Manmeet Singh Khanna
CS 584
Homework 3

HW 3 (584-Rangwala): K-Means Clustering

User Name: Deviant
Accuracy Score-Iris: 84%

Accuracy Score-Image: 71%

Approach:

The assignment included two steps, which were:

1. Preprocessing Data
2. Using K-Means Clustering

Preprocessing Data:

The data in part1 and 2, was directly saved as a csv file and then imported to Jupyter Notebooks. The Iris dataset contains four features which are sepal's length and width and the petal's length and width. It has 150 samples. The image clustering data contained Each row is a record (image), which contains 784 comma-delimited integers with shape 10000x784. At first I checked if there were any null values in the dataset, but there were none in both the datasets. I have made use of the pandas.isnull().sum().

The data is then normalised. In the IRIS dataset, the data is normalised in values between 0 and 1. In the Image dataset, the data is divided by 255 as there were too many 0's which give a NAN value if it is normalised between 0 and 1.

I had a closer look at the data set where there were many columns with only zero values in the image dataset but not in the Iris dataset. I removed all columns with only zeros as that could help reduce some features and help improve accuracy of the prediction. This reduced the features from 784 to 668. After this step I scaled the data and converted it into a Numpy array,

TSNE is used for dimensionality reduction as this is a very high-dimensional data. It is based on Stochastic Neignbour Embedding. It ia a type of unsupervised non-linear learning technique for dimensionality reduction and data visualisation technique. Here, the similar pairs are assigned a high probability and dissimilar pairs are assigned a low probability. Thus t-SNE converts the data into two dimensions.
TSNE(n_components=2, verbose=1, perplexity=30, n_iter=300)

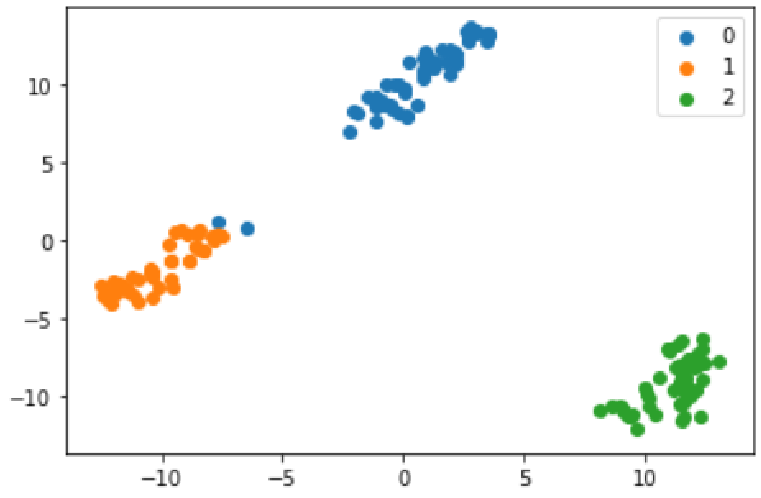
N_components: Dimension of the embedded space.
Perplexity: Related number of neighbors used in the manifold algorithm. For this dataset I've tried with 30, 35 and 40. (Currently 30 works best).
N_iter: Maximum number of iterations for the optimization. Should be at least 250. For the sake
of convenience, I've used 300.

Initialize Random Centroids and Assigning the clusters to the data points: To initial random centroids, I've used np.random_choice to select 3 random indices from the dataset and used them as centroids. After that I've clustered the data based on the distance of each point to the
centroid. To compute the distance I've used numpy's np.linalg.norm which is basically Euclidian
distance. This functionality is implemented in Cluster_Assignmet function.
It takes 3 parameters

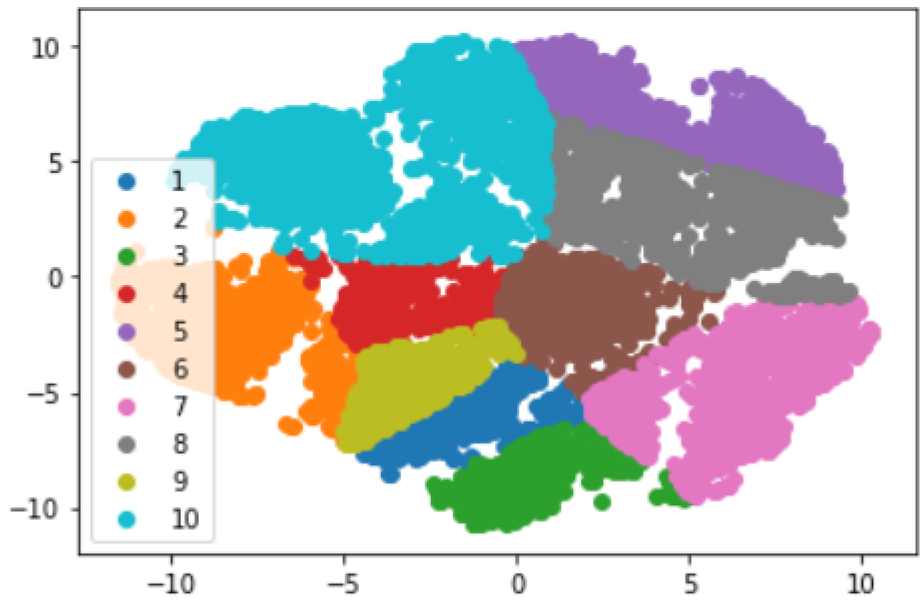
1) dataset
2) Current centroids and K

After computing the initial clusters, I compute new centroids based on the clusters.
Take the average of the all datapoints in the cluster. This average will be new centroids.
Returns the previous centroids and the current centroids. This function is implemented in Move_Centroid function. It takes 6 parameters
data: n dimensional Numpy array
Centroids: K-dimensional Numpy array
K: Default value is 3(for iris dataset and 2 for image dataset), splits the data into k clusters
Attributes: Number of Features in the dataset
Instances: # rows in the dataset
cluster_assignments: The current cluster assignment of each data point in the dataset.
If the centroids are changed this whole process of assigning clusters and recomputing the centroids based on the mean value of the clusters continues until all the centroids do not change or the silhouette score is less than 0.25
To compute the silhouette score, I've used the silhouette_score from sklearn.metrics
Silhouette score is calculated using the mean intra_distance of the cluster and the mean nearest cluster distance. This function returns the mean score of all samples. A score of 1 is the
best and -1 is the worst.
The silhouette_score takes 2 parameters 1) dataset (N-dimensional numpy array) 2) Labels of
the clusters.

This is the graph for IRIS dataset. It contains three clusters. Clearly, they are assigned clusters properly.



This is the graphical representation for the Image Clustering. Clearly, it wasn't clustered that well but has defined points.



References:

https://scikit-learn.org/
[pandas - Python Data Analysis Library (pydata.org)]()
[Natural Language Toolkit — NLTK 3.6.2 documentation]()
https://automaticaddison.com/k-means-clustering-algorithm-from-scratch-machine-learning/