实验三《k-means聚类算法》

### 实验题目

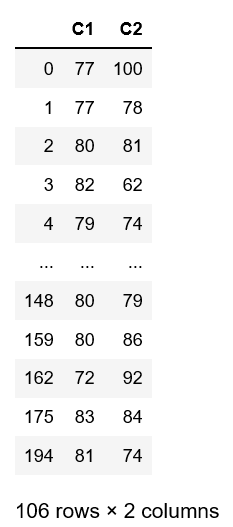
用C++实现k-means聚类算法：

1. 对实验二中的z-score归一化的成绩数据进行测试，观察聚类为2类，3类，4类，5类的结果，观察得出什么结论？

2.由老师给出测试数据，进行测试，并画出可视化出散点图，类中心，类半径，并分析聚为几类合适。

### 实验数据

以C1、C2举例运算



### 实验环境

Windows 10、Visual Studio 2017、Jupyter

### 实验代码

#### C++代码

#include <iostream>

#include <sstream>

#include <fstream>

#include <string>

#include <vector>

#include <time.h>

using namespace std;

//创建一个Point结构体,用来存储点与点所属聚类的类型

typedef struct Point {

float x;

float y;

int cluster;//代表所属聚类的类型

Point() {}//构造函数

Point(float a, float b, int c) {//构造函数

x = a;

y = b;

cluster = c;

}

}point;

//利用stringstream将string类型转换成float类型

float stringToFloat(string s) {

stringstream sf;

float score = 0;

sf << s;

sf >> score;

return score;

}

//读取data文件，并返回拥有每一个点结构体的矢量dataset

vector<point> openFile(const char\* data) {

fstream file;

file.open(data, ios::in);//以可读方式打开文件

vector<point> dataset;//向量dataset用来存储每一个点的结构体

while (!file.eof()) {//读取文件每一行

string temp;

file >> temp;//将读取到的行赋值给temp

int split = temp.find(',', 0);//split存储,的下标

//将点的信息用结构体表示，聚类类型初始化为0

point p(stringToFloat(temp.substr(0, split)), stringToFloat(temp.substr(split + 1, temp.length())), 0);

dataset.push\_back(p);//将点的结构体添加进矢量dataset中

}

file.close();

return dataset;

}

//计算两点距离的平方

float squareDistance(point a, point b) {

return (a.x - b.x)\*(a.x - b.x) + (a.y - b.y)\*(a.y - b.y);

}

//k聚类算法

void k\_means(vector<point> dataset, int k) {//dataset为拥有每一个点结构体的矢量，k为聚类的类型数

vector<point> centroid;//类中心集合

int n = dataset.size();//点个数

srand((int)time(0));

//在dataset中随机选取k个类中心

for (int i = 1; i <= k; i++) {

int cen = rand() % n;

point cp(dataset[cen].x, dataset[cen].y, i);

centroid.push\_back(cp);

}

int f = 1;

//聚类

do {

cout << "第" << f++ << "次\n";

for (int i = 0; i < k; i++) {

cout << "第" << centroid[i].cluster << "类类中心(" << centroid[i].x << "," << centroid[i].y << ")\n";

}

//找到所有点的最近的类中心

for (int i = 0; i < n; i++) {

float shortest = INT\_MAX;//记录距离平方的最小值

//用每个点分别与每个类中心进行比较得出shortest，将对应的类型赋值

for (int j = 0; j < k; j++) {

float temp = squareDistance(dataset[i], centroid[j]);

if (temp < shortest) {

shortest = temp;

dataset[i].cluster = j + 1;

}

}

}

int S\_cluster = 0;//记录点到对应聚类中心的平方和

for (int i = 0; i < n; i++)

S\_cluster += squareDistance(centroid[dataset[i].cluster - 1], dataset[i]);

//聚类平均值

vector<point> centroid\_mean;//记录每种聚类的平均中心点

int \*cs = new int[k];//记录每一类拥有的个数，并初始化为0

for (int i = 0; i < k; i++) {

centroid\_mean.push\_back(point(0, 0, i + 1));

cs[i] = 0;

}

for (int i = 0; i < n; i++) {//记录每一类的点数值之和

centroid\_mean[dataset[i].cluster - 1].x += dataset[i].x;

centroid\_mean[dataset[i].cluster - 1].y += dataset[i].y;

cs[dataset[i].cluster - 1]++;

}

for (int i = 0; i < k; i++) {//平均化

centroid\_mean[i].x /= cs[i];

centroid\_mean[i].y /= cs[i];

}

int S\_mean = 0;//记录点到对应平均中心的平方和

for (int i = 0; i < n; i++)

S\_mean += squareDistance(centroid\_mean[dataset[i].cluster - 1], dataset[i]);

if (S\_cluster > S\_mean)//比较误差平方和是否局部最小，是则聚类完毕，否则更新类中心

centroid = centroid\_mean;

else break;

} while (1);

//将结果写进文件

fstream file;

file.open("C:/Users/ASUS/Desktop/cluster5.txt", ios::out);//以写形式打开文件

for (int i = 0; i < n; i++) {

file << dataset[i].x << "," << dataset[i].y << "," << dataset[i].cluster << "\n";

}

file.close();

}

int main() {

//以C1、C2举例

//读取文件，接收拥有每一个点结构体的矢量dataset

vector<point> dataset = openFile("C:/Users/ASUS/Desktop/data.txt");

//进行k聚类，可改变k

k\_means(dataset, 5);

return 0;

}

#### Jupyter代码

#导包

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

#根据条件获取数据

data\_2 = pd.DataFrame(np.loadtxt('C:/Users/ASUS/Desktop/cluster2.txt', dtype = str, delimiter = ',', encoding = 'UTF-8'))

x1 = data\_2[0][data\_2[2] == '1']

y1 = data\_2[1][data\_2[2] == '1']

x2 = data\_2[0][data\_2[2] == '2']

y2 = data\_2[1][data\_2[2] == '2']

fig = plt.figure()

ax1 = fig.add\_subplot()

#设置标题

ax1.set\_title("Clustering of 2")

plt.scatter(x1, y1, c = 'r', label = '1')

plt.scatter(x2, y2, c = 'g', label = '2')

#设置标签

plt.legend()

plt.show()

#根据条件获取数据

data\_3 = pd.DataFrame(np.loadtxt('C:/Users/ASUS/Desktop/cluster3.txt', dtype = str, delimiter = ',', encoding = 'UTF-8'))

x1 = data\_3[0][data\_3[2] == '1']

y1 = data\_3[1][data\_3[2] == '1']

x2 = data\_3[0][data\_3[2] == '2']

y2 = data\_3[1][data\_3[2] == '2']

x3 = data\_3[0][data\_3[2] == '3']

y3 = data\_3[1][data\_3[2] == '3']

fig = plt.figure()

ax1 = fig.add\_subplot()

#设置标题

ax1.set\_title("Clustering of 3")

#制作散点图

plt.scatter(x1, y1, c = 'r', label = '1')

plt.scatter(x2, y2, c = 'g', label = '2')

plt.scatter(x3, y3, c = 'b', label = '3')

#设置标签

plt.legend()

plt.show()

#根据条件获取数据

data\_4 = pd.DataFrame(np.loadtxt('C:/Users/ASUS/Desktop/cluster4.txt', dtype = str, delimiter = ',', encoding = 'UTF-8'))

x1 = data\_4[0][data\_4[2] == '1']

y1 = data\_4[1][data\_4[2] == '1']

x2 = data\_4[0][data\_4[2] == '2']

y2 = data\_4[1][data\_4[2] == '2']

x3 = data\_4[0][data\_4[2] == '3']

y3 = data\_4[1][data\_4[2] == '3']

x4 = data\_4[0][data\_4[2] == '4']

y4 = data\_4[1][data\_4[2] == '4']

fig = plt.figure()

ax1 = fig.add\_subplot()

#设置标题

ax1.set\_title("Clustering of 4")

#制作散点图

plt.scatter(x1, y1, c = 'r', label = '1')

plt.scatter(x2, y2, c = 'g', label = '2')

plt.scatter(x3, y3, c = 'b', label = '3')

plt.scatter(x4, y4, c = 'c', label = '4')

#设置标签

plt.legend()

plt.show()

#根据条件获取数据

data\_5 = pd.DataFrame(np.loadtxt('C:/Users/ASUS/Desktop/cluster5.txt', dtype = str, delimiter = ',', encoding = 'UTF-8'))

x1 = data\_5[0][data\_5[2] == '1']

y1 = data\_5[1][data\_5[2] == '1']

x2 = data\_5[0][data\_5[2] == '2']

y2 = data\_5[1][data\_5[2] == '2']

x3 = data\_5[0][data\_5[2] == '3']

y3 = data\_5[1][data\_5[2] == '3']

x4 = data\_5[0][data\_5[2] == '4']

y4 = data\_5[1][data\_5[2] == '4']

x5 = data\_5[0][data\_5[2] == '5']

y5 = data\_5[1][data\_5[2] == '5']

fig = plt.figure()

ax1 = fig.add\_subplot()

#设置标题

ax1.set\_title("Clustering of 5")

#制作散点图

plt.scatter(x1, y1, c = 'r', label = '1')

plt.scatter(x2, y2, c = 'g', label = '2')

plt.scatter(x3, y3, c = 'b', label = '3')

plt.scatter(x4, y4, c = 'c', label = '4')

plt.scatter(x5, y5, c = 'm', label = '5')

#设置标签

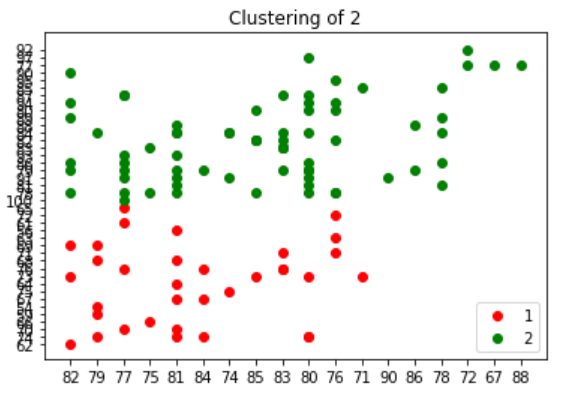
plt.legend()

plt.show()

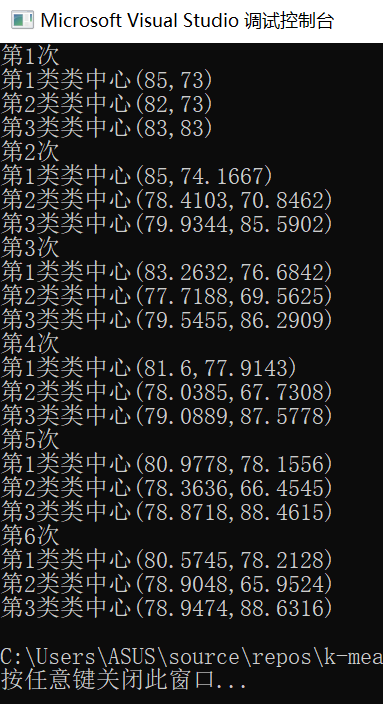
### 结果截图

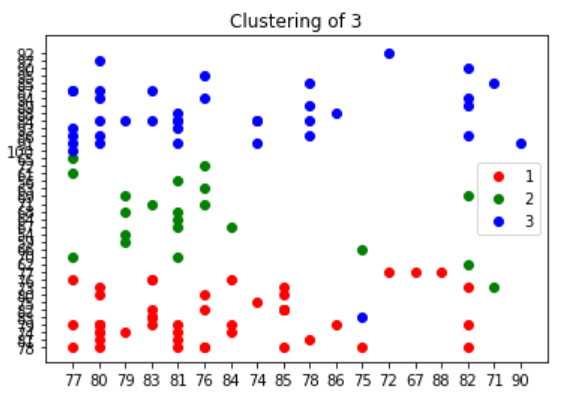
#### 聚类为2



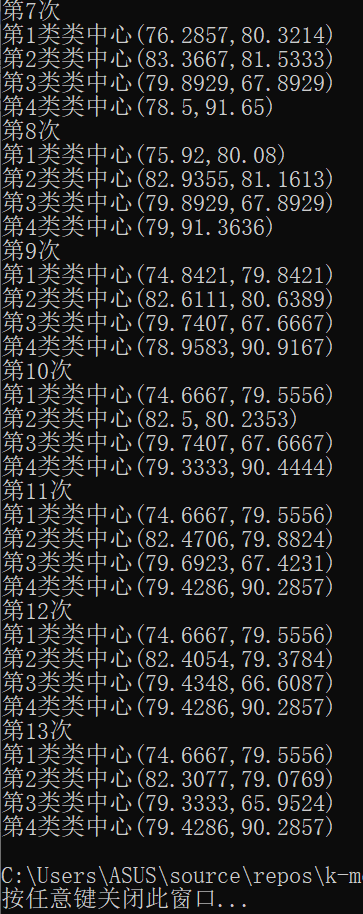
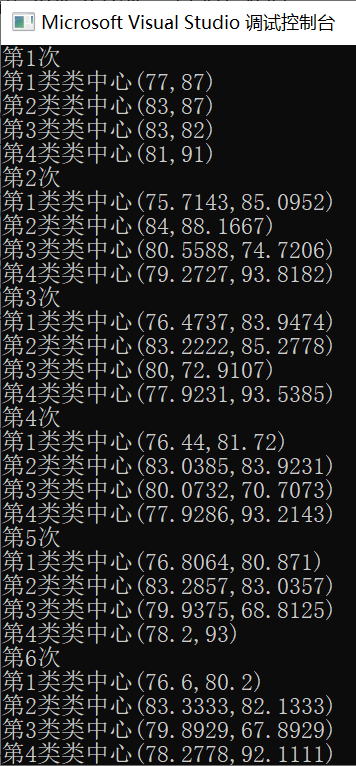


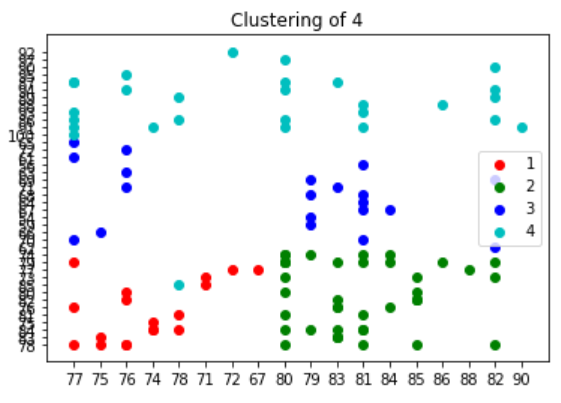
#### 聚类为3



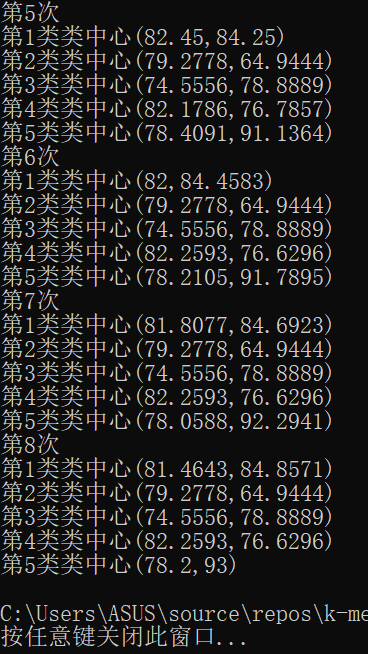
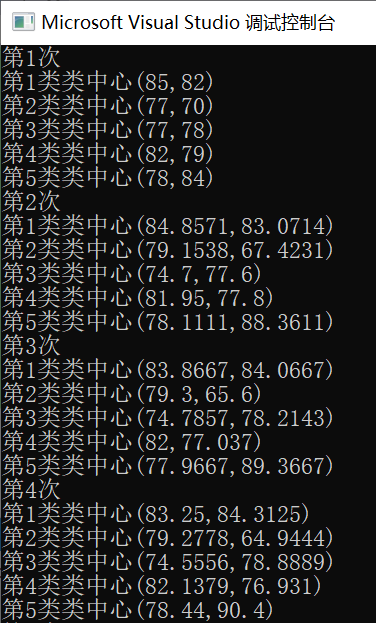


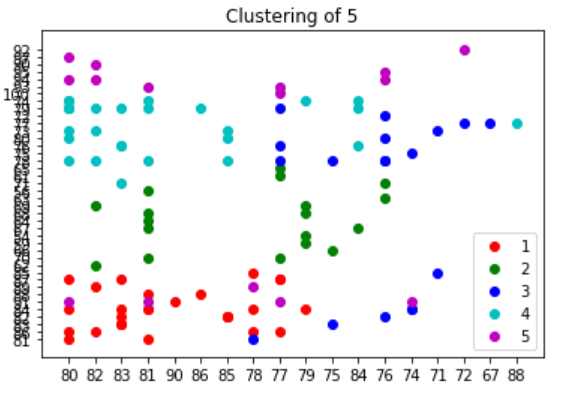
#### 聚类为4





#### 聚类为5





### 实验心得

由聚类为2类，3类，4类，5类的运行结果可知，k-means聚类算法的特点是不能保证该算法收敛域全局最优解，并且它常常终止于一个局部最优解。结果很大程度依赖于初始簇中心的随机选择。

算法优点：

1）原理比较简单，实现也是很容易，收敛速度快。

2）聚类效果较优。

3）算法的可解释度比较强。

4）主要需要调参的参数仅仅是簇数k。

算法缺点：

1）K值的选取不好把握

2）对于不是凸的数据集比较难收敛

3）如果各隐含类别的数据不平衡，比如各隐含类别的数据量严重失衡，或者各隐含类别的方差不同，则聚类效果不佳。

4） 采用迭代方法，得到的结果只是局部最优。