Name: Tania Rajabally

Branch: Comps

Batch: C

Roll no: 43

UID:2017130047

Date: 11/09/2020

**Experiment 3**

**Aim:** To implementCure Algorithm (Clustering)

**Theory:**The CURE (Clustering Using Representatives) Algorithm is large scale clustering algorithm in the point assignment class which assumes Euclidean space. It does not assume anything about the shape of clusters; they need not be normally distributed, and can even have strange bends, S-shapes, or even rings.

Instead of representing clusters by their centroid, it uses a collection of representative points, as the name implies.

The CURE algorithm is divided into into phases:

* Initialization in CURE
* Completion of the CURE Algorithm

It is a hierarchical based clustering technique, that adopts a middle ground between the centroid based and the all-point extremes. Hierarchical clustering is a type of clustering, that starts with a single point cluster, and moves to merge with another cluster, until the desired number of clusters are formed.

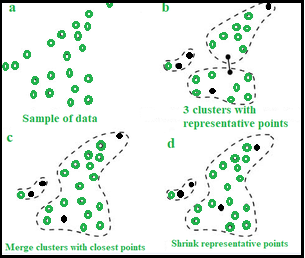
It is used for identifying the spherical and non-spherical clusters.

It is useful for discovering groups and identifying interesting distributions in the underlying data.

Instead of using one point centroid, as in most of data mining algorithms, CURE uses a set of well-defined representative points, for efficiently handling the clusters and eliminating the outliers.

Steps in Cure:

* Select target sample number ‘gfg’.
* Choose ‘gfg’ well scattered points in a cluster.
* These scattered points are shrunk towards centroid.
* These points are used as representatives of clusters and used in ‘Dmin’ cluster merging approach. In Dmin(distance minimum) cluster merging approach, the minimum distance from the scattered point inside the sample ‘gfg’ and the points outside ‘gfg sample, is calculated. The point having the least distance to the scattered point inside the sample, when compared to other points, is considered and merged into the sample.
* After every such merging, new sample points will be selected to represent the new cluster.
* Cluster merging will stop until target, say ‘k’ is reached.



**Code:**

import numpy as np

import sys

import datetime

from functools import cmp\_to\_key

from functools import reduce

def comparator(x, y):

if x[0] < y[0]:

return -1

if x[0] == y[0]:

if x[1] < y[1]:

return -1

if x[1] > y[1]:

return 1

return 0

return 1

samplefile = "sample\_data.txt"

fullfile = "full\_data.txt"

ratio = 0.2

list = []

n = 0

sum = 0

k = 3

start = datetime.datetime.now()

#hierachy clustering

with open(samplefile, "r") as file:

lines = file.readlines()

for line in lines:

value = line.strip().split(",")

list.append([[float(value[0]),float(value[1])], 1, [[float(value[0]),float(value[1])]]])

n += 1

num = 1

while (n != 3):

min\_coordinate = None

min\_distace = sys.maxsize

num += 1

for i in range(len(list)):

cluster1 = list[i]

centroid1 = np.array(cluster1[0]) / cluster1[1]

for j in range(len(list)):

if j > i:

cluster2 = list[j]

centroid2 = np.array(cluster2[0]) / cluster2[1]

distance = (centroid2 - centroid1) \*\* 2

if np.sum(distance) < min\_distace:

min\_coordinate = [i, j]

min\_distace = np.sum(distance)

cor1 = list[min\_coordinate[0]]

cor2 = list[min\_coordinate[1]]

new\_cluster = [np.ndarray.tolist(np.array(cor1[0]) + np.array(cor2[0])), cor1[1] + cor2[1], cor1[2] + cor2[2]]

list.remove(cor1)

list.remove(cor2)

list.append(new\_cluster)

n -= 1

centroid = []

# find representatives and calculate centroids

id = 0

re = []

for cluster in list:

sortedcluster = sorted(cluster[2], key = cmp\_to\_key(comparator))

repre = [sortedcluster[0]]

iter = 0

while (iter < n):

iter += 1

candi = None

min\_distace = sys.maxsize

if len(repre) == 1:

center = np.array(repre)

else:

center = reduce(np.add, np.array(repre)) / len(repre)

for coor in sortedcluster:

if coor not in repre:

dis = np.sum((center - np.array(coor)) \*\* 2)

if dis < min\_distace:

min\_distace = dis

candi = coor

repre.append(candi)

clustercentroid = reduce(np.add, np.array(cluster[2])) / len(cluster[2])

centroid.append(clustercentroid)

l = []

for representative in repre:

representative = np.ndarray.tolist(np.array(representative) + ratio \* (clustercentroid - np.array(representative)))

l.append(representative)

re.append(l)

# CURE

with open(fullfile, "r") as file:

lines = file.readlines()

points = map(lambda line: [float(x) for x in line.strip().split(",")],lines)

for point in points:

min\_dis = sys.maxsize

indexcluster = 0

for i in range(len(re)):

cluster = re[i]

# print "no:", i

for r in range(len(cluster)):

representative = cluster[r]

dis = np.sum((np.array(representative) - np.array(point)) \*\* 2)

if dis < min\_dis:

min\_dis = dis

indexcluster = i

list[i][2].append(point)

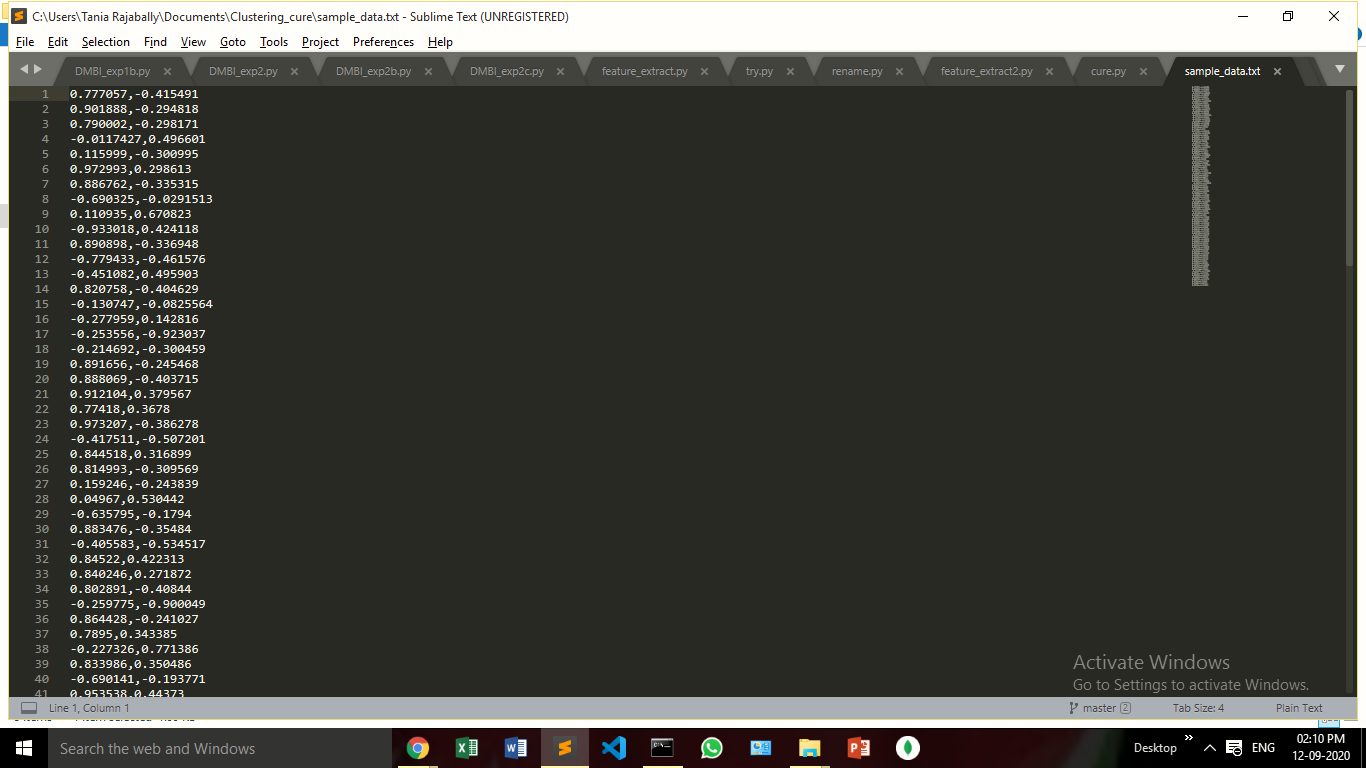
for x in range(len(list)):

print(len(list[x][2]))

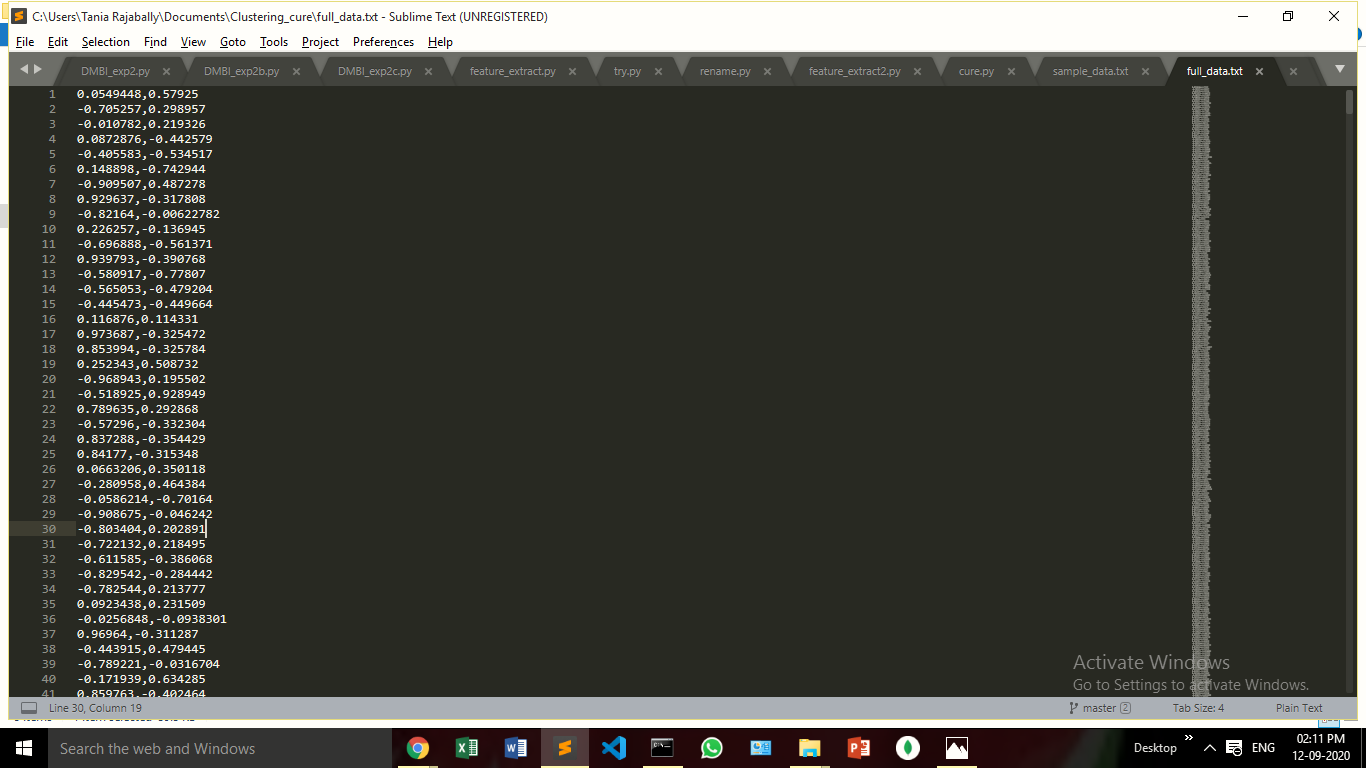
print("------------------")

print(len(list))

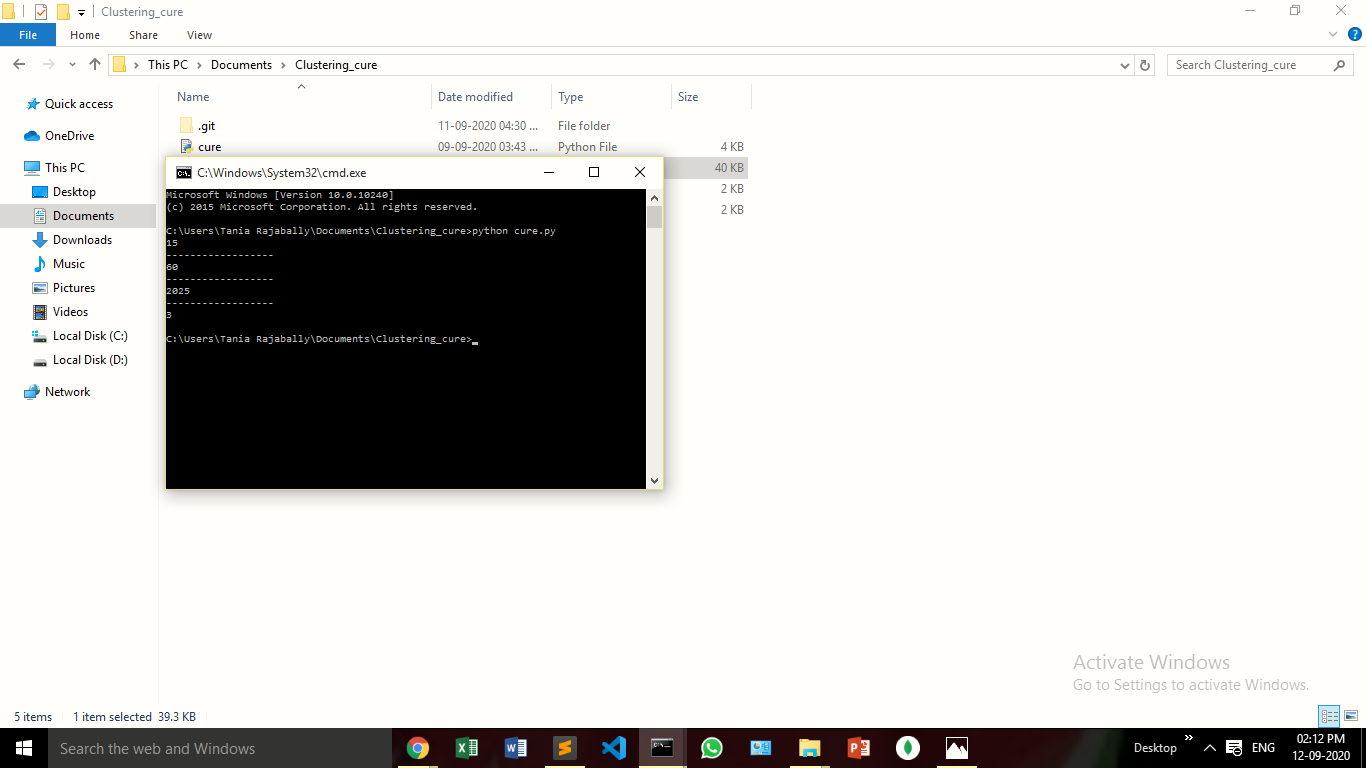
**Sample\_data.txt:**

****

**Full\_data.txt:**

****

**Output:**

****

**Conclusion:**

In this experiment, we implemented the CURE algorithm and saw how clusters were formed. We saw how representative points were used for hierarchical clustering.