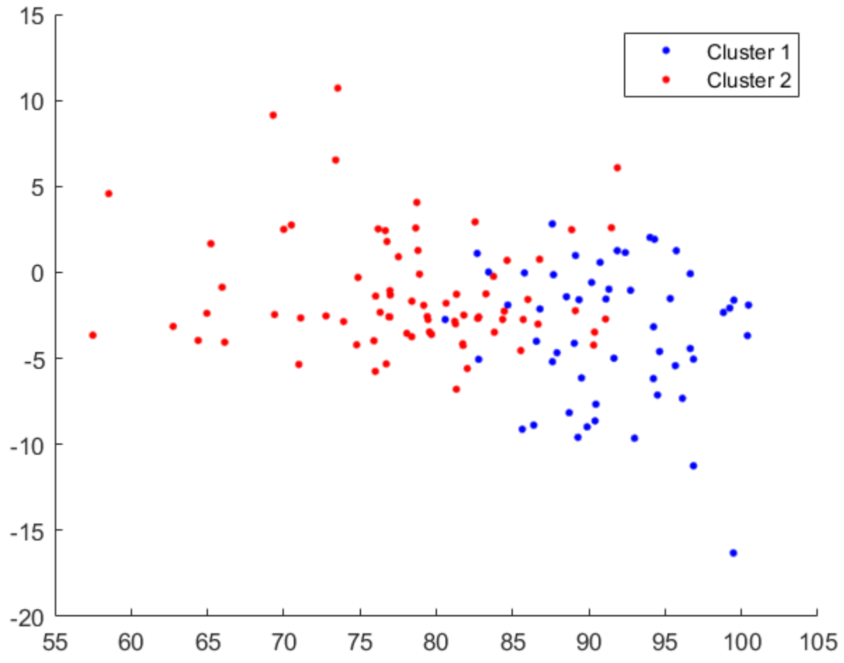

Figure 4: Results of GMM with K=2. Blue represents Class 1 and Red represents Class 2.

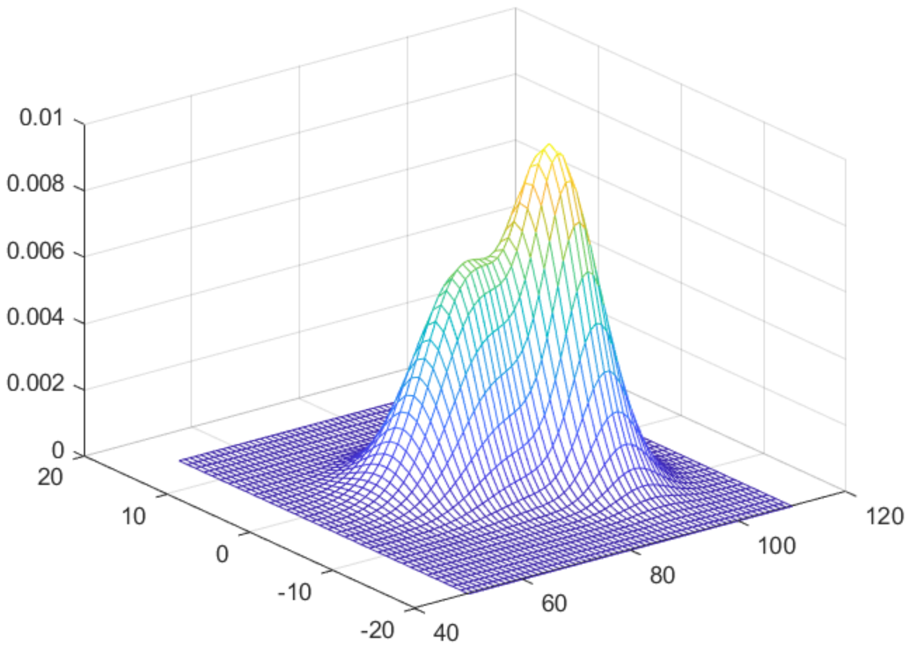

Figure 5: Gaussian Mixtures for the first 2 features after convergence of the EM algorithm.

Code file for this question is $q1\_gmm.m$. I have also provided $q1\_gmm.mlx$ live script file for easy evaluation. The dataset provided for this assignment is 13D and hence it is difficult to visualize the correctness of the implementation. There is an $extra$ code file for this question that uses a custom 2D dataset for checking the correctness of implementation.