K-Means Algorithm

K-Means algorithm is an unsupervised machine learning algorithm used for clustering. This algorithm has wide applications in market segmentation, social network analysis, astronomical data analysis etc. It can be realized in the following steps:

1) randomly choose K cluster centroids
2) classify the data based on their distance with the K cluster centroids
3) move the cluster centroids to the center of the cluster
4) repeat steps 2) and 3) until the cluster centroids won't change further

For example, we have a data.csv file (see **data.csv** file in this repository) which contains two columns, namely x1 and x2. All data are float values. We want to cluster this data. Here our optimization objective J is defined as the average Euclidean distance of the data points to their corresponding centroids. Mathematically, it is defined as follows:

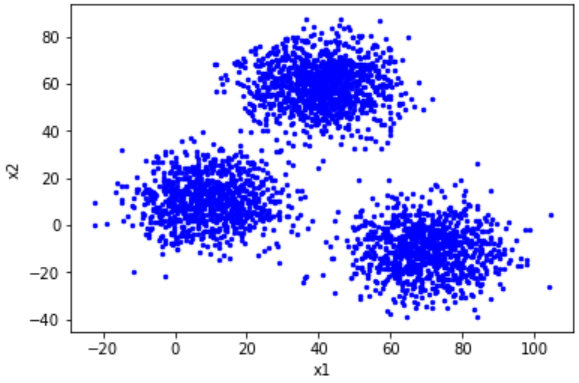
$$J = \frac{1}{m}\sum_{i=1}^{m}\left\| x^{(i)} - \mu_{c^{(i)}} \right\|^2$$

Here $x^{(i)}$ is the coordinates of the ith data point, $c^{(i)}$ is the label of centroid (0, 1, 2,… K-1) the ith data point corresponds to, $\mu_{c^{(i)}}$ is the coordinates of the centroid with label $c^{(i)}$. When the position of the centroids don't change any more, the J value will stop changing either. Thus, the optimization of the algorithm is realized.

Below is step-by-step realization of the K-Means algorithm. Let's start with visualizing the data in Python using the following code:

```
import pandas as pd
data = pd.read_csv('data.csv')
f1 = data['x1'].values
f2 = data['x2'].values
plt.scatter(f1, f2, c='black', s=6)
```

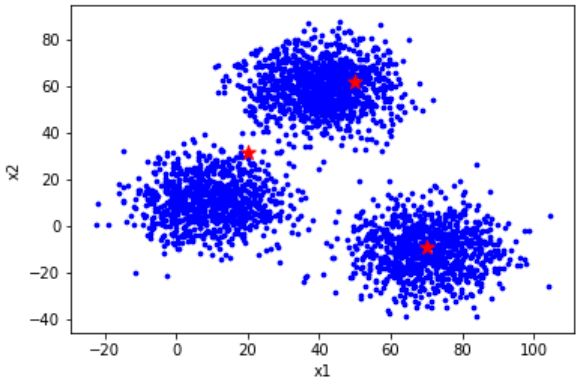
We get the following scattering plot:



1) The data visually forms three clusters, so let's start with three randomly selected centroids (K=3) using the following Python code:

```
Cx = np.random.random_sample(K)*(np.max(X)- np.min(X)) + np.min(X)
Cy = np.random.random_sample(K)*(np.max(X)- np.min(X)) + np.min(X)
C = np.array(list(zip(Cx, Cy)), dtype=np.float32)
plt.scatter(f1, f2, c = 'b', s = 6)
plt.scatter(Cx, Cy, marker='*', c='r', s = 100)
plt.xlabel('x1')
plt.ylabel('x2')
```

The three randomly selected centroids are labeled as red stars in the scattering plot shown below:



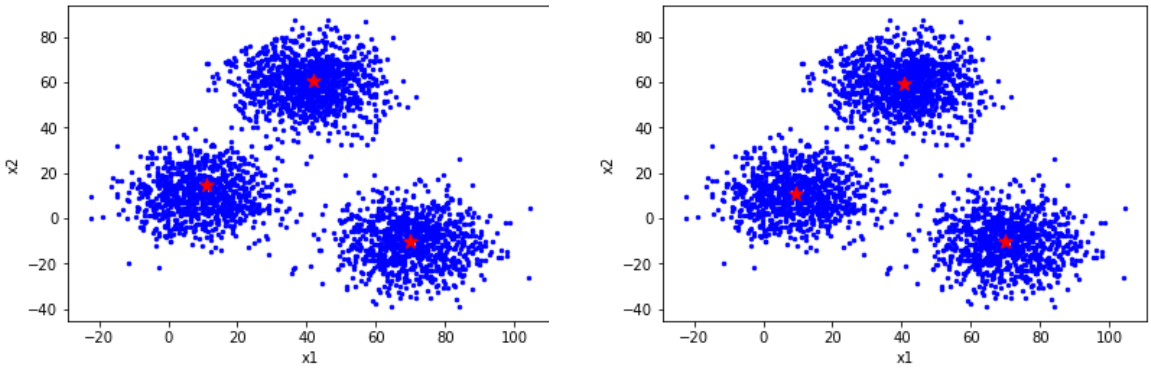
2) The next step is to classify the data based on their distance with the K cluster centroids and move the cluster centroids to the center of the clusters.
Let's first label each data point with their closest centroid number from 0 to K-1 using the following Python code:

```
idx = np.zeros(len(X))
for i in range(len(X)):
    dist = np.linalg.norm(X[i] - C, axis = 1)
    idx[i] = np.argmin(dist)
```

Next, we update the position of the K centroids based on the average position of the data points labeled with the corresponding centroid number using the following Python code:

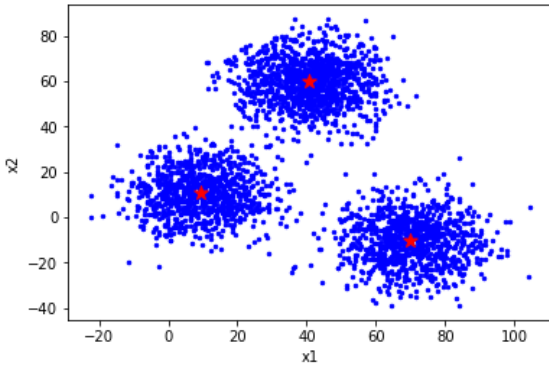
```
for i in range(K):
        points = [X[j] for j in range(m) if idx[j] == i]
        C[i] = np.mean(points, axis = 0)
plt.scatter(f1, f2, c = 'b', s = 6)
plt.scatter(C[:, 0], C[:, 1], c='r', s=100, marker='*')
plt.xlabel('x1')
plt.ylabel('x2')
```

If we run the above code multiple times, we will notice that the position of the centroids will change after each iteration as follows:

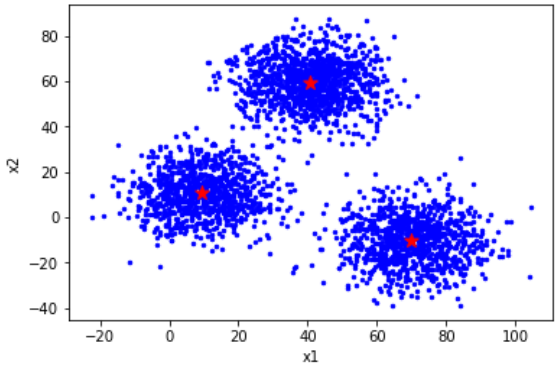


1st iteration

3rd iteration



2nd iteration



4th iteration

After the third iteration, the positions of the centroids stopped changing so that the convergence condition is met and the iteration stops. Thus, the input data was successfully clustered into three groups using K-Means algorithm.

Reference:
Machine Learning by Andrew Ng, from Coursera