CS 584: THEORY AND APPLICATIONS OF DATA MINING
HW4: K-MEANS CLUSTERING PART-1 AND PART-2

NAME : SRIPATH CHERUKURI
MINER2 USER-ID : MAVERICK
RANK PART-1 : 277
MINER SCORE PART-1 : 0.82
RANK PART-2 : 87
MINER SCORE PART-2 : 0.77
G-NO : G01395231

PROBLEM STATEMENT
The main objective of this assignment is to develop k-means clustering from scratch for the Iris data set in part one and test our k-means clustering algorithm and prepare the algorithm for the second part where we have to deal with processed vector data of 10,000 images of handwritten digits from (zero to nine) and predict the digit.

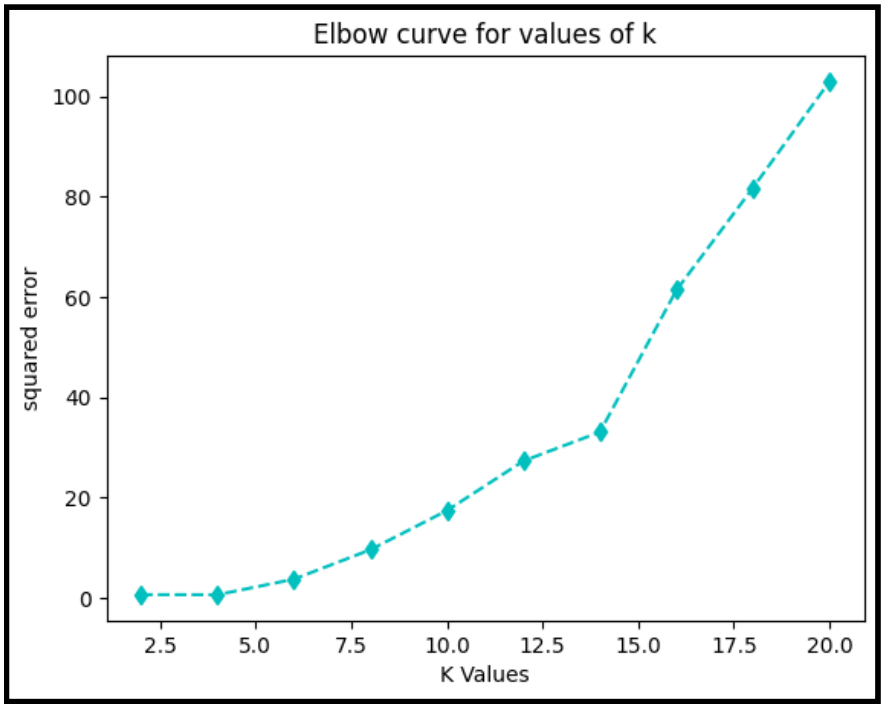
K-MEANS CLUSTERING ALGORITHM
The means algorithm is an algorithm which repeats itself until the data points with similarities are grouped under a single cluster containing similar data points. The means algorithm can be divided into steps as below:
1. Define the number of clusters we need (k value).
2. Now arbitrarily choose some k points in the data as centroids.
3. Form random clusters from the data
4. Compute euclidean distance for data points in cluster and all centroids.
5. Update and assign nearest centroids based on distance obtained above.
6. Calculate the centroids by computing the mean of all data points in a cluster.
7. Repeat the steps until there is no change in the centroids.

PART - 1
In the part one, the data given had 150 samples and 4 features. The dataset is the famous iris data set which contains sepal length, sepal width, petal length, petal width. Therefore we have to pre-process data by performing dimensionality reduction and prepare it for the algorithm.
Parameters:
1. **K value : 3.** In part one the default k value considered is 3 as it is already given in objective.

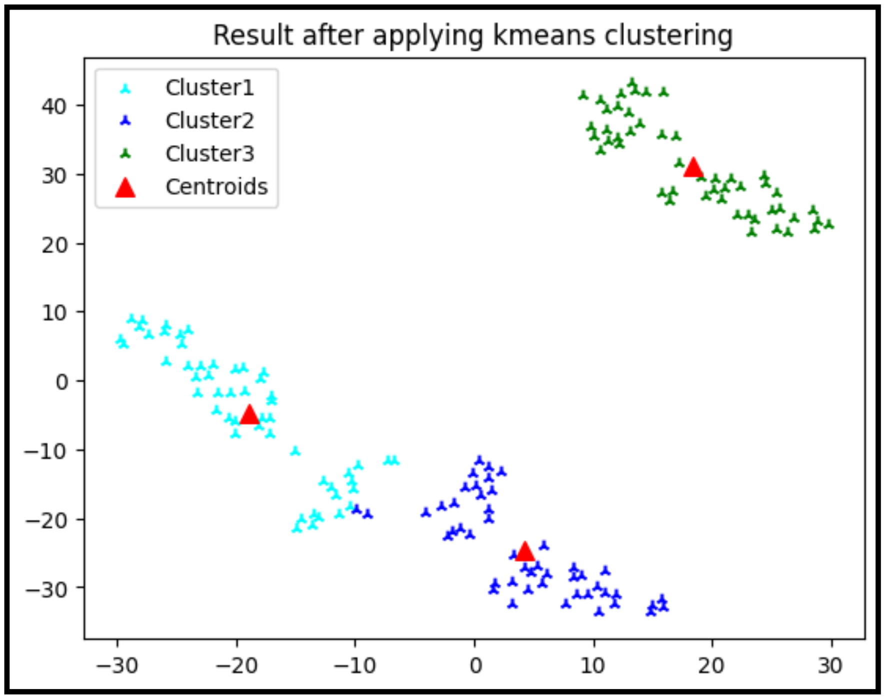
2. **Dimensionality reduction : (n_components value = 2).** The dimensionality reduction is done using TSNE (T-distributed stochastic neighbor embedding) which is good for non-linear data. The data is reduced from 4 features to 2 features for easier execution and visualization. Other dimensionality reduction techniques such as PCA, UMAP were also tried but TSNE proved to be the best for this dataset.

3. **Maximum iterations : 50.** The maximum number of iterations set here is 50, tried changing the number but the output doesn't seem to change after a certain point.

Also plotted the elbow graph for values of k from 2 to 20 with a step value of 2 to find optimal value of k. The plot can be seen below:



The resulting plot after applying kmeans on the iris data can be seen below:



PART - 2

In the part one, the data given had 10,000 samples and 784 features. The dataset is in the form of processed vectors which contains pixel data of images in integer format. Therefore we have to pre-process data by performing dimensionality reduction and prepare it for the algorithm.

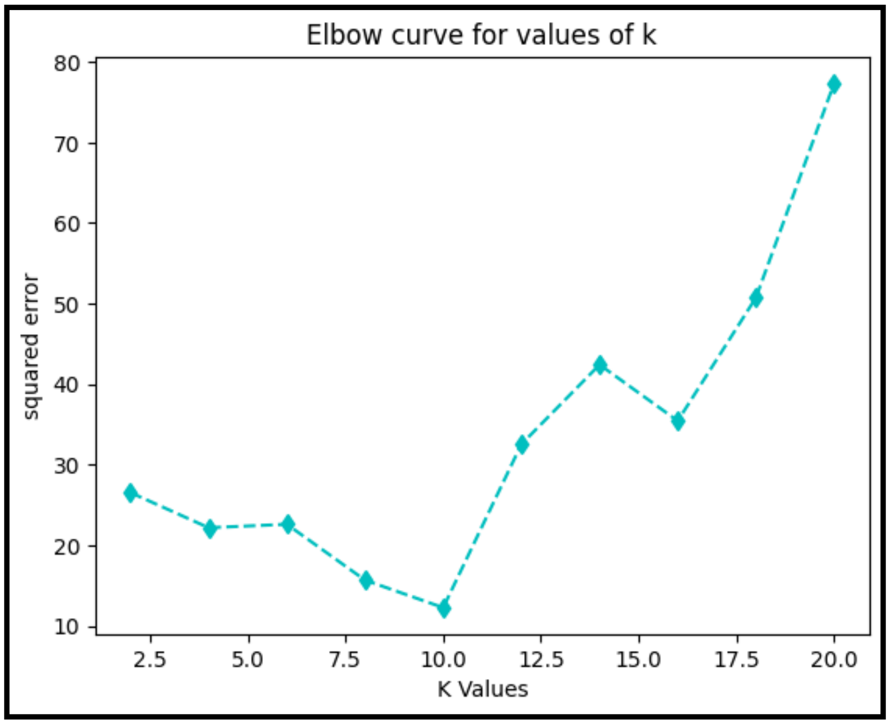
Parameters:

1. **K value : 10.** In part one the default k value considered is 10 as it is already given in objective.

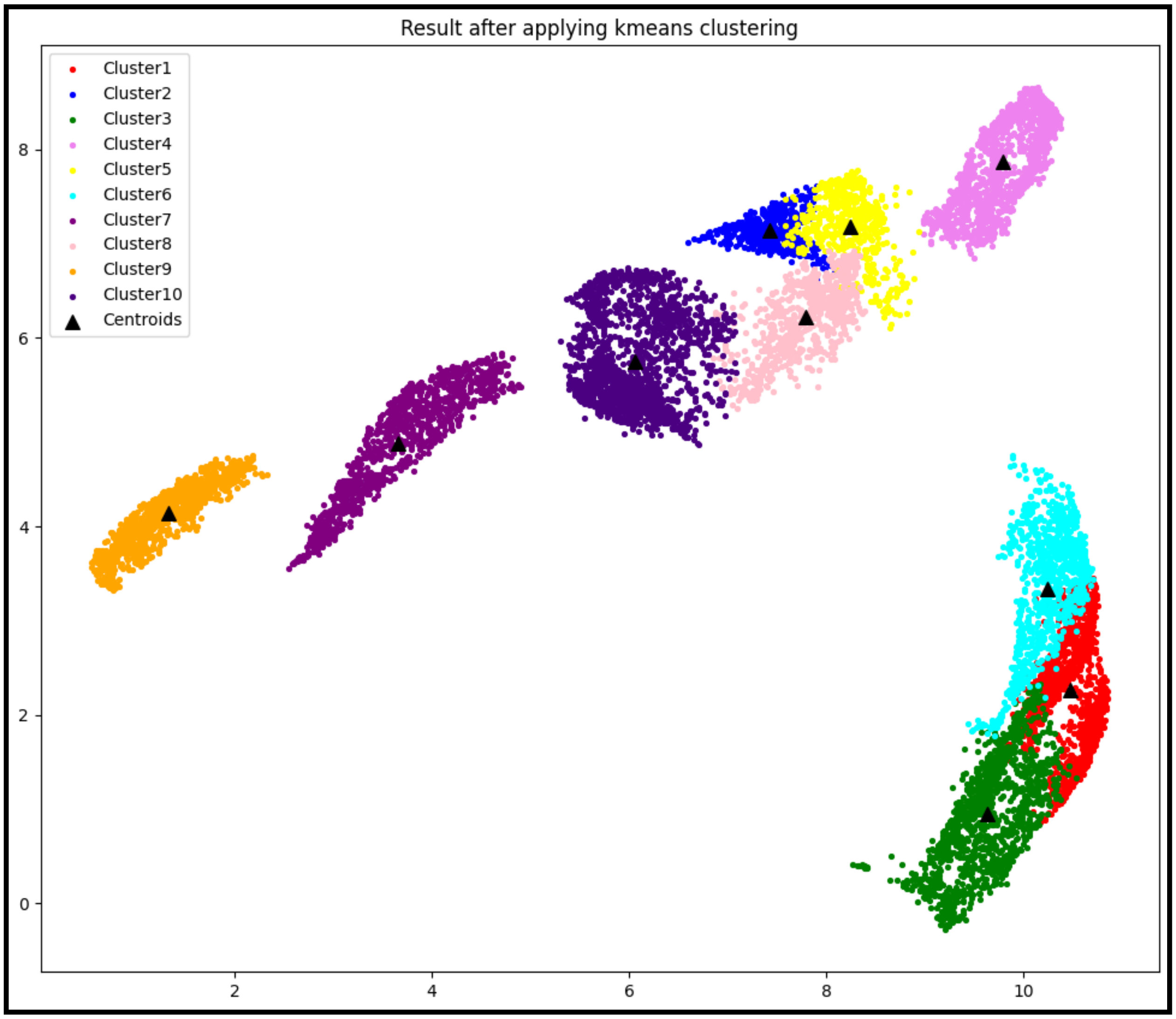
2. **Dimensionality reduction : (n_components value = 600).** The dimensionality reduction is done using UMAP (uniform manifold approximation and projection for dimension reduction) which is good for non-linear data. The data is reduced from features to 784 features to 600 for easier execution and visualization. Other dimensionality reduction techniques such as PCA, TSNE were also tried but UMAP proved to be the best for this dataset in second part. The number of features were tested with different numbers by trial and error. The optimal value was found at n_components = 600.

3. **Maximum iterations : 150.** The maximum number of iterations set here is 150, tried changing the number, but the output doesn't seem to change after a certain point. Also this being a huge dataset increasing number of iterations increases the execution time.

Also for part2 plotted the elbow graph for values of k from 2 to 20 with a step value of 2 to find optimal value of k. The plot can be seen below:



The resulting plot after applying kmeans on the image data can be seen below for part2:



CONCLUSION

The image dataset being the large, there's still scope for betterment as kmeans depends too much on the initially taken centroids. If the initially taken centroids are too far away from data points they are left out. We can improve more by using kmeans++ algorithm where initial centroids are taken more efficiently than in kmeans.