**一、实验目的及要求**

1. 掌握对基本机器学习概念的理解

2. 掌握聚类实现的原理和方法

3. 按照既定格式书写实验报告

**二、实验设备与平台**

1. 实验设备：计算机；

2. 平台：Windows 10操作系统

**三、实验内容**

**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。

现有样例(x,y)数据对，

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| x | 3.45 | 1.76 | 4.29 | 3.35 | 3.17 | 3.68 | 2.11 | 2.58 | 3.45 | 6.17 | 4.2 | 5.87 | 5.47 | 5.97 | 6.24 | 6.89 | 5.38 | 5.13 | 7.26 | 6.32 |
| y | 7.08 | 7.24 | 9.55 | 6.65 | 6.41 | 5.99 | 4.08 | 7.1 | 7.88 | 5.4 | 6.46 | 3.87 | 2.21 | 3.62 | 3.06 | 2.41 | 2.32 | 2.73 | 4.19 | 3.62 |

**实验实施**：

（在此详述平台，技术栈，思路，处理逻辑等等）

平台：Windows10

技术栈：java

思路：先理解Kmeans算法，用java代码实现，用提供的数据集实例化运行。

**代码：**

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

def random\_point(k):

'''

随机质心

'''

b = set()

while(len(b) < k):

b.add(np.random.randint(0, 150))

return(b)

# 每个点到中心点距离距离

def get\_distance(point\_x, point\_y, cent\_x, cent\_y, k):

x = point\_x

y = point\_y

x0 = cent\_x

y0 = cent\_y

i = 0

j = 0

ds = [[]for i in range(len(x))]

while i < len(x):

while j < k:

M = np.sqrt((x[i] - x0[j]) \* (x[i] - x0[j]) +

(y[i] - y0[j]) \* (y[i] - y0[j]))

M = round(M, 1)

j = j + 1

ds[i].append(M)

j = 0

i = i + 1

return(ds)

def error\_distance(point\_x, point\_y, cent\_x, cent\_y, k):

'''

计算距离误差

'''

x = point\_x

y = point\_y

x0 = cent\_x

y0 = cent\_y

i = 0

j = 0

sum = 0

while i < k:

while j < len(x):

M = (x[j] - x0[i]) \* (x[j] - x0[i]) + \

(y[j] - y0[i]) \* (y[j] - y0[i])

M = round(M, 1)

sum += M

j = j + 1

# ds[i].append(M)

j = 0

i = i + 1

return(sum)

def cent(lable):

'''

质心计算

'''

temp = lable

mean\_x = []

mean\_y = []

i = 0

j = 0

while i < 3:

cent\_x = 0

cent\_y = 0

count = 0

while j < len(x):

if i == temp[j]:

count = count + 1

cent\_x = cent\_x + x[j]

cent\_y = cent\_y + y[j]

j = j + 1

cent\_x = cent\_x / count

cent\_y = cent\_y / count

# 更新中心点

mean\_x.append(cent\_x)

mean\_y.append(cent\_y)

j = 0

i = i + 1

return[mean\_x, mean\_y]

# 按照k值聚类

def k\_means(ds, x):

x = x

x = len(x)

i = 0

temp = []

while i < x:

temp.append(ds[i].index(min(ds[i])))

i = i + 1

return(temp)

def k\_view():

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

fig = plt.figure(figsize=(20, 10), num='聚类图与原图对比')

ax1 = fig.add\_subplot(1, 2, 1)

ax1.scatter(x0, y0, color='r', s=50, marker='s')

ax1.scatter(x, y, c=temp, s=25, marker='o')

plt.xlabel('花萼面积Sepalarea')

plt.ylabel('花瓣面积Petalarea')

plt.title("聚类后的数据")

ax2 = fig.add\_subplot(1, 2, 2)

plt.xlabel('花萼面积Sepalarea')

plt.ylabel('花瓣面积Petalarea')

plt.title("原来的数据")

ax2.scatter(x, y, c=lable\_code, s=25, marker='o')

plt.show()

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

iris\_data = pd.read\_csv('iris.csv')

X = np.array(iris\_data.iloc[:, 0:4]) # 特征向量，并且是按顺序排列的

lable = np.array(iris\_data.iloc[:, 4]) # 标签

lable\_mapping = {

'Iris-setosa': 0,

'Iris-versicolor': 1,

'Iris-virginica': 2

}

lable\_code = iris\_data.iloc[:, 4].map(lable\_mapping).tolist()

# 数据集预处理，以花萼面积为横坐标，以花瓣面积做纵坐标

hua\_e = X[:, 0] \* X[:, 1]

hua\_ban = X[:, 2] \* X[:, 3]

k = 3

b = random\_point(k)

ceshi\_hua\_e = [hua\_e[i] for i in range(len(hua\_e)) if (i in b)]

ceshi\_hua\_ban = [hua\_ban[i] for i in range(len(hua\_ban)) if (i in b)]

ceshi\_lable = [lable[i] for i in range(len(lable)) if (i in b)]

x = hua\_e

y = hua\_ban

x0 = ceshi\_hua\_e

y0 = ceshi\_hua\_ban

n = 0

ds = get\_distance(x, y, x0, y0, k)

temp = k\_means(ds, x)

temp1 = error\_distance(x, y, x0, y0, k)

n = n + 1

center = cent(temp)

x0 = center[0]

y0 = center[1]

ds = get\_distance(x, y, x0, y0, k)

temp = k\_means(ds, x)

temp2 = error\_distance(x, y, x0, y0, k)

n = n + 1

# 判断迭代是否继续

while np.abs(temp2 - temp1) != 0:

temp1 = temp2

center = cent(temp)

x0 = center[0]

y0 = center[1]

ds = get\_distance(x, y, x0, y0, k)

temp = k\_means(ds, x)

temp2 = error\_distance(x, y, x0, y0, k)

n = n + 1

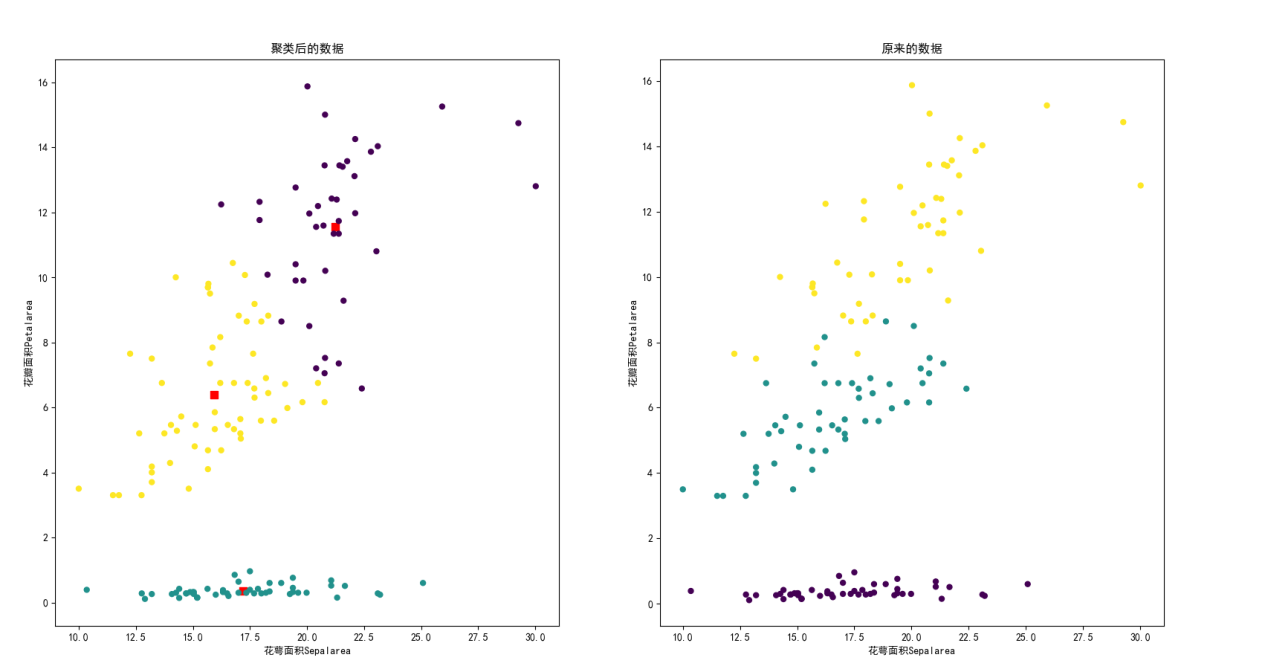
print(n, temp2)

print("迭代次数: ", n)

k\_view()

**运行结果**：

（运行结果、现象的说明与截图）



由于之前在人工智能实验做过该实验，因此直接使用了之前实验的鸢尾花数据集，得出的散点图如上。

**实验总结**：

（遇到的问题，有什么感想和收获等等）

通过这次实验，对Kmeans算法有了更进一步的了解。