# Strategy2

September 18, 2022

[50]:
```
from Precode2 import *
import numpy

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

[51]:
```
k1,i_point1,k2,i_point2 = initial_S2('8816') # please replace 0111 with your␣
↪last four digit of your ID
```

```
Strategy 2: k and initial points
```

[52]:
```
print(k1)
print(i_point1)
print(k2)
print(i_point2)
```

```
4
[3.35409838 5.79603723]
6
[4.95185958 4.11756694]
```

[53]:
```
def randompick(samples,k):
    random=numpy.random.randint(len(samples),size=k)
    random_arr = data[random,:]
    return random_arr
```

[54]:
```
i_point2=randompick(data,1)
#print(rand_arr)
#rand_arr = i_point1
#rand_arr = i_point2
print(i_point2)
#print(type(rand_arr))
```

```
[[6.2153903  6.26139225]]
```

[55]:
```
from scipy.spatial import distance
def assign_label(rand_arr, samples):
    labels= numpy.zeros(len(samples),dtype=int)
    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
```

[56]:
```
import pandas as pd

def find_othercentroids(rand_arr,k1,data):
    #print(type(data))
    #print(len(data))
    #i_point=randompick(data,1)
    i_point=i_point2
    rand_arr = numpy.zeros((k1,2))
    rand_arr1= numpy.zeros((k1,2))
    dist = numpy.zeros(len(data))
    #print(rand_arr)
    #print(len(data))
#     for m in range(0,len(data)):
#         print("1st printer",m," ",data[m])

    i=0
    rand_arr[i]=i_point
    #print("first centroid insert\n",rand_arr)
    index = numpy.where(data == rand_arr[i])
    #print("index\n",index[0])
    #print(data[index])
    data = numpy.delete(data, index[0], 0)
#     for m in range(0,len(data)):
#         print("2nd printer",m," ",data[m])
    #print(len(data))

    i=i+1

    for j in range(0,len(data)):
        dist[j]=distance.euclidean(rand_arr[0],data[j])
    max_dist_index= numpy.argmax(dist)
    #print(max_dist_index)

    rand_arr[i]=data[max_dist_index]
    #print(" bool val\n",data[38].all()!=rand_arr[1].all())
    #print(len(data))
```

```
    data = numpy.delete(data, max_dist_index, 0)
#     for m in range(0,len(data)):
#         print("3rd printer",m," ",data[m])
    #print(len(data))

    i=i+1
    #print("second centroid insert\n",rand_arr)

    #print(numpy.count_nonzero(rand_arr))

    while numpy.count_nonzero(rand_arr)!=k1*2:
        rand_arr1=rand_arr[~numpy.all(rand_arr == 0.0, axis=1)]
        #print(" arr 1 \n",rand_arr1)
        maxi = numpy.zeros(len(data))
        for k in range(0,len(data)):
            summation = 0
            for j in range(0,len(rand_arr1)):
#                 if(data[k].all()==rand_arr1[j].all()):
#                     print("data k \n",data[k] )
#                     print(" array \n",rand_arr1[j])
#                     break

                summation=summation + distance.euclidean(rand_arr1[j],data[k])
            avg = summation/ len(rand_arr1)
            maxi[k]= avg
#             if i==3:
#                 print("average" ,k," " ,maxi[k])
        max_index= numpy.argmax(maxi)
        #print(max_index)
#         if i==3:
#             print("max average"," " ,data[max_index])

        #print(i)
        rand_arr[i]=data[max_index]
        data = numpy.delete(data, max_index, 0)
#         for m in range(0,len(data)):
#             print("4th printer",m," ",data[m])
        #print(len(data))
        #print("insert ith \n",rand_arr)
        #print(i)
        i=i+1
        #print(i)

    #print("the initial centroids\n",rand_arr)
    return rand_arr
```

[58]:
```
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
```

[59]:
```
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
```

[60]:
```
objective=numpy.zeros(9)
clusters=numpy.zeros(9, dtype=int)
for k2 in range(2,11):

    rand_arr = numpy.zeros((k2,2))
    rand_arr1= numpy.zeros((k2,2))
    #print(rand_arr)
    #rand_arr[1]=[1.2,3.2]
    #print(rand_arr)
    #print(numpy.count_nonzero(rand_arr))



    rand_arr = find_othercentroids(rand_arr,k2,data)


    new_centroids = numpy.zeros((k2, 2))
    df = pd.DataFrame()
    itr=0
    rand_arr,new_centroids,df,itr = Kmeans(rand_arr, new_centroids,df,itr)
    #print(rand_arr)
    #print(new_centroids)
    #print(itr)
    df.columns = ['X', 'Y', 'Cluster']
    #print(df)
    df.groupby('Cluster').count()
    #print(df['Cluster'][0])

    objective_sum = objective_func(df,new_centroids)
    objective[k2-2] = objective_sum
    clusters[k2-2] = k2
print(objective)
print(clusters)
```

```
[1921.03348586 1293.77745239  803.21672381  613.98662861  469.10726238
  368.96063376  290.92433447  273.57309879  260.39547164]
[ 2  3  4  5  6  7  8  9 10]
```

[61]:
```
import matplotlib.pyplot as plt
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('Strategy 2:Objective function vs Number of cluster k')
plt.ylabel('Objective function')
plt.xlabel('Number of cluster k')
plt.show()
```

```
   clusters    objective
0         2  1921.033486
1         3  1293.777452
2         4   803.216724
3         5   613.986629
4         6   469.107262
5         7   368.960634
6         8   290.924334
7         9   273.573099
8        10   260.395472
```