**机器学习实验报告**

**（一）**

****

姓名：崔玉峰

学号：201600301079

班级：2016级4班

1. **题目陈述**

考虑上机题2中的3个类别，设P(wi)=1/3。

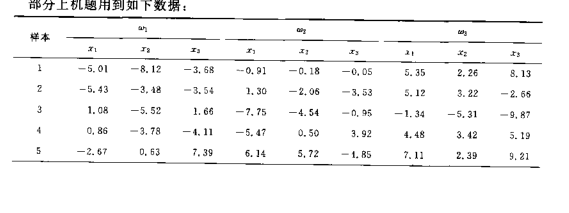
(a)以下各测试点与上机练习2中各类别均值间的Mahalanobis距离分别是多少:

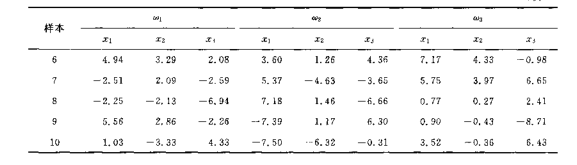
(1,2,1)^t,(5,3,2)^t,(0,0,0)^t ,(1,0,0)^t

(b)对以上各点进行分类。

(c)若设P(w1)=0.8,P(w2)= P(w3)=0.1,再对以上测试点进行分类。

**数据：**





1. **实验思路**

本实验共有三个题目，下面分析每个题目的具体解题思路：

1. **题目a的求解**

题目a的相对简单，主要用到了马氏距离公式，难点在于对与马氏距离公式的掌握情况，以及如何用代码实现。

*马氏距离公式：*



*变量说明：*

 d维均值向量

X d维向量

 d\*d协方差矩阵

r 马氏距离

 样本均值 可以通过样本数据求出，每个类别的样本均值为

X 为给出的各测试点

 为样本协方差矩阵 可以通过样本数据通过协方差公式求出，每

个类别的样本协方差为

所有的自变量都能求出，所以只要将它们带入公式就能求出测试点到各类别样本均值的马氏距离r

1. **题目b的求解**

① 根据题目可以假设 似然函数p(x|) 服从多元正态密度函数即：



***(1)***

② 根据贝叶斯公式可知：



③ 其中=1，所以此公式可以化为：



④ 通过对数分解简化可转化为：



***(2)***

⑤ 将***(1)***式带入***(2)***式可得正态函数判别式：

***(3)***

测试点在该类别算出的越大，则后验概率越大，则该测试点则更有可能属于该类别。只需算出每个测试点在在不同类别样本中，取最大值的那个类别即可.

**变量说明**

每个类别的样本均值为

X 为给出的各测试点

每个类别的样本协方差矩阵

 协方差行列式的值

d 样本维度

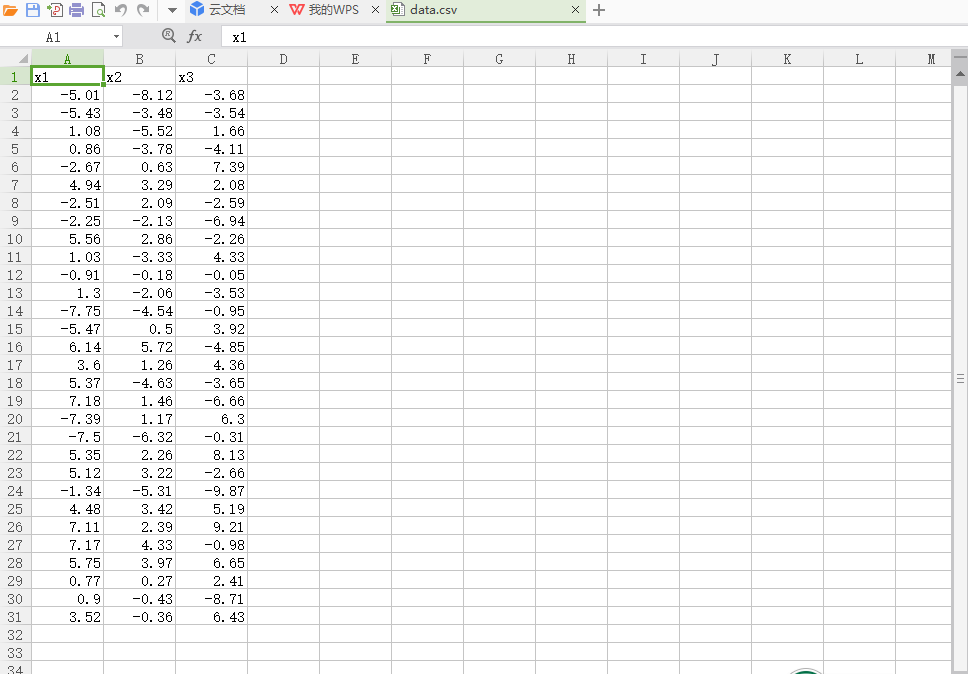
先验概率

1. **题目c的求解**

题目c与题目b的区别，就在于先验概率不同，只需将先验概率，更改再代入***(3)***即可。

1. **具体实现：**
2. **实验环境：**
3. 编程语言：Python
4. 软件环境 ：Jupyter Notebook
5. 硬件环境 ： PC
6. **实验准备：**

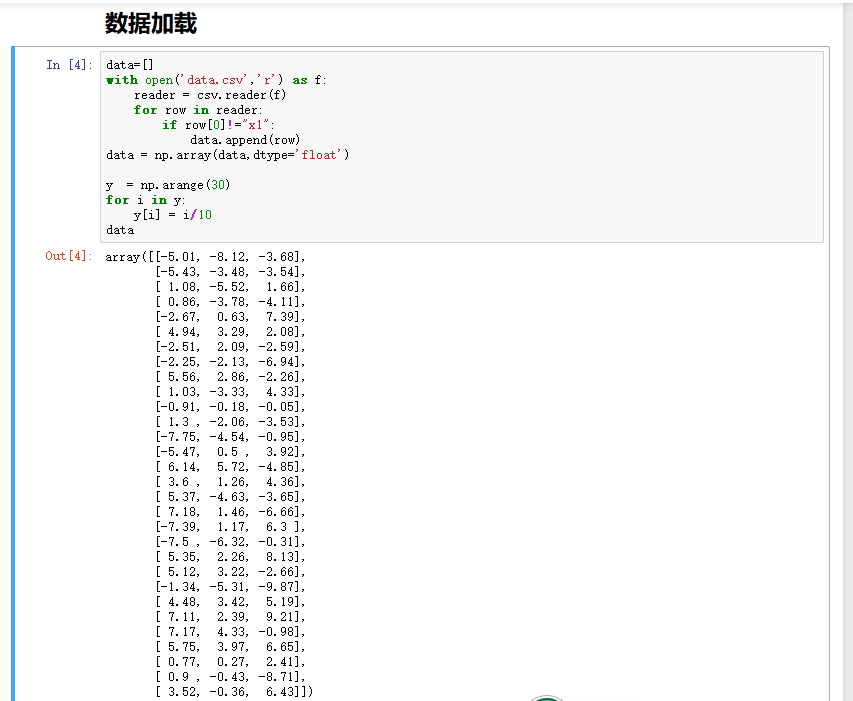
① 将上机实验所用到的数据，手动录入成csv文件方便程序编程使用，可以方便通过程序读入读出。



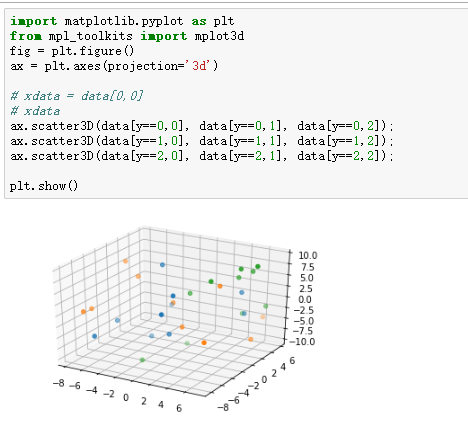
② 准备Numpy库：方便进行矩阵的运算操作。

1. **实验具体操作：**
2. **数据加载：**

将数据30条样本读入，每个样本三维（x1,x2,x3）,前十个样本属于类，中间十个样本属于类，最后十个样本属于类，将这三十个样本存入一个数组data中，并用一个数组y存储每个样本的类别。



1. **可以对样本数据点简单做绘图，看一看分布：**



从图上很难看出什规律，接下来开始具体方法的编写

1. **求马氏距离方法的编写：**

传入样本数据*dataw*和测试点*x*，该方法返回测试点到该样本均值的马氏距离。

样本协方差矩阵可由np.cov()方法求出

样本均值可用np.mean()方法求出

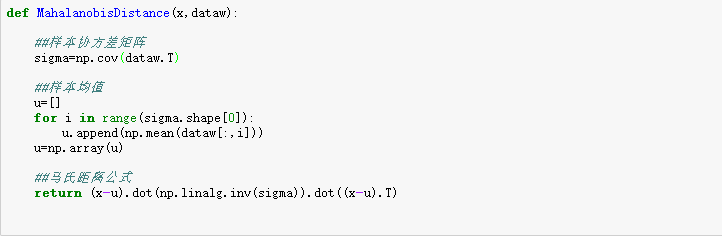
最后带入公式：



代码实现：

(x-u).dot(np.linalg.inv(sigma)).dot((x-u).T)

**具体代码实现：**



1. **正态分布判别函数的编写：**

传入测试点x\_predict，该样本先验概率Pw，以及样本数据dataw，函数返回测试点在该类别样本下的后验概率：

正态分布判别函数：



其中部分与马氏距离公式相同所以直接可以复用求解马氏距离公式的代码。

样本的维数为d。

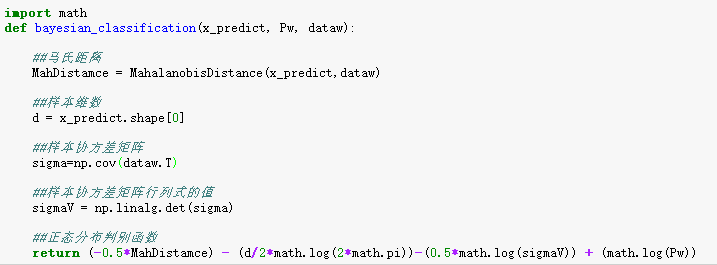
带入正态分布判别函数：



代码实现：

(-0.5\*MahDistamce) - (d/2\*math.log(2\*math.pi))-(0.5\*math.log(sigmaV)) + (math.log(Pw))

**具体代码实现：**



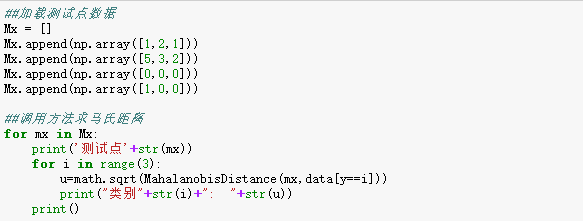
1. **问题a具体求解**

问题a的要求是求测试点到个类别均值间的马氏距离：

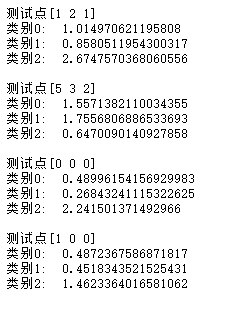
首先加载测试点数据，然后调用方法

def MahalanobisDistance(x,dataw):

**具体实现：**



**输出结果：**



每个测试点到每个类样本均值的马氏距离如上图所示。

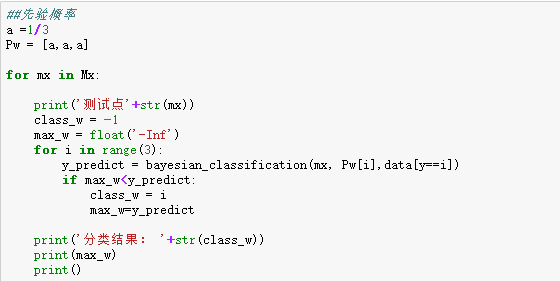
1. **问题b的求解：**

问题b为具体贝叶斯分类问题，每类的先验概率均为1/3，只要调用正态分布判别函数

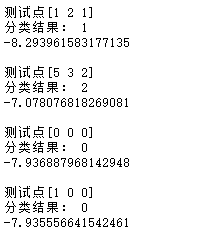
def bayesian\_classification(x\_predict, Pw, dataw):

求出测试点在每个类别的后验概率，后验概率最大的类别，即为该测试点的最终分类类别。

**具体代码实现：**



**测试结果：**



通过输出结果可知：

测试点[1,2,1] 的分类结果为

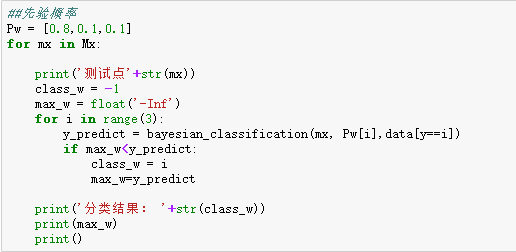
测试点[5,3,2] 的分类结果为

测试点[0,0,0] 和[1,0,0] 的分类结果为

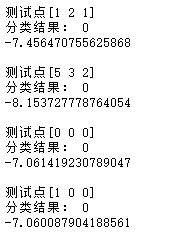
1. **问题c的求解：**

问题c与问题b的唯一差距在先验概率，只要更改先验概率继续重复问题b的操作即可得出结果。

**具体代码实现：**



**测试结果：**



通过输出结果显示所有的测试点均被分类为

本次实验的所有代码和数据，我会以.ipynb文件的形式上传，可以直接运行并且查看运行结果。

1. **实验心得与结论：**

实验完成了上机实验4的全部题目，并且测试通过，通过实验基本掌握了对于贝叶斯分类器的基本实现，并且巩固了贝叶斯决策论的内容，强化了课上所学的知识。

贝叶斯算法作为机器学习中最重要的算法之一，必须要掌握通过实验了解贝叶斯算法的具体实现，在开始进行实验时进展缓慢，对样本数，样本维数，样本类别数等最基本的概念也搞不清楚，在不断查询资料和回顾老师所讲的知识的过程中进步了很多，最后在弄懂了公式后编写代码，测试结果的速度也就变快，最终完成了整个实验，收获了很多。

机器学习算法的编写最重要的点在于，算法公式的理解，只要充分的理解了算法的公式，具体编写代码就简单不少了。本次实验的所有代码和数据，我会以.ipynb文件的形式上传，可以直接运行并且查看运行结果。

1. **实验代码：**

##数据加载

data=[]

with open('data.csv','r') as f:

reader = csv.reader(f)

for row in reader:

if row[0]!="x1":

data.append(row)

data = np.array(data,dtype='float')

y = np.arange(30)

for i in y:

y[i] = i/10

##马氏距离计算

def MahalanobisDistance(x,dataw):

##样本协方差矩阵

sigma=np.cov(dataw.T)

##样本均值

u=[]

for i in range(sigma.shape[0]):

u.append(np.mean(dataw[:,i]))

u=np.array(u)

##马氏距离公式

return (x-u).dot(np.linalg.inv(sigma)).dot((x-u).T)

##问题a的求解

##加载测试点数据

Mx = []

Mx.append(np.array([1,2,1]))

Mx.append(np.array([5,3,2]))

Mx.append(np.array([0,0,0]))

Mx.append(np.array([1,0,0]))

##调用方法求马氏距离

for mx in Mx:

print('测试点'+str(mx))

for i in range(3):

u=math.sqrt(MahalanobisDistance(mx,data[y==i]))

print("类别"+str(i)+": "+str(u))

print()

##正态分布判别函数

def bayesian\_classification(x\_predict, Pw, dataw):

##马氏距离

MahDistamce = MahalanobisDistance(x\_predict,dataw)

##样本维数

d = x\_predict.shape[0]

##样本协方差矩阵

sigma=np.cov(dataw.T)

##样本协方差矩阵行列式的值

sigmaV = np.linalg.det(sigma)

##正态分布判别函数

return (-0.5\*MahDistamce) - (d/2\*math.log(2\*math.pi))-(0.5\*math.log(sigmaV)) + (math.log(Pw))

##问题b求解

##先验概率

a =1/3

Pw = [a,a,a]

for mx in Mx:

print('测试点'+str(mx))

class\_w = -1

max\_w = float('-Inf')

for i in range(3):

y\_predict = bayesian\_classification(mx, Pw[i],data[y==i])

if max\_w<y\_predict:

class\_w = i

max\_w=y\_predict

print('分类结果： '+str(class\_w))

print(max\_w)

print()

##问题c求解

##先验概率

Pw = [0.8,0.1,0.1]

for mx in Mx:

print('测试点'+str(mx))

class\_w = -1

max\_w = float('-Inf')

for i in range(3):

y\_predict = bayesian\_classification(mx, Pw[i],data[y==i])

if max\_w<y\_predict:

class\_w = i

max\_w=y\_predict

print('分类结果： '+str(class\_w))

print(max\_w)

print()

1. **参考文献**

*[美]RichardO.Duda PeterE.Hart DavidG.Stork 著 模式分类 第二版*