CSE 575 – STATISTICAL MACHINE LEARNING

Class # 18194 – Spring 2021 Instructor **– Nupur Thakur**

PROJECT Part – 2

Unsupervised Learning (K-means Clustering)

Report by - **Subba Raja Kashyap Saligrama** **(Asu Id:** 1219510851**)**

Dataset used:

The dataset given is a set of 2-D co-ordinate points. It is provided in form of a single mat file called 'AllSamples.mat'.
Samples in the dataset: **300** 2-D points with *x-coordinate* and *y-coordinate* values.

Algorithms Implemented:

- K-Means clustering with two strategies for initialization –
  - Selecting 'k' random cluster centers from points in the given dataset
  - Selecting one random cluster center, then choosing other centers with maximal average distance from previous ones

Programming Language, workspace, software used:

- Python is used to implement the source code of K-Means clustering from the scratch.
- Python 3.9 environment is used to execute the program.
- 'Scipy' library is used to import and read the mat file as the dataset is in form of a dictionary.
- 'Numpy' library is used to perform mathematical operations on the datasets.
- 'Matplotlib' library used to plot the graphs.
- 'Random' library to select random cluster centers from the given set of points.

Importing and Reading Dataset:

Initially, we first load the data from the mat file with help of 'Scipy' library into a numpy array *inpData* and find its shape as – $\left(\mathbf{300}\ (no.of\ points)\ \ \text{x}\ \ \mathbf{2}\ (x, y\ values)\right)$

Implementing K-Means Clustering:

Input : 'n' data samples, 'k' value – no. of clusters
Goal : Partition the samples into 'k' cluster sets D_i $(1 \le i \le k)$ with respective means/centers $\mu_1, \mu_2, \dots, \mu_k$ such that – the sum of squared error is minimized.
sum of squared error, $\boldsymbol{J_e} = \sum_{i=1}^{k} \sum_{\mathbf{x} \in D_i} \|\boldsymbol{x} - \boldsymbol{\mu_i}\|^2$ *where* $\boldsymbol{\mu_i} = \frac{1}{n_i} \sum_{\mathbf{x} \in D_i} \mathbf{x}$

Basic K-Means Algorithm –

```
Given: n samples, a number k.
Begin
    Initialize μ1, μ2,…, μk cluster centers randomly
    Do
        Classify n samples according to nearest μi
        Recompute μi
    Until no change in μi
    Return μ1, μ2,…, μk
End
```

There are four steps here –

1. **Initialization** - Initialize the cluster centers (Strategy 1 or Strategy 2).
   - Strategy 1 : Pick 'k' random cluster centers from points in the given dataset.
   - Strategy 2 : From points in the given dataset, pick first cluster center randomly, then choose other *(k-1)* centers such that – the sample point with maximal average distance from previous *(i-1)* centers becomes the i^{th} *(i>1)* center.

- Random selections are done using **`random.sample()`** method on given samples dataset.

2. **Assignment** - Assign all samples to their nearest clusters based on geometric proximity.
   - Euclidean norm is used to calculate the geometric proximity between the sample points and the cluster centers.

```
for each sample X in dataset {
   distanceToAllCenters = []
   for each center μ in the cluster centers {
      dist = Euclidean distance(X,μ)
       append dist to distanceToAllCenters
   }
   assign X to cluster Di with center μi = argmin_μ distanceToAllCenters
}
```

3. **Updation** - Recompute/Update the cluster centers (centroids) based on mean of the samples under that cluster.

$$\mu_i = \frac{1}{|D_i|} \sum_{x \in D_i} x \qquad for\ \ i = 1\ \ to\ \ k$$

4. **Repeat** steps 2 & 3 until convergence (cluster centers no longer update).

Calculating Objective function:

The objective function for K-Means is nothing but the Sum-of-Squared-Errors term –

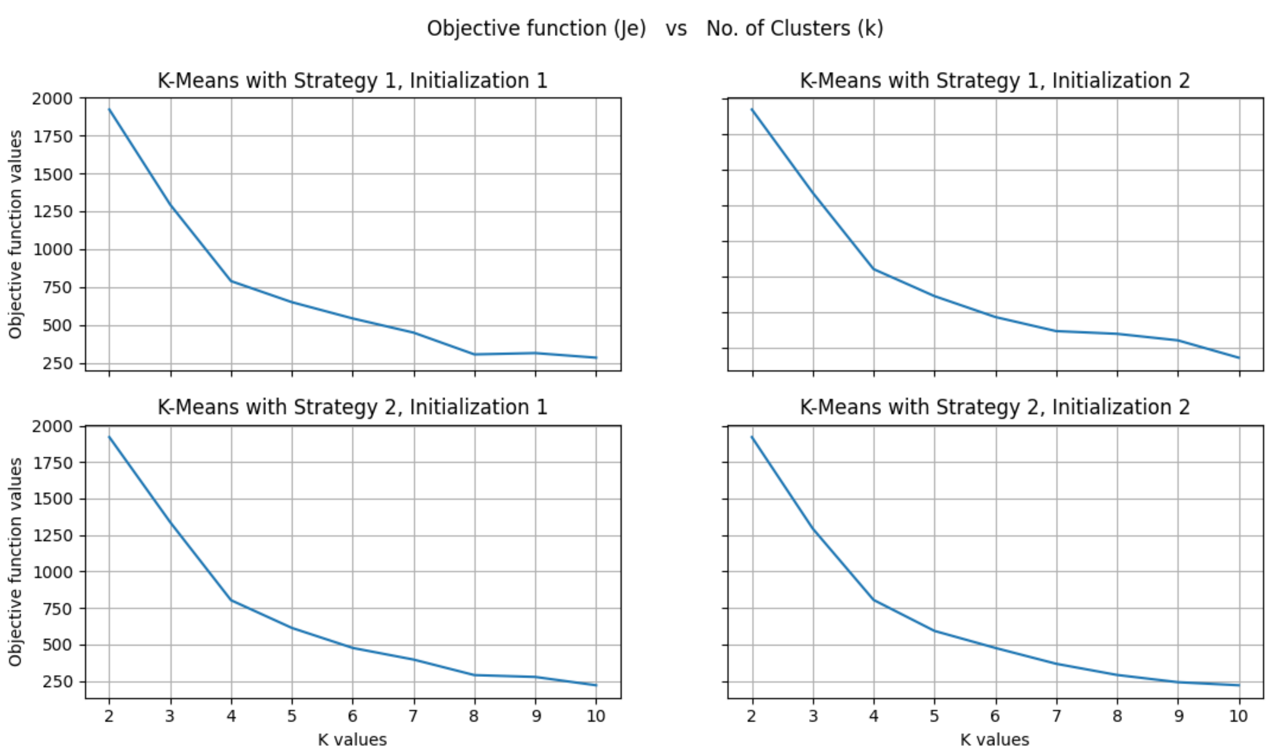
$$\boldsymbol{J_e} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{k}} \sum_{\mathbf{x} \in \boldsymbol{D_i}} \|\boldsymbol{x} - \boldsymbol{\mu_i}\|^2$$

Our objective is to find a $\boldsymbol{k}$ value and the cluster sets $\{\, \boldsymbol{D_i}, 1 \le i \le k \,\}$ such that $\boldsymbol{J_e}$ is minimized.

Also, we know that K-Means algorithm is sensitive to initialization of cluster centers. So, we –

- Implement two strategies of cluster centers initialization.
- Try two iterations for each strategy so that initializations will vary.
- Compute the objective function $\boldsymbol{J_e}$ values for different $\boldsymbol{k}$ values in each iteration of each strategy.
- Then, compare the $\boldsymbol{J_e}$ values by plotting them against $\boldsymbol{k}$ values and
- Find which strategy and which $\boldsymbol{k}$ value gives the best clustering of samples (best minimizes $\boldsymbol{J_e}$).

Plotting Objective function vs No. of clusters:



Observations from the above plots:

From the two graphs related to '**K-Means with Strategy 1**' –

- As '*k'* value (number of clusters) increases, the objective function value decreases most of the time (abruptly at some '*k*' values).
- As all the initial cluster centers are chosen completely randomly, there might not be correlation between two initializations.
- Different initializations of cluster centers can significantly vary the final resultant clusters and thereby the objective function value for the same '*k*' value. Hence, it becomes difficult to choose some '*k*' value that guarantees stable resultant clusters.
- After a certain '*k*' value, the objective function value doesn't vary significantly. Here, such points can be observed at '*k*' value = 4 and 8.

From the two graphs related to '**K-Means with Strategy 2**' –

- As '*k'* value (number of clusters) increases, the objective function value decreases but not abruptly most of the time.
- As the randomness in choosing the initial cluster centers is reduced to just the first center, the similarity between two initializations may become evident as the '*k*' value increases.
- Different initializations of cluster centers don't significantly vary the final resultant clusters and thereby the objective function value for the same '*k*' value. Hence, a '*k*' value that gives reasonably small objective function value can be selected irrespective of the initial cluster centers.
- So, even though there is more computation involved here compared to strategy 1 in choosing the initial cluster centers, this strategy is more probable to give stable resultant clusters.
- After a certain '*k*' value, the objective function value doesn't vary significantly. Here, such points can be observed at '*k*' value = 4 and 8.