**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра алгоритмической математики**

**отчет**

**по практической работе №7**

**по дисциплине «Статистический анализ»**

**Тема: Кластерный анализ. Метод поиска сгущений.**

|  |  |  |
| --- | --- | --- |
| Студент гр. 9372 |  | Иванов Р. С. |
| Преподаватель |  | Сучков А. И. |

Санкт-Петербург

2021

**Цель работы.**

Освоение основных понятий и некоторых методов кластерного анализа.

**Основные теоретические положения**

Метод поиска сгущений:

Радиус R лежит в интервале (Rmin, Rmax)

Rmin = min {dij > 0}, Rmax = max {dij}

Радиус выбирается произвольно, среди всех расстояний между точками, не считая наибольшего расстояния (будет один кластер при любом центроиде) и наименьшего (число кластеров всегда равно числу точек при любых центроидах).

Алгоритм:

1. Выбираем случайную точку из множества.

2. Вокруг этой точки раздуваем окружность радиуса R(в многомерном пространстве многомерную сферу).

3. Добавляем в кластер все точки, попавшие в окружность(сферу).

4. Перемещаем центр из первоначально выбранной точки в центр масс получившегося кластера.

5. Если центр масс сместился, переходим к пункту 3.

6. Исключаем получившийся кластер из множества точек.

7. Если множество точек не пустое, переходим на следующую итерацию алгоритма - возвращаемся к пункту 1.

**Постановка задачи.**

Дано конечное множество из объектов, представленных двумя признаками (в качестве этого множества принимаем исходную двумерную выборку, сформированную ранее в практической работе №4). Выполнить разбиение исходного множества объектов на конечное число подмножеств

(кластеров) с использованием метода поиска сгущений. Полученные результаты содержательно проинтерпретировать.

**Выполнение работы.**

***Пункт 1***

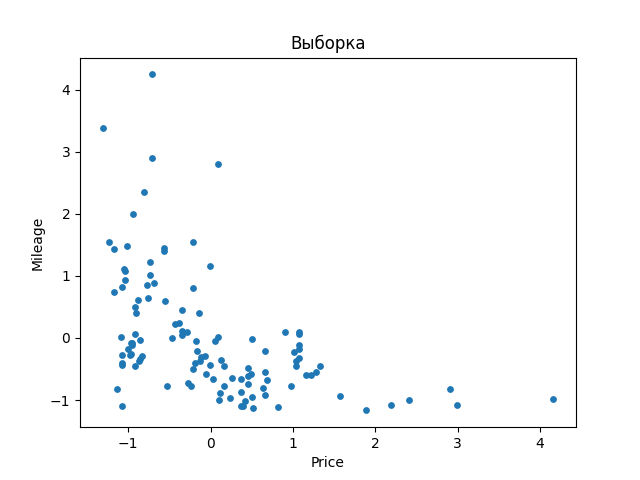
****Нормализовали множество точек по формуле (1)

Рисунок 1 (Выборка)

(1)

***Пункт 2***

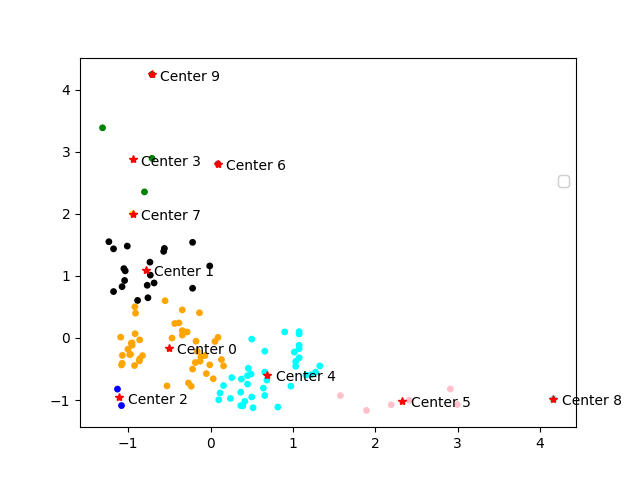
****Реализовали алгоритм поиска сгущений, отобразили полученные кластеры, выделив каждый кластер разным цветом и отметив центроиды звездочкой для .

Рисунок 2 (Метод сгущений)

***Пункт 3***

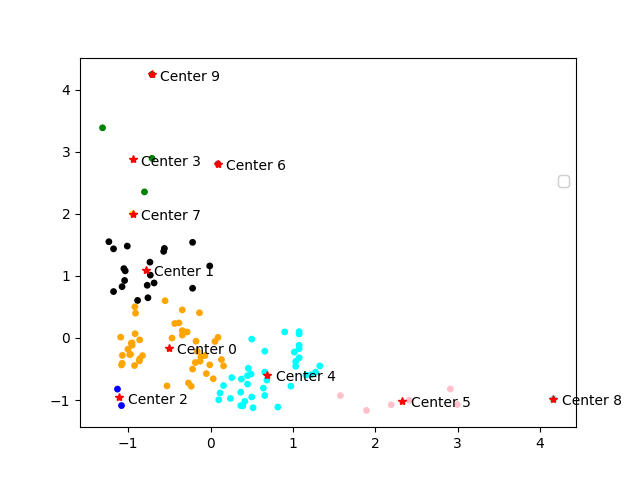
****Для проверки чувствительности метода берём те же начальные центроиды, сгенерированные еще в предыдущем пункте, добавляя погрешность к радиусу и прогоняя алгоритм.

Рисунок 3 (Проверка чувствительности к погрешностям)

Метод к погрешностям не чувствителен.

***Пункт 4***

Сравним с методами из практической работы №6.

Рассчитаем оценку качества разбиения для всех трех методов кластеризации:

|  |  |  |
| --- | --- | --- |
| K-means | K-medoids | Forel |
| 21.94 | 35.31 | 28.04 |

Таким образом, FOREL при R=0.7 показал приблизительно такое же качество разбиения, как алгоритмы k-means и k-medoids. Хуже всего из всех трёх методов себя показал метод k-medoids

**Выводы.**

В процессе выполнения практической работы был реализованы такой алгоритм разбиения на кластеры, как метод поиска сгущений Forel. А также установлена его чувствительность к погрешности и сравнение с таким алгоритмами, как k-means и k-medians, которое привело к выводу, что на данной выборке метод справляется лучше k-medoids, но хуже k-means.

Приложение А

Код программы

import numpy as np  
import matplotlib.pyplot as plt  
import math  
import csv  
# import scipy  
import random  
  
  
# from openpyxl.drawing.image import Image  
  
  
def calc\_distance(coordinate\_1, coordinate\_2):  
 return (coordinate\_1[0] - coordinate\_2[0]) \*\* 2 + (coordinate\_1[1] - coordinate\_2[1]) \*\* 2  
  
  
def recalc\_medoids(s, clu, medoid):  
 m = 10 \*\* 10  
 ans = -1  
 for i in range(len(clu)):  
 med = medoid  
 cluL = clu  
 med = s[cluL[i]]  
 rang = all\_ranges(s, clu, med)  
 if rang < m:  
 m = rang  
 ans = i  
 if ans != -1:  
 medoid = s[clu[ans]]  
 return clu, medoid  
  
  
def all\_ranges(s, clu, medoid):  
 ans = 0  
 for i in range(len(clu)):  
 ans += calc\_distance(medoid, s[clu[i]])  
 return ans  
  
  
def cluster\_index(D, cluster):  
 return [D.index(coordinate) for coordinate in cluster]  
  
  
def recalc\_centroids(cluster):  
 centroid = []  
 temp\_1 = 0  
 temp\_2 = 0  
 len\_cluster = len(cluster)  
 for i in range(len\_cluster):  
 temp\_1 += cluster[i][0]  
 temp\_2 += cluster[i][1]  
 centroid.append(temp\_1 / len\_cluster)  
 centroid.append(temp\_2 / len\_cluster)  
 return centroid  
  
  
def make\_plot(centers, sample, clusters):  
 mycolors = ['orange', 'black', 'blue', 'green', 'cyan', 'pink', 'brown', 'yellow', 'aqua','darkgreen','darkmagenta', 'darkseagreen','fuchsia','goldenrod','indianred','lightseagreen','palegreen','salmon','skyblue','turquoise','coral','darkblue','firebrick','gold','indigo','lime']  
 sample = np.array(sample)  
 fig, ax = plt.subplots()  
 for i, coordinate in enumerate(centers):  
 ax.plot(coordinate[0], coordinate[1], 'r\*')  
 ax.text(coordinate[0] + 0.1, coordinate[1] - 0.1, 'Center ' + str(i))  
 labels\_ = [0] \* len(sample)  
 sum = 0  
 for i in range(len(clusters)):  
 for index in range(len(clusters[i])):  
 labels\_[clusters[i][index]] = mycolors[i]  
  
 # for cluster in clusters:  
 # for index in cluster:  
 # ax.text(D[index, 0] + 0.01, D[index, 1] + 0.1, index)  
 scatter = ax.scatter(sample[:, 0], sample[:, 1], s=15, c=labels\_, cmap='rainbow')  
 legend = ax.legend(\*scatter.legend\_elements(fmt='Cluster {x:.0f}'), bbox\_to\_anchor=(1, 0.7))  
 ax.add\_artist(legend)  
 plt.show()  
  
  
sample = []  
  
with open('Price-Mileage.csv') as csv\_file: # Читаем выборку из файла 0 работы  
 spam\_reader = csv.reader(csv\_file, quotechar='|')  
 for row in spam\_reader:  
 x, y = row[0].split(';')  
 if y.isdigit() and x.isdigit():  
 sample.append([int(x), int(y)])  
  
n = len(sample)  
k = math.ceil(math.sqrt(n / 2))  
  
Xs = 0 # Выброчное среднее по x  
Ys = 0 # По y  
  
Dsx = 0 # Дисперсия по x  
Dsy = 0 # По s  
  
sy = 0 # Исправленное СКО по y  
sx = 0 # По x  
  
for i in range(n):  
 Xs += sample[i][0]  
Xs = Xs / n  
  
for i in range(n):  
 Ys += sample[i][1]  
Ys = Ys / n  
  
for i in range(n):  
 Dsx += (sample[i][0] - Xs) \*\* 2  
Dsx = Dsx / n  
  
for i in range(n):  
 Dsy += (sample[i][1] - Ys) \*\* 2  
Dsy = Dsy / n  
  
sx = math.sqrt(Dsx) \* n / (n - 1)  
sy = math.sqrt(Dsy) \* n / (n - 1)  
  
for i in range(n): # Нормализация точек  
 sample[i][0] = (sample[i][0] - Xs) / sx  
 sample[i][1] = (sample[i][1] - Ys) / sy  
  
X = [sample[i][0] for i in range(n)]  
Y = [sample[i][1] for i in range(n)]  
  
plt.scatter(x=X, y=Y, s=15)  
ax = plt.gca()  
ax.set(title='Выборка')  
# Добавляем подписи к осям:  
ax.set\_xlabel('Price')  
ax.set\_ylabel('Mileage')  
plt.show()  
  
def ranges\_clusters(clus, centrs, samp):  
 ans = 0  
 for i in range(len(clus)):  
 ans += all\_ranges(samp, clus[i], centrs[i])  
 return ans  
  
  
def rang\_R(point, sam, R):  
 clu = []  
 for i in sam:  
 ze = calc\_distance(point, i)  
 if ze <= R:  
 clu.append(i)  
 return clu  
  
  
def find\_thick(R, sam):  
 samp = sam # Список из которого мы будем изымать точки  
 cluster = []  
 cents = []  
 c = -1  
 for i in range(len(samp)):  
 samp[i].append(i)  
 while len(samp) != 0:  
 c += 1  
 thr = 1  
 init = random.choice(samp)  
 while thr != 0:  
 clu = []  
 cluN = []  
 clu = rang\_R(init, samp, R)  
 for i in clu:  
 cluN.append(i[2])  
 new\_init = recalc\_centroids(clu)  
 thr = calc\_distance(init, new\_init)  
 init = new\_init  
 for i in clu:  
 samp = list(filter(lambda e: e != i, samp))  
 cluster.append(cluN)  
 cents.append(init)  
 return cluster, cents  
  
  
def ranges\_clusters(clus, centrs, samp):  
 ans = 0  
 for i in range(len(clus)):  
 ans += all\_ranges(samp,clus[i],centrs[i])  
 return ans  
  
centers = []  
Radius = 0.7  
clusters, centers = find\_thick(Radius, sample)  
make\_plot(centers, sample, clusters)  
  
range = ranges\_clusters(clusters, centers, sample)  
print('range = ', range)  
  
b=0