Strategy1

September 18, 2022

[18]:
```
from Precode import *
import numpy

import pandas as pd
from sklearn.cluster import KMeans

data = np.load('AllSamples.npy')
```

[19]:
```
k1,i_point1,k2,i_point2 = initial_S1('8816') # please replace 0111 with your␣
↪last four digit of your ID
```

```
Strategy 1: k and initial points
```

[20]:
```
print(k1)
print(i_point1)
print(k2)
print(i_point2)
```

```
3
[[6.79251832 2.56208095]
 [7.56399709 7.83135288]
 [8.527899   8.55183237]]
5
[[6.39627447 1.24125663]
 [5.07250754 7.89834048]
 [6.79251832 2.56208095]
 [4.95728696 6.90897984]
 [7.85355511 2.53104656]]
```

[21]:
```
def randompick(samples,k):
    random=numpy.random.randint(len(samples),size=k)
    random_arr = data[random,:]
    return random_arr
```

[22]:
```
from scipy.spatial import distance
def assign_label(rand_arr, samples):
    labels= numpy.zeros(len(samples),dtype=float)
```

```
    dis=numpy.zeros(len(rand_arr))
    for i in range(0,len(samples)):
        for j in range(0,len(rand_arr)):
            dis[j]=distance.euclidean(rand_arr[j],samples[i])
        #print(dis)
        low=dis.min()
        #print(low)
        indx = numpy.where(dis==low)[0]
        labels[i]=indx
        #print(i)
    return labels
```

[23]:
```
def Kmeans(rand_arr,new_centroids,df,itr):
    #new_centroids = numpy.zeros((k1, 2))
    #print(new_centroids)

    #print(not(rand_arr==new_centroids).all())
    while not((rand_arr==new_centroids).all()):
        if itr!=0:
            rand_arr=new_centroids
        itr=itr+1
        label = assign_label(rand_arr,data)
        #print(label)
        data1 = numpy.column_stack((data, label))
        #print(data1)
        df = pd.DataFrame(data1)
        #print(df)
        df1=df.groupby([2]).mean()
        new_centroids= df1.to_numpy()
        #print('run once')
    return rand_arr,new_centroids,df,itr
```

[24]:
```
def objective_func(df,new_centroids):
    total_sum = 0
    df1= df[['X','Y']]
    data = df1.to_numpy()
    #print(data[0])
    #print(new_centroids[0])
    dist=0
    for i in range(0,len(df)):
        if(df['Cluster'][i]==0.0):
            dist = distance.euclidean(new_centroids[0],data[i])
        if(df['Cluster'][i]==1.0):
            dist = distance.euclidean(new_centroids[1],data[i])
```

```
        if(df['Cluster'][i]==2.0):
            dist = distance.euclidean(new_centroids[2],data[i])
        if(df['Cluster'][i]==3.0):
            dist = distance.euclidean(new_centroids[3],data[i])
        if(df['Cluster'][i]==4.0):
             dist = distance.euclidean(new_centroids[4],data[i])
        if(df['Cluster'][i]==5.0):
            dist = distance.euclidean(new_centroids[5],data[i])
        if(df['Cluster'][i]==6.0):
            dist = distance.euclidean(new_centroids[6],data[i])
        if(df['Cluster'][i]==7.0):
            dist = distance.euclidean(new_centroids[7],data[i])
        if(df['Cluster'][i]==8.0):
            dist = distance.euclidean(new_centroids[8],data[i])
        if(df['Cluster'][i]==9.0):
            dist = distance.euclidean(new_centroids[9],data[i])

        total_sum= total_sum + (dist*dist)

    return total_sum
```

[28]:
```
import pandas as pd
def plotobjecfunc(k):
    new_centroids = numpy.zeros((k, 2))
    df = pd.DataFrame()
    itr=0
    #rand_arr=i_point1
    rand_arr=randompick(data,k)
    #print(rand_arr)
    rand_arr,new_centroids,df,itr = Kmeans(rand_arr, new_centroids,df,itr)
    #print(rand_arr)
    print("The final centroids are:\n",new_centroids)
    #print(itr)
    df.columns = ['X', 'Y', 'Cluster']
    objective_sum = objective_func(df,new_centroids)
    return objective_sum

objective=numpy.zeros(9)

clusters=numpy.zeros(9, dtype=int)
for k in range(2,11):
    objective[k-2] = plotobjecfunc(k)
    clusters[k-2] = k

print(objective)
print(clusters)
```

```
print(type(objective))
print(type(clusters))



#print(df)
#df.groupby('Cluster').count()
#print(df['Cluster'][0])
```

```
The final centroids are:
 [[3.01682343 4.47741928]
 [6.80713674 5.40112426]]
The final centroids are:
 [[7.30031546 2.44948999]
 [3.29726377 2.60250684]
 [4.83375318 7.31605824]]
The final centroids are:
 [[3.39262114 6.8928815 ]
 [7.13560727 7.91651726]
 [7.22707673 2.52234361]
 [3.21257461 2.49658087]]
The final centroids are:
 [[7.75648325 8.55668928]
 [2.60123296 6.91610506]
 [5.40252508 6.73636175]
 [7.25262683 2.40015826]
 [3.21257461 2.49658087]]
The final centroids are:
 [[3.15179364 7.14121713]
 [7.42803426 2.28291867]
 [3.14506148 0.90770655]
 [7.47192641 8.55142802]
 [6.07873305 5.86993697]
 [3.49556658 3.56611232]]
The final centroids are:
 [[5.23053667 4.2793425 ]
 [2.54165252 7.00267832]
 [7.55616782 2.23516796]
 [7.91430998 8.51990981]
 [2.24204752 3.25100749]
 [3.16906145 0.81432515]
 [5.24028296 7.53131029]]
The final centroids are:
 [[4.91251497 3.56314096]
 [2.18321462 7.70355341]
 [4.98627902 7.79737812]
```

```
 [7.91430998 8.51990981]
 [2.76148421 1.93894294]
 [6.15468228 5.70140721]
 [7.55616782 2.23516796]
 [2.92352456 5.69062523]]
The final centroids are:
 [[2.00857179 3.54850646]
 [4.91181532 3.75514791]
 [3.44650803 0.47784504]
 [7.05668293 1.33319679]
 [2.56333815 6.9782248 ]
 [2.69805343 2.0242299 ]
 [7.95957401 3.08441042]
 [5.41183685 6.90440774]
 [7.75648325 8.55668928]]
The final centroids are:
 [[6.96223538 0.97764815]
 [2.56333815 6.9782248 ]
 [7.55538041 2.43703142]
 [3.3923978  0.50655764]
 [5.41183685 6.90440774]
 [1.90378175 3.66474083]
 [2.67243649 2.26123087]
 [7.75648325 8.55668928]
 [4.86460556 3.73249939]
 [8.10524433 3.61006161]]
[2498.11356032 1338.0878542   797.96018408  613.28243921  511.13758931
  362.93311405  326.11069069  299.84119268  287.69503025]
[ 2  3  4  5  6  7  8  9 10]
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
```

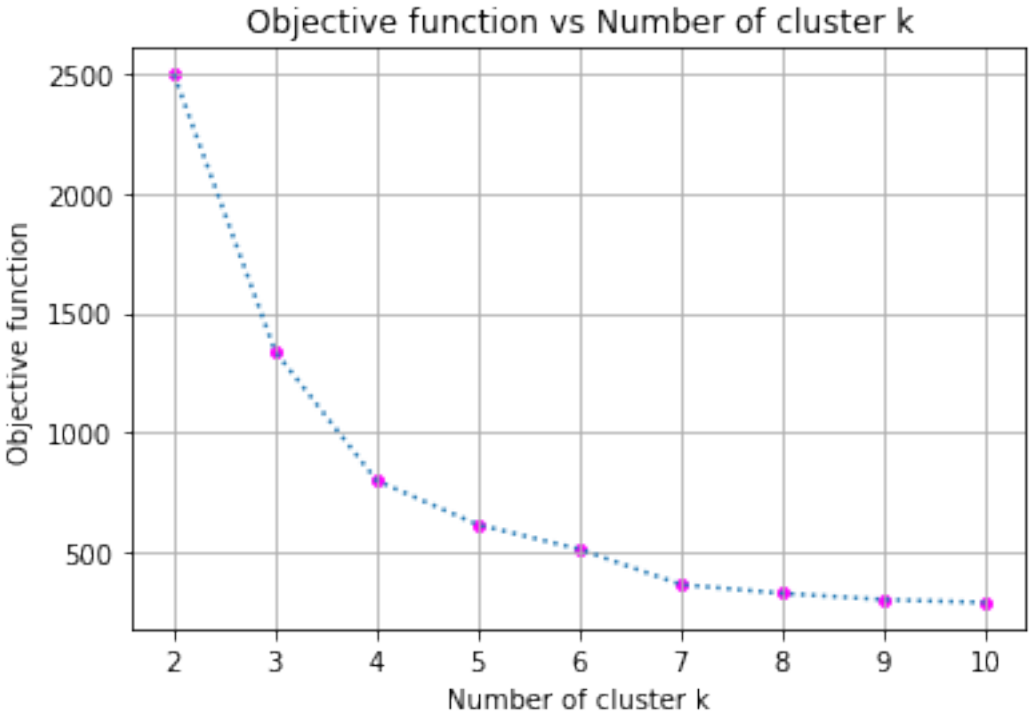
[29]:
```
df1 = pd.DataFrame(clusters, columns=['clusters'])
df2 = pd.DataFrame(objective, columns=['objective'])
df = pd.concat([df1, df2], axis=1)
print(df)

plt.scatter(df['clusters'], df['objective'],s=15,c="magenta")
plt.plot(df['clusters'], df['objective'], linestyle = 'dotted')
plt.grid()
plt.title('Objective function vs Number of cluster k')
plt.ylabel('Objective function')
plt.xlabel('Number of cluster k')


plt.show()
```

```
   clusters    objective
```

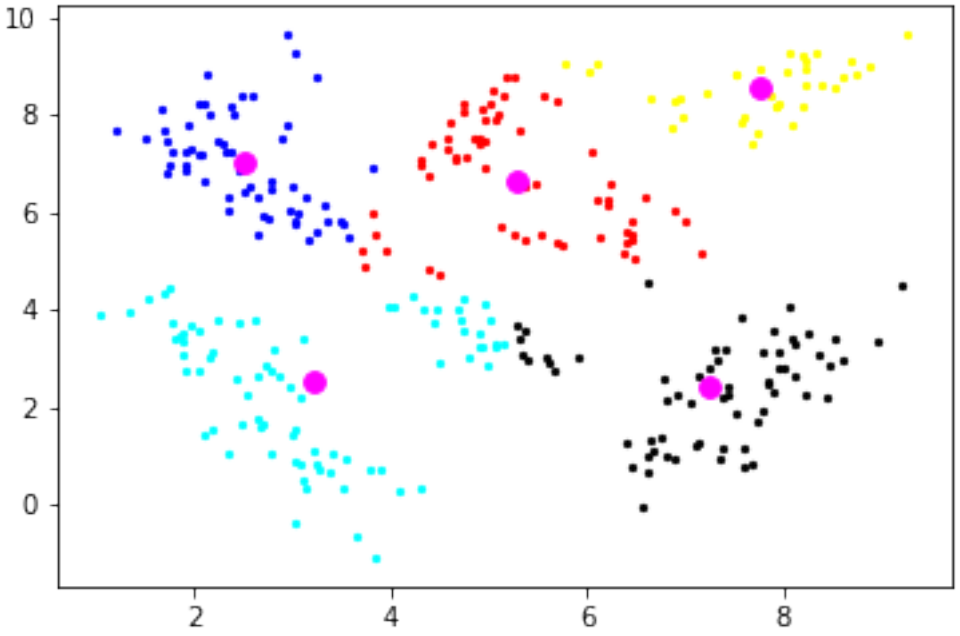
```
0          2  2498.113560
1          3  1338.087854
2          4   797.960184
3          5   613.282439
4          6   511.137589
5          7   362.933114
6          8   326.110691
7          9   299.841193
8         10   287.695030
```



[27]:
```
# used for quiz and understanding the Kmeans clustering

df_cluster0 = df[df.Cluster==0.0]
df_cluster1 = df[df.Cluster==1.0]
df_cluster2 = df[df.Cluster==2.0]
df_cluster3 = df[df.Cluster==3.0]
df_cluster4 = df[df.Cluster==4.0]
final_centroids_5 = pd.DataFrame(new_centroids)
final_centroids_5.columns = ['X','Y']
plt.scatter(df_cluster0['X'], df_cluster0['Y'],s=5,c="blue")
plt.scatter(df_cluster1['X'], df_cluster1['Y'],s=5,c="red")
plt.scatter(df_cluster2['X'], df_cluster2['Y'],s=5,c="black")
plt.scatter(df_cluster3['X'], df_cluster3['Y'],s=5,c="cyan")
plt.scatter(df_cluster4['X'], df_cluster4['Y'],s=5,c="yellow")
```

```
plt.scatter(final_centroids_5['X'], final_centroids_5['Y'],s=60,c="magenta")
plt.show()
```



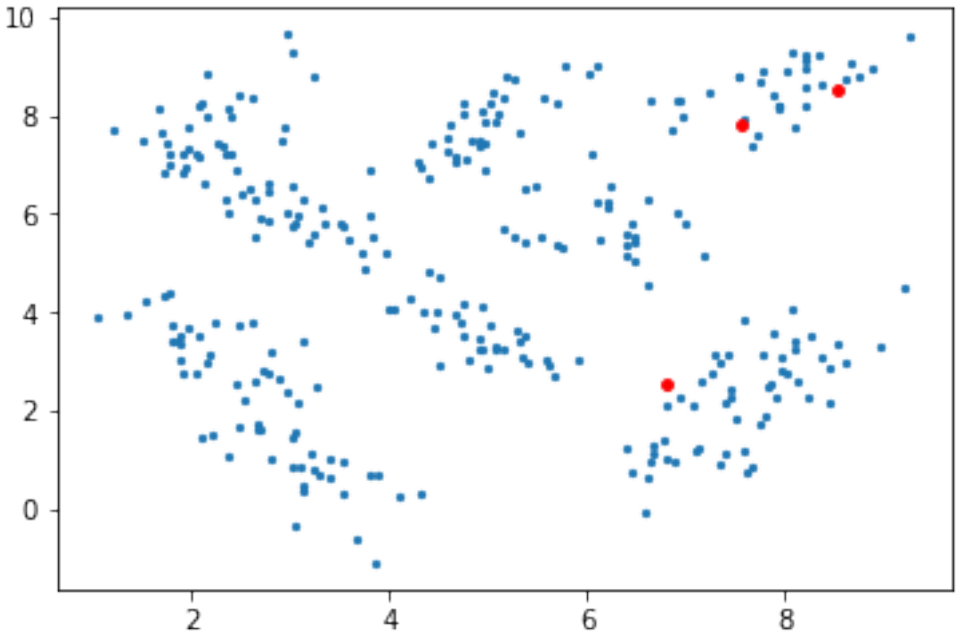
[54]:
```
#print(data)
# used for quiz and understanding the Kmeans clustering

df=pd.DataFrame(data)
dfi=pd.DataFrame(i_point1)
df.columns=["X","Y"]
dfi.columns=["X","Y"]
```

[55]:
```
# used for quiz and understanding the Kmeans clustering

#print(df)
#type(df)
plt.scatter(df['X'], df['Y'],s=5)
plt.scatter(dfi['X'], dfi['Y'],s=15,c="red")

plt.show()
```


10
8
6
4
2
0
2
4
6
8