## 实验三 《k-means聚类算法》

**组长：卢学文**

**组员：卢学文、 夏旭、 李鸿洋**

#### 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。

样例数据(x,y)数据对：

|  |  |
| --- | --- |
| 3.45 | 7.08 |
| 1.76 | 7.24 |
| 4.29 | 9.55 |
| 3.35 | 6.65 |
| 3.17 | 6.41 |
| 3.68 | 5.99 |
| 2.11 | 4.08 |
| 2.58 | 7.10 |
| 3.45 | 7.88 |
| 6.17 | 5.40 |
| 4.20 | 6.46 |
| 5.87 | 3.87 |
| 5.47 | 2.21 |
| 5.97 | 3.62 |
| 6.24 | 3.06 |
| 6.89 | 2.41 |
| 5.38 | 2.32 |
| 5.13 | 2.73 |
| 7.26 | 4.19 |
| 6.32 | 3.62 |

找到聚类中心后，判断(2,6)是属于哪一类？

import pandas as pd

import numpy as np

from math import sqrt

import copy

import pickle

import random

import matplotlib.pyplot as plt

# def init\_k(mean\_point, k):

# k\_point = {}

# for i in range(0, k):

# k\_point[i] = {0: 0, 1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 0, 10: 0}

# for j in mean\_point:

# k\_point[i][j] = (1 + random.uniform(-0.5,0.5)) \* mean\_point[j]

# return k\_point

def init\_k(dataMatrix, k,num):

k\_point={}

record=[]

while(len(record))!=k:

r=random.randint(0,num-1)

if r not in record:

k\_point[len(record)]=dataMatrix[r]

record.append(r)

return k\_point

def tag(k\_point, k, dataMatrix,coord\_num):

p\_num={i: 0 for i in range(k)}

point\_tag = []

flag=-1

for line\_list in dataMatrix:

dis = 1000000000000

for i in range(0, k):

d = 0

for j in range(0, coord\_num):

d += (line\_list[j] - k\_point[i][j]) \*\* 2

if d < dis:

dis=d

flag = i

point\_tag.append(copy.deepcopy(flag))

p\_num[flag]+=1

return point\_tag,p\_num

def upgrade(point\_tag,k\_point,k,num,new\_dataMatrix,p\_num,coord\_num):

for i in range(0,k):

k\_point[i]=list(0 for i in range(coord\_num))

for i in range(0, num):

for j in range(0,coord\_num):

k\_point[point\_tag[i]][j]+=copy.copy(new\_dataMatrix[i][j])

for i in k\_point:

# if p\_num[i]==0:

# continue

k\_point[i]=[z/p\_num[i] for z in k\_point[i]]

return k\_point

def kmeans(dataMatrix,coord\_num,num,k):

k\_point = init\_k(dataMatrix, k,num)

p\_num={i: 0 for i in range(k)}

for i in range(0,1000):

flag\_num = copy.deepcopy(p\_num)

point\_tag ,p\_num= tag(k\_point, k, dataMatrix,coord\_num)

flag =0

for i in flag\_num:

if flag\_num[i]!=p\_num[i]:

flag=1

if flag==0:

break

k\_point=upgrade(point\_tag,k\_point,k,num,dataMatrix,p\_num,coord\_num)

for i in range(0,k):

print(k\_point[i],'\n')

return point\_tag,k\_point

def experiment(dataMatrix, k):

# 实验3第一题

mean\_point = {0: 0, 1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 0, 10: 0}

num = len(dataMatrix[0])

coord\_num=len(dataMatrix)

old\_dataMatrix = list(map(list, zip(\*dataMatrix)))

orderedNames =list(range(num))

new\_dataMatrix = np.array([old\_dataMatrix[i] for i in orderedNames])

kmeans(new\_dataMatrix,coord\_num,num,k)

# 实验三第二题

test\_x=[3.45,1.76,4.29,3.35,3.17,3.68,2.11,2.58,3.45,6.17,4.20,5.87,5.47,5.97,6.24,6.89,5.38,5.13,7.26,6.32]

test\_y=[7.08,7.24,9.55,6.65,6.41,5.99,4.08,7.10,7.88,5.40,6.46,3.87,2.21,3.62,3.06,2.41,2.32,2.73,4.19,3.62]

dataMatrix=list(zip(test\_x,test\_y))

num=len(test\_y)

coord\_num=len(dataMatrix[0])

point\_tag,k\_point=kmeans(dataMatrix,coord\_num,num,k)

plt.xlabel('X')

plt.ylabel('Y')

plt.xlim(xmax=11, xmin=0)

plt.ylim(ymax=11, ymin=0)

# 画两条（0-11）的坐标轴并设置轴标签x，y

plt.rcParams['font.sans-serif'] = ['SimHei']

plt.rcParams['axes.unicode\_minus'] = False

x={i: [] for i in range(k)}

y={i: [] for i in range(k)}

colors=['b','c','g','y','r']

labels=['类别A','类别B','类别C','类别D','类别E']

area = np.pi \* 4\*\*2 # 点面积

r={i: -1 for i in range(k)}#类半径

for i in range(0,num):

d = sqrt(pow((dataMatrix[i][0] - k\_point[point\_tag[i]][0]),2) + pow((dataMatrix[i][1] - k\_point[point\_tag[i]][1]),2))

if d > r[point\_tag[i]]:

r[point\_tag[i]] = d

x[point\_tag[i]].append(dataMatrix[i][0])

y[point\_tag[i]].append(dataMatrix[i][1])

theta = np.arange(0, 2 \* np.pi, 0.01)

for i in range(0,k):

plt.scatter(x[i],y[i], s=area, c=colors[i], alpha=0.4, label=labels[i])

plt.scatter(k\_point[i][0], k\_point[i][1], s=np.pi \* 2\*\*2, c=colors[i], alpha=1, label=labels[i]+'聚类中心')

plt.plot(k\_point[i][0] + r[i] \* np.cos(theta), k\_point[i][1] + r[i] \* np.sin(theta),c=colors[i])

plt.plot([0,9.5],[9.5,0],linewidth = '0.5',color='#000000')

plt.legend()

plt.show()

testpoint=[]

for x in range(0,k):

testpoint.append(sqrt((2-k\_point[x][0])\*\*2+(6-k\_point[x][1])\*\*2))

ans={0: 'A', 1: 'B', 2: 'C', 3: 'D', 4: 'E'}

for y in range(0,k):

if testpoint[y]<r[y]:

print('点（2，6）属于%c类'%(ans[y]))

def main():

pickle\_file=open('z-scoreData.txt','rb')

dataMatrix=pickle.load(pickle\_file)

for k in (2,3,4,5):

print("当分类类数为%d时"%(k))

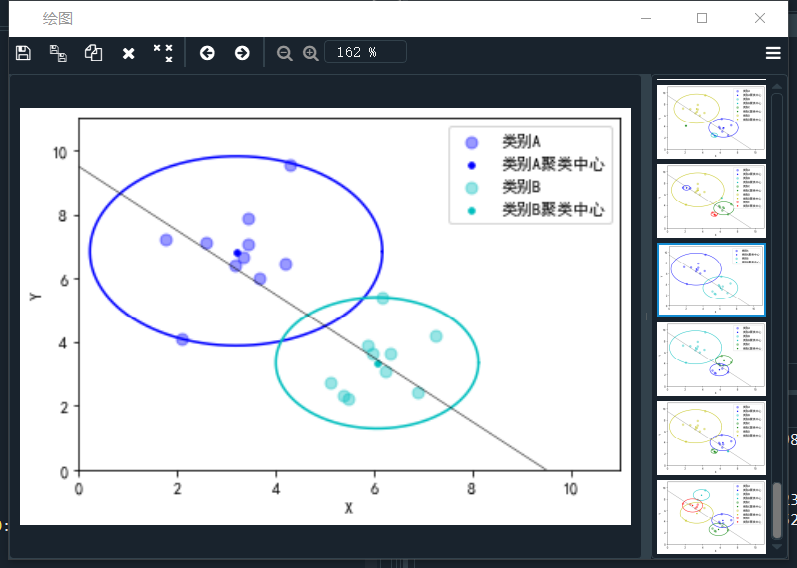
experiment(dataMatrix, k)

if \_\_name\_\_ == '\_\_main\_\_':

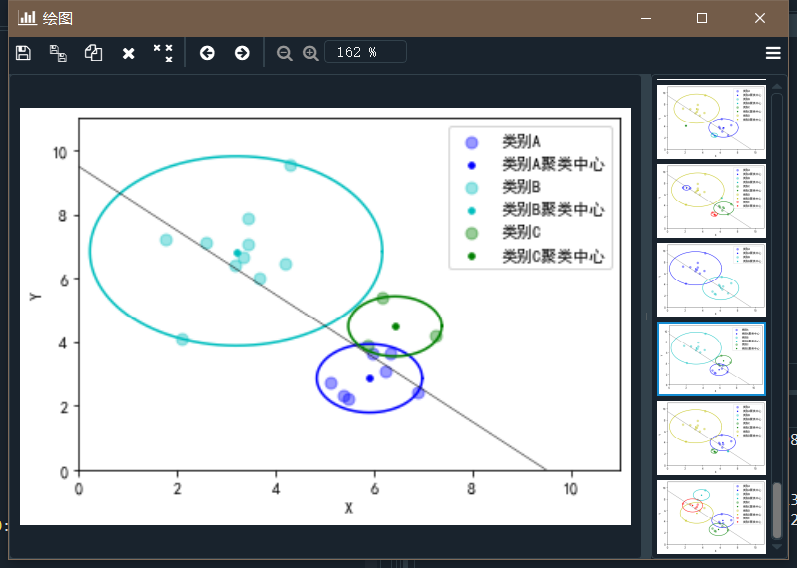
main()

结果图：

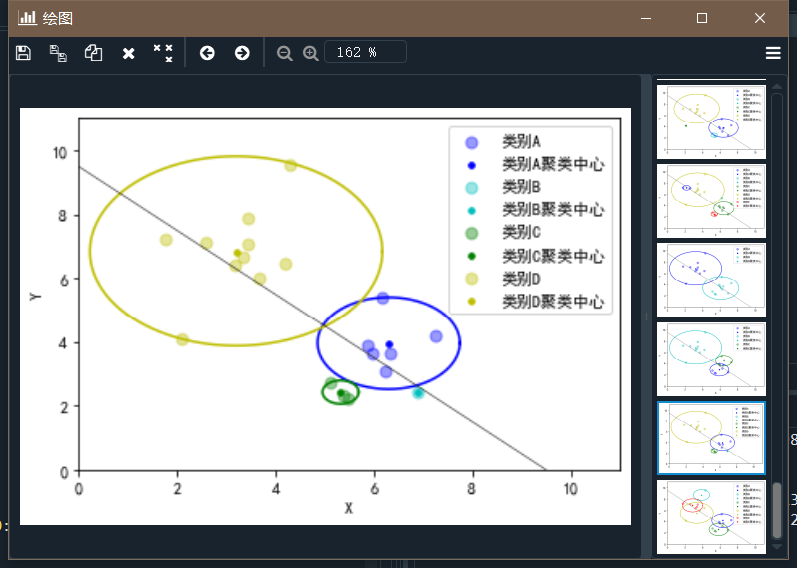
分为2类时



分为3类时



分为4类时



分为5类时

