**实验五 使用K-均值算法实现非监督模式识别**

# 一、实验目的

1．理解聚类的过程

2．理解并掌握K-均值算法的过程

3．理解PCA算法进行降维的原理和步骤

# 二、实验内容及要求：

1．实验数据：iris数据，一共150个数据，每个数据包含4个特征，假设样本类别未知，但已知类别数为3。

2．实验要求

1）采用PCA的方式将原始特征进行降维，要求降维后的特征能够保留原始特征80%以上的信息；

2）将降维后的新特征在新的特征空间画出样本点；

3）采用K-均值算法对降维后的特征进行分类，并画出分类后的样本点（以不同的颜色进行表示）；要求分别采用两种不同的初始代表点的确定方法，比较分类结果的不同；并与样本的真实类别进行比较，计算错误率。

4）对实验结果进行分析，分析初始代表点的不同对分类结果的影响。

5）使用python实现。

# 三、实验原理

第一步：选择K个初始聚类中心，

第二步：逐个将待分模式样本{X}按最小距离准则分配给某个聚类中心，

若 则

其中，k表示迭代次序，表示第j个聚类，是聚类中心。

第三步：分配完成后，计算各个聚类的新中心

其中，是第j类中所含样本数

第四步：

如果 则回到第二步，将模式样本重新聚类，重复迭代计算。

如果，则算法收敛，结束。

# 四、实验代码及结果分析

#-\*- coding: utf-8 -\*-

import numpy as np

from matplotlib import pyplot as plt

import math

# 将列表中的数据切片读入矩阵

def Read(lines,m,n):

    A = np.zeros((m, n))

    A\_row = 0  # 表示矩阵的行，从0行开始

    for line in lines:  # 把lines中的数据逐行读取出来

        list = line.strip('\n').split('\t')  # 处理逐行数据：strip表示把头尾的'\n'去掉，split表示以空格来分割行数据，然后把处理后的行数据返回到list列表中

        A[A\_row:] = list[0:5]  # 把处理后的数据放到方阵A中。list[0:4]表示列表的0,1,2,3列数据放到矩阵A中的A\_row行

        A\_row += 1  # 然后方阵A的下一行接着读

    return A

def DataInit():

    f = open("test.txt")

    lines = f.readlines() # 把全部数据文件读到一个列表lines中

    Line = len(lines) # 读取训练集合测试集的行数

    A = Read(lines,Line,5)

    return A

'''1.数据初始化'''

A = DataInit() # 四特征样本集(75\*5矩阵)

id = np.mat(A[:,0]).transpose() # 记录样本集的标号(75\*1列向量)

'''2.利用主成分分析方法进行样本降维'''

'''（1）计算协方差矩阵和均值向量'''

means = np.mean(A[:,1:],axis=0) # 特征均值

cov = np.cov(A[:,1:],rowvar=False) # 特征方差

'''（2）计算协方差矩阵的特征值、特征向量以及各主成分贡献率'''

val,vec = np.linalg.eig(cov) # 特征值与特征向量(原理：AX=λX )

val\_rate = val/np.sum(val) # 贡献率

'''（3）可视化特征值的贡献率'''

plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签

plt.rcParams['axes.unicode\_minus']=False #用来正常显示负号

plt.figure()

x = ["特征1","特征2","特征3","特征4"]

y = val

colors = [ 'gold', 'lightskyblue','yellowgreen', 'lightcoral']

plt.pie(y,labels=x,colors=colors,autopct='%1.1f%%',shadow=False,startangle=90)

plt.axis('equal') # 显示为圆（避免比例压缩为椭圆）

plt.title('各特征成分所占比例')

'''（4）求解变换后的二维新特征'''

index = np.argsort(-val) # 降序排序返回下角标

index = index[:2] # 取出前两大特征

A\_new = np.dot(A[:,1:],vec[index].transpose())

A\_new = np.concatenate((id,A\_new),axis=1) # 插入标签

# print(A\_new) # 降维后的新数据集（75\*3）

'''（5）降维后的新特征在新的特征空间画出样本点'''

plt.figure()

size = 10 # 点的大小

for item in A\_new:

    if item[0,0]==1.0: # 第一类

        plt.scatter(item[0,1], item[0,2],c='red',s=size)

    elif item[0,0]==2.0: # 第二类

        plt.scatter(item[0,1], item[0,2],c='blue',s=size)

    elif item[0,0]==3.0: # 第三类

        plt.scatter(item[0,1], item[0,2],c='yellow',s=size)

plt.title("降维后的新特征在样本空间的分布")

# plt.show()

'''3.k均值聚类实现'''

'''（1）算法初始化'''

A = A\_new

# 随机选取代表点

A\_random = A

np.random.shuffle(A\_random) # 随机打乱训练集

A1,A2,A3 = A\_random[0:25],A\_random[25:50],A\_random[50:75] # 分成三类

means = np.mean(A[:,1:],axis=0) # 特征均值

zz1,zz2,zz3 = np.mean(A1,axis=0),np.mean(A2,axis=0),np.mean(A3,axis=0)

z1 = [zz1[0,1],zz1[0,2]]

z2 = [zz2[0,1],zz2[0,2]]

z3 = [zz3[0,1],zz3[0,2]]

def get\_dis(item,z):

    return math.sqrt(math.pow(item[0,1]-z[0],2)+math.pow(item[0,2]-z[1],2))

def get\_center(res):

    x,y = 0,0

    num = len(res)

    for item in res:

        x += item[0,1]

        y += item[0,2]

    x,y = x/num, y/num

    return [x,y]

count =1

'''（2）根据聚类中心进行聚类的迭代过程'''

while 1:

    res1,res2,res3 = [],[],[]

    for item in A:

        dis = [get\_dis(item,z1),get\_dis(item,z2),get\_dis(item,z3)]

        min\_index = dis.index(min(dis))+1

        if min\_index == 1:

            res1.append(item)

        elif min\_index == 2:

            res2.append(item)

        elif min\_index == 3:

            res3.append(item)

    '''计算新的聚类中心'''

    old1 ,old2 ,old3 =z1,z2,z3

    z1,z2,z3 =get\_center(res1),get\_center(res2),get\_center(res3)

    if old1 == z1 and old2 == z2 and old3 == z3:

        print("质心在第%d次迭代中确定下来，迭代完毕！"%count)

        break

    else:

        print("第%d次迭代中质心发生变化，继续迭代"%count)

        count += 1

    print("now:",z1,z2,z3)

    print("old:",old1,old2,old3)

print("最终三类质心为：")

print(z1,z2,z3)

'''（3）测试准确率'''

def count\_accuracy(true,false,res,id):

    for item in res:

        if item[0,0] == id:

            true += 1

        else:

            false += 1

    return true,false

true,false = 0,0

true,false = count\_accuracy(true,false,res1,id=1)

true,false = count\_accuracy(true,false,res2,id=2)

true,false = count\_accuracy(true,false,res3,id=3)

print("基于k均值算法对三类样本进行分类：")

print("正确个数：",true)

print("错误个数：",false)

print("准确率：",true/(true+false),'\n')

'''（4）画出分类后的样本点'''

plt.figure()

for item in res1:

    plt.scatter(item[0,1], item[0,2],c='red',s=size) # 第一类

for item in res2:

    plt.scatter(item[0,1], item[0,2],c='blue',s=size) # 第二类

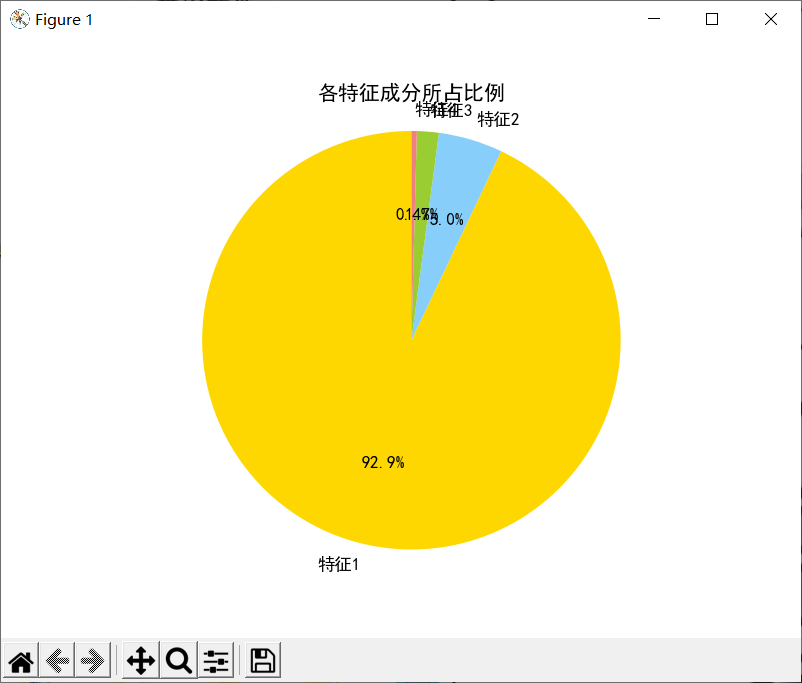
for item in res3:

    plt.scatter(item[0,1], item[0,2],c='yellow',s=size) # 第三类

plt.title("分类后的样本点在样本空间的分布")

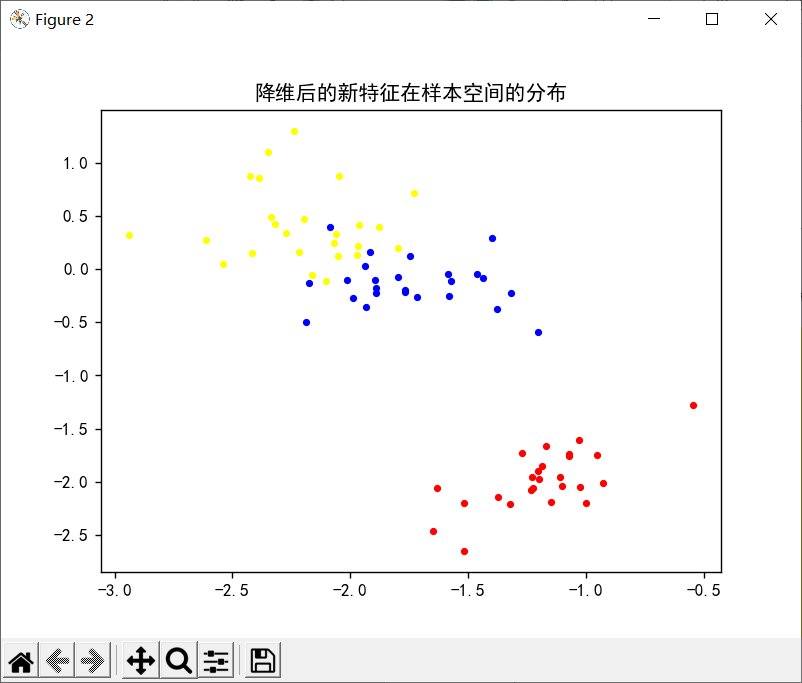
plt.show()

首先经过PCA降维，可以求出四个特征对应的特征值：

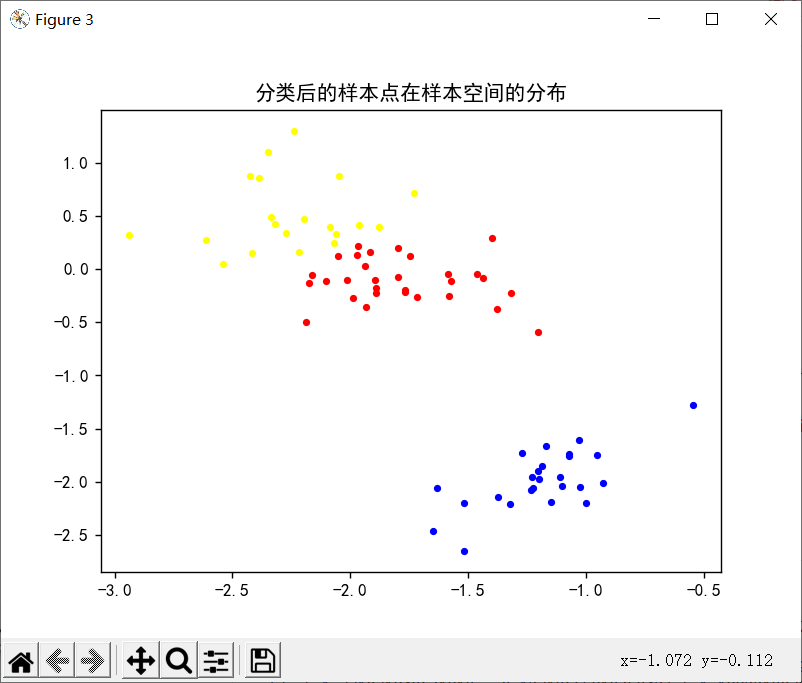
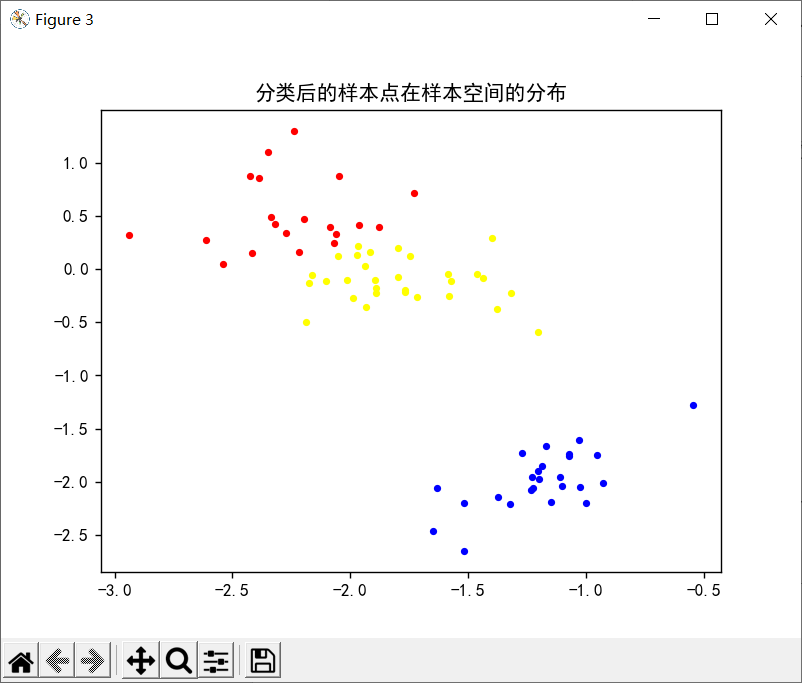


因此如果选择从四维降到二维，可以选择第一个、第二个特征：

下图为降维后新特征在样本空间的分布



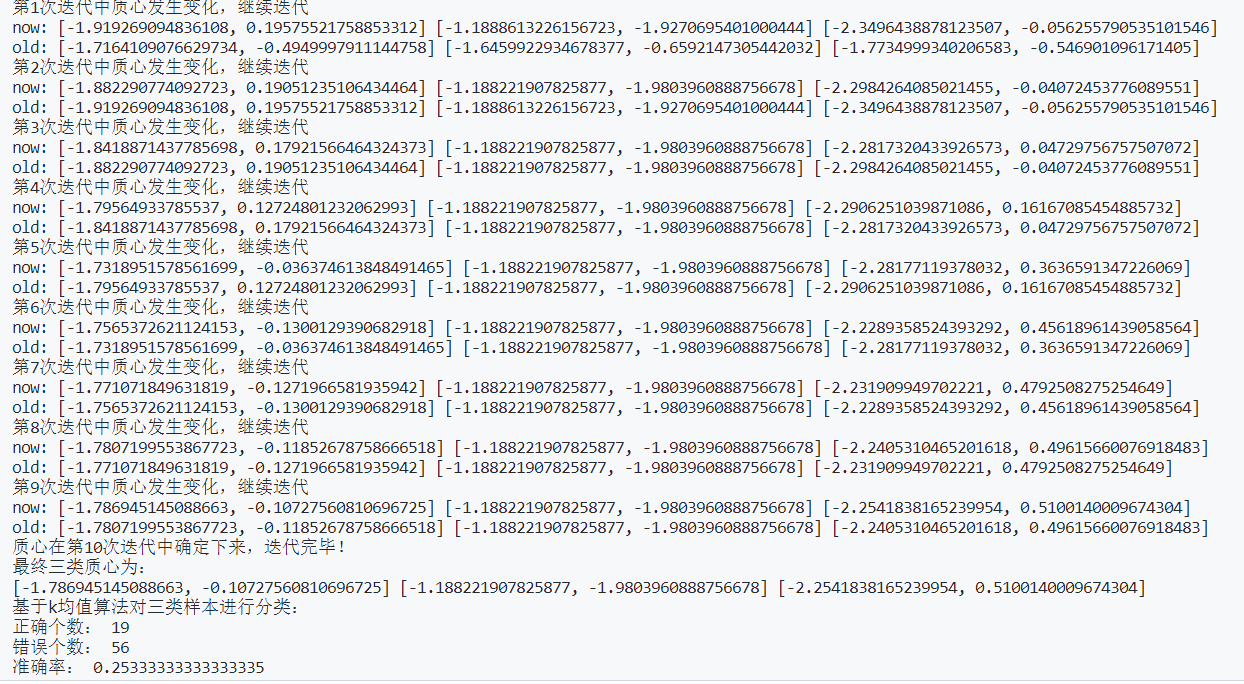
使用该样本进行k聚类：



运行多次是因为在选取初始质心时为随机选取，不同的初始质心会对分类造成不同影响。

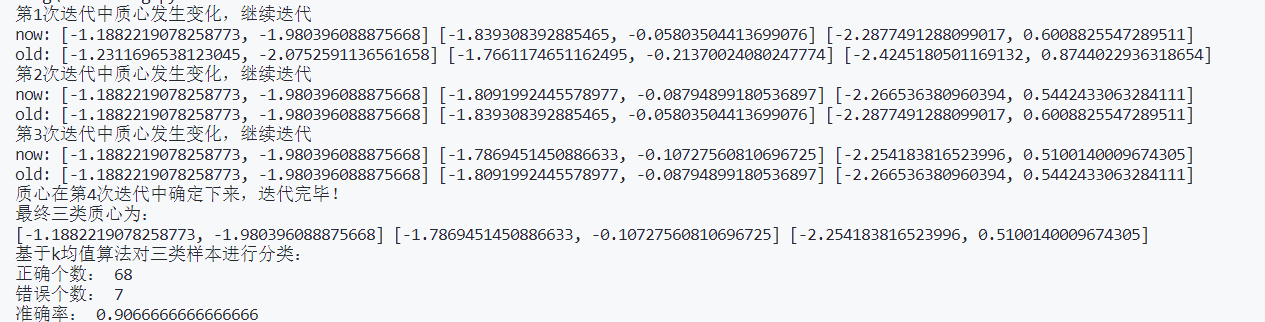
发现两次运行均成功将样本分成三类，分类面也较为明显

第一次迭代过程：



准确率25%

第二次迭代过程：



准确率90%，分类效果较为良好，达到了实验要求。