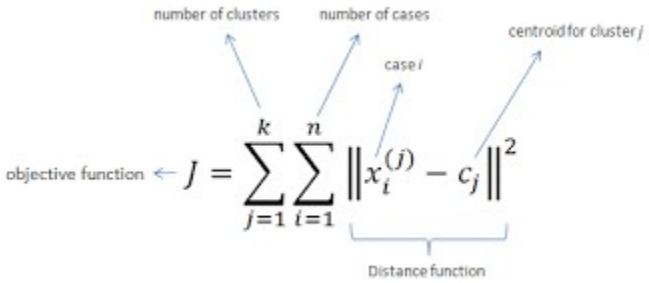
# Project: 2
# Unsupervised Learning (K-means)
# Name- Kunal Kumar
# ASU ID- 1217167445

## Introduction:

In this project we will be implementing a supervised learning method(K-means) to classify data. We are given the data set which contains a set of 2-D points. K-means clustering is used when we have unlabeled data (i.e. data without defined categories or groups). The goal of the K-means algorithm is to find the groups in the data, with the number of groups represented by the variable K. The results of the K-means clustering algorithm are:

1. The centroid of the K cluster, which can be used to label new data.
2. Labels for the training data (each data point is assigned to a single cluster)

The objective function for k-means is given as follows:




Our goal is to implement the K-means algorithm using 2 different strategies for choosing initial clusters centers and then calculating the objective function for them and then plot the respective graphs.

**Strategy:1**

Pick the initial cluster from the given samples randomly.

**Goal:**

Implement Strategy:1 and calculate and plot objective function.

**Strategy:2**

Select the first cluster randomly, and then choose the i-th cluster such that the distance between the i-th cluster and all the other (i-1)-th cluster is maximal.

**Goal:**

Implement Strategy:2 and calculate and plot objective function.

## Strategy:1

- **Initialize Centroid Randomly:**
  Centroid are selected at random and assigned from the given dataset. For different values of k, there will be k different clusters.

```
# Initializing the centroid randomly
def centroid_init(k, data_sets):
    randomly = np.random.choice(dat.shape[0], k, replace=False)
     centroids = data_sets[randomly]
```

- **Data Extraction:**
  "AllSamples.mat" is a set of 2-D dataset from which data is extracted for the learning algorithm.

```
# Importing the data from the "AllSamples" file for the learning algorithm
def dataext():
    dataset = scipy.io.loadmat("C:\\Users\\kunal\\Desktop\\Statistical Machine
                                    Learning\\Project 2\\AllSamples.mat")
    data = dataset['AllSamples']
```

- **Objective Function:**
  Calculating the Objective function for the K-means algorithm

```
# Objective Function for the K-means Algorithm
def objFun(data_sets, centroids):
    objectiveval = []
    for r in data_sets:
        objectiveval.append(((np.linalg.norm((r - centroids), axis=0) *
                              *2)))
    return np.sum(objectiveval)
```

- **Euclidean Distance:**
  Calculating the euclidean distance between 2 points

```
# Calculating the Euclidean Distance from the given Coordinates
def dist_euc(x_cord, y_cord, x_cent, y_cent):
    x_new = (x_cent - x_cord) ** 2
    y_new = (y_cent - y_cord) ** 2
    disteuc = math.sqrt(x_new + y_new)
    return disteuc
```

- **K-means Algorithm:**
  1-Loop k for k = 2,3,4,5,6,7,8,9,10:
  2- Initialize centroid
  3- Loop until the centroids have converges i.e. no change to the centroid
  4- Loop for all the data sets:
     a. Calculate the Euclidean-Distance of the data set and the centroid
     b. Select the minimum distance calculated and assign to the particular centroid

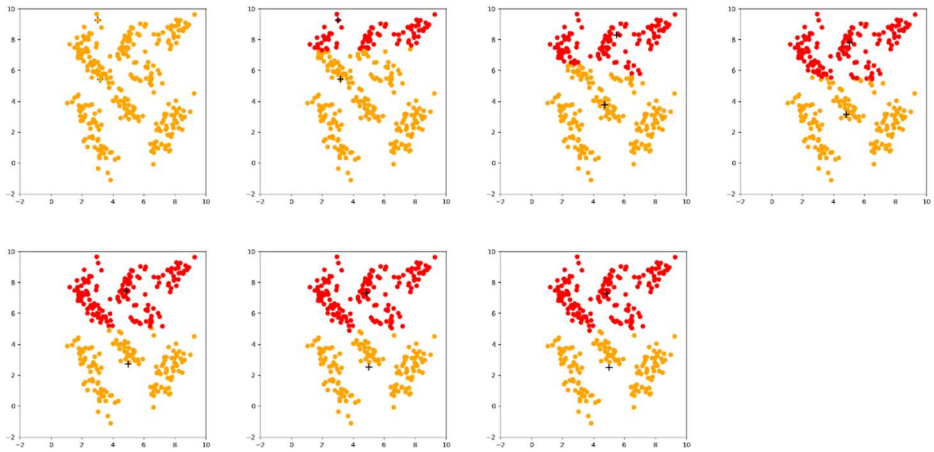
c. Calculate the mean of the points under cluster of centroids to find new centroid
d. Compare both the new centroid and the old Centroid for convergence

5- Loop Ends

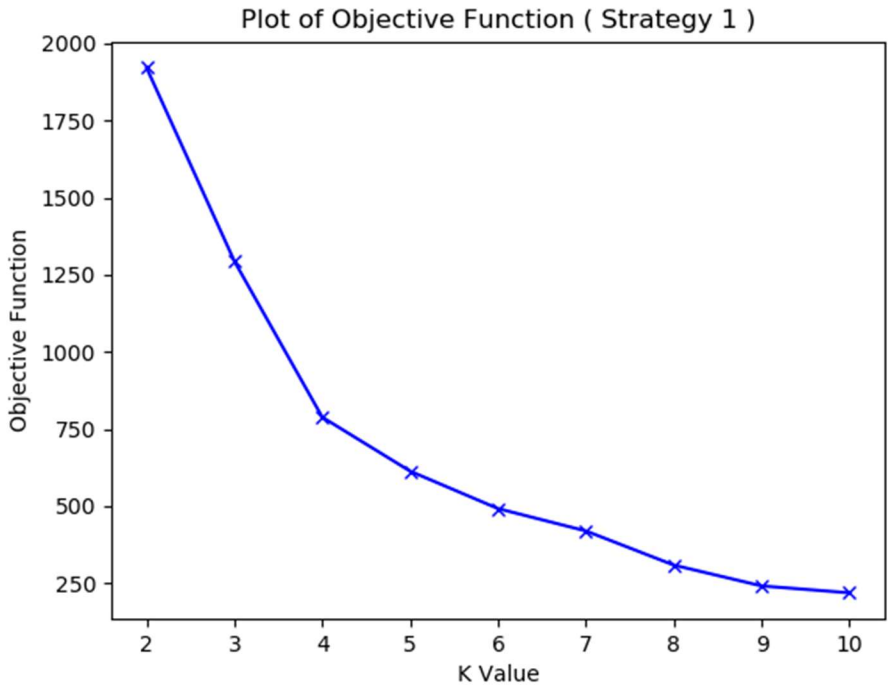
6- Loop Ends

7-Loop Ends

- **Plot for K=2**



- **Plot of Objective Function ( Strategy 1 ):**



## Strategy: 2

- **Data Extraction:**
  "AllSamples.mat" is a set of 2-D dataset from which data is extracted for the learning algorithm.

```
# Importing the data from the "AllSamples" file for the learning algorithm
def dataext():
    dataset = scipy.io.loadmat("C:\\Users\\kunal\\Desktop\\Statistical Machine
                                Learning\\Project 2\\AllSamples.mat")
    data = dataset['AllSamples']
```

- **Initializing Centroids:**
  Initializing the first centroid randomly and then selecting other centroid whish are at maximum distance with other.

```
#Initializing randomly the First Centroid
def centroid_init(k, data_sets):
    index_list = []
    centroids = []
    temp = np.zeros([len(data_sets), k - 1])
    randomly = np.random.choice(data_sets.shape[0], 1, replace=False)
    index_list.append(randomly[0])
    centroids.append(data_sets[randomly])
    # Selecting Centroids which are at maximum distance from each other
    for i in range(k - 1):
        temp[:, i] = np.linalg.norm(centroids[i] - data_sets, axis=1)
        condition = True
        temp2 = np.mean(temp[:, :i + 1], axis=1)
        condition = True
        i = np.argmax(temp2)
        while (condition):
            if i in index_list:
                temp2[i] = 0
                i = np.argmax(temp2)
            else:
                condition = False
        centroids.append(np.asarray(data_sets[i]))
        index_list.append(i)
    centroidss = data_sets[index_list]
    return centroids
```

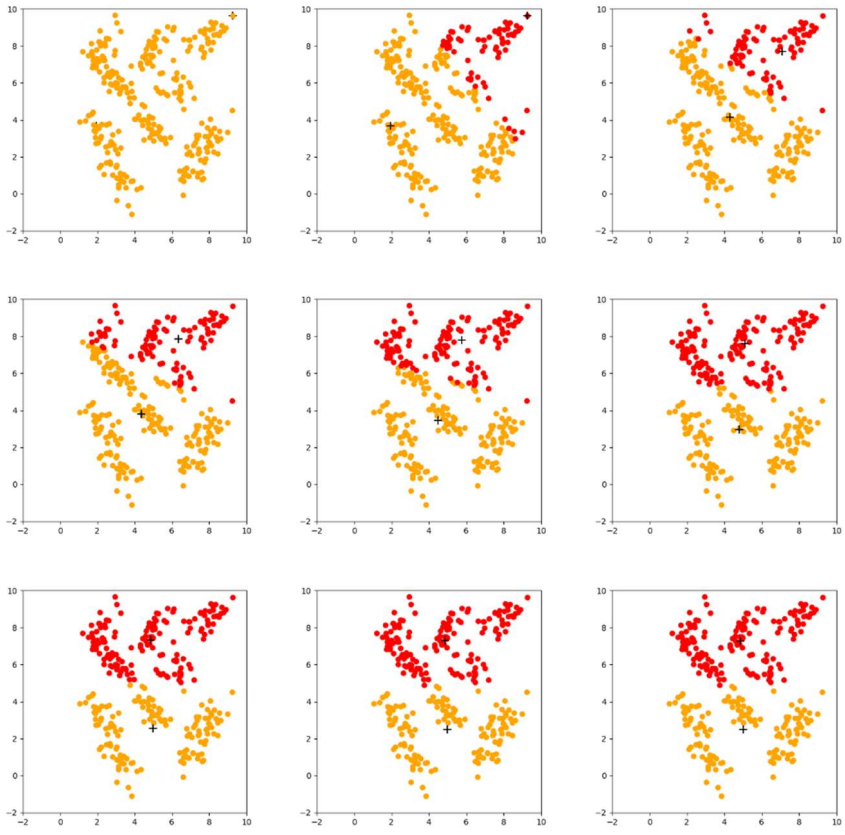
- **K-means Algorithm**

1-Loop k for k = 2,3,4,5,6,7,8,9,10:
2- Initialize centroid:
3- Loop for all the data sets:

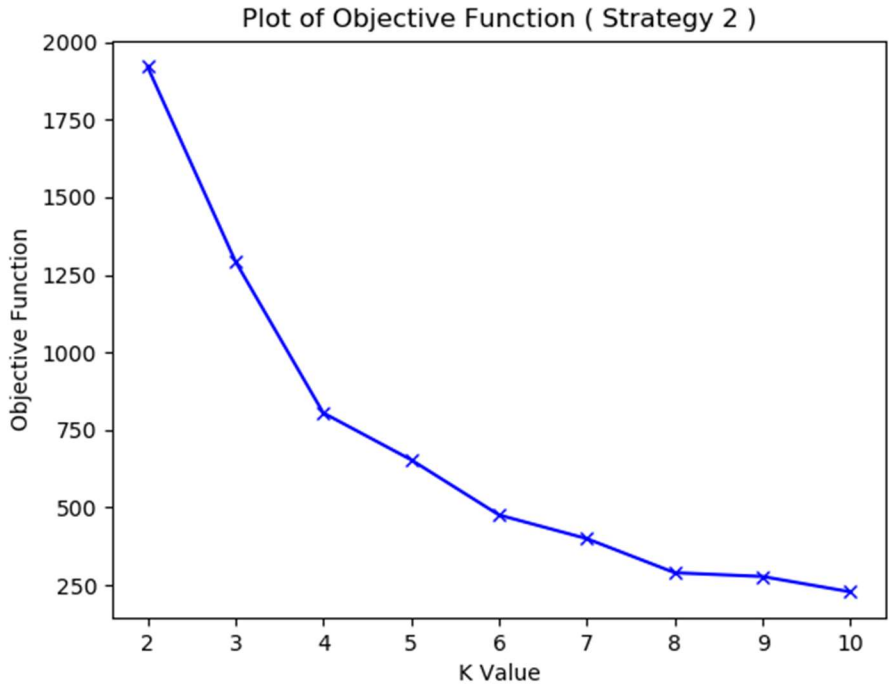
a. Calculate the Euclidean-Distance of the data set and the centroid
b. Select the minimum distance belonging to the particular centroid
c. Calculate the mean under cluster of centroids to project new centroids
d. Compare both the new centroid and the old Centroid for convergence

5- Loop Ends
6- End Loop
7-End Loop

- **Plot for K=2**



- **Plot of Objective Function ( Strategy 2 ):**




## Conclusion:

If we look on the plot of the K-value and the objective function, we can say that the value of objective function decreases if we increase the number of k. Also, after “k=4”, the objective function follows a linear fashion. So, we can say that the optimal number of cluster k=4.