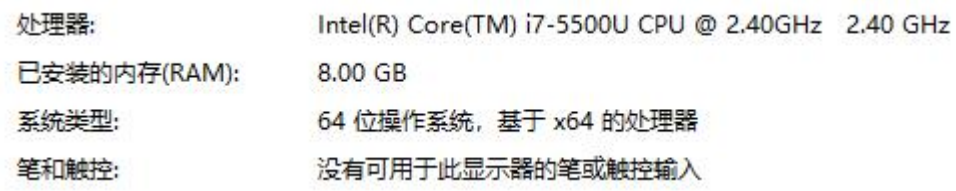
软件学院 机器学习 课程实验报告

实验题目：贝叶斯		学号：201700301166
日期：2019/10/21	班级： 网安 17	姓名： 尹成林
Email：sdyinruichao@163.		

实验软件和硬件环境：

编程语言：python3.7
使用到的库：numpy 库等（没有使用 sklearn 库）
操作系统：win10
硬件配置：



实验原理：

贝叶斯公式
最小分线贝叶斯分类器的公式等等
贝叶斯分类器编写的相关知识
Numpy 的多维高斯分布产生
Numpy 的统计计算
Random 库的随机选择进行数据分类

实验方法、实验结果及实验结果分析：

编程中的相关函数具体的作用在代码中已写好注释，这里略过。

问题 1 求两种分类器，在这里我实现了之前选作题目的相关要求，可以输入均值，协方差矩阵生成数据并训练

问题 2 给出数据要求进行数据分类并测试性能。

问题 1 实验结果：

结果为 1 为正样本，结果为 2 为负样本

```
请输入正样本第1维的均值1
请输入正样本第1维的方差2 0
请输入正样本第2维的均值3
请输入正样本第2维的方差0 2
请输入负样本第1维的均值10
请输入负样本第1维的方差10 0
请输入负样本第2维的均值20
请输入负样本第2维的方差0 10
**********产生样本并训练中**********
```

```
**********产生样本并训练中**********
**********训练完成**********
请输入一个二维向量(输入a a停止)
1 3
数据为正样本的风险为2.047355827035071e-08
数据为负样本的风险为0.056523088469508924
最小风险贝叶斯分类器所作出判断为1
数据为正样本的概率为0.09420514744918154
数据为负样本的概率为5.1183895675876765e-08
最小化错误率贝叶斯分类器做出判断为1
请输入一个二维向量(输入a a停止)
```

```
请输入一个二维向量(输入a a停止)
10 20
数据为正样本的风险为0.033464055280572925
数据为负样本的风险为1.0067476045089792e-10
最小风险贝叶斯分类器所作出判断为2
数据为正样本的概率为1.677912674181632e-10
数据为负样本的概率为0.08366013820143231
最小化错误率贝叶斯分类器做出判断为2
```

问题 1 实验方法：

1. 了解两种不同贝叶斯分类器的区别：简单的来看，最小错误率贝叶斯分类器就是于朴素贝叶斯分类器，该怎么写就怎么写。
而最小风险贝叶斯分类器是基于朴素贝叶斯之上进一步的方法，首先会有权威认证的评估出来的错误风险，把错误风险与概率的乘积作为贴标签的标准，取其中风险最小的。
2. 使用 python 的 numpy 库产生二维高斯分布正样本 10000、负样本 20000
3. 使用产生的数据产生类条件概率分布函数
4. 对数据进行分类

问题 1 实验结果分析：

通过不同的概率比较，得出较大的可能性，皆为判断结果，因为实验数据随机产生且为高斯分布，好像也没有进行测试的必要，准确率推测应该较高。

问题二实验结果：

```
当训练数据大小为1350个时，剩余数据的检测正确率为0.8170984455958549
想要进行多少层十折交叉验证10
经10层十折交叉验证，该朴素贝叶斯分类器平均分类正确率为0.8225007978694332


Process finished with exit code 0
```

问题 2 实验方法：

1. 了解贝叶斯分类器分类离散数据的原理：通过离散的类型进行统计，计算出相关的概率，先验后验概率等等等等
2. 使用 python 的 csv 库导入数据
3. 使用 python 的 random 库随机进行分组，分为十组以方便十折检验
4. 编写相关函数

问题 2 实验结果分析：

在刚做这道题的时候，我没有用十折测试，而是不断增加训练集的大小，发现随着训练集的增大，分类器的正确率也不断增大，这也符合我们的一贯认知。但同时，可能因为数据量在 10^3 也不是很大，尽管使用了十折，但每一次执行下来平均的正确率差异还是较大的，比较了多次甚至可能超过百分之 2，于是为了避免这种情况，我设计进行了多次的十折测试，最后得出的结果在不同时间下运行得到的结果差异不大，我觉得满足的要求，于是就采取了这样的办法。

实验感悟与收获：

1. 对课上所学的公式进行了实现，感受到了说与做之间的差别，以后要多多动手。
2. 发现即使通过 10 折交叉验证，验证出来的结果也不是变化较小，于是我选择进行多重 10 折交叉验证，减少这样的误差
3. 发现自己在听课过程中的一些不足，下一次上课我必坐第一排

实验代码：

第一问实验代码：

```
# 编写日期 2019/10/21
# 编写人 尹成林
# 学号 201700301166
# 实验 贝叶斯实验第 1 问

import numpy as np
from numpy.linalg import cholesky
import matplotlib.pyplot as plt
import math

# 定义数据
c = math.sqrt(2*math.pi)
posim = 10000
antisin  = 20000

def makepodata(m1, m2, s1, s2, s3, s4):
    sampleNo = posim;
    mu = np.array([[m1, m2]])
    Sigma = np.array([[s1, s2], [s3, s4]])
    R = cholesky(Sigma)
    s = np.dot(np.random.randn(sampleNo, 2), R) + mu
    return s

def makeantidata(m1, m2, s1, s2, s3, s4):
    sampleNo = antisin;
    mu = np.array([[m1, m2]])
    Sigma = np.array([[s1, s2], [s3, s4]])
    R = cholesky(Sigma)
    s = np.dot(np.random.randn(sampleNo, 2), R) + mu
    return s

def getu(s):
    h = np.mean(s, axis=0)
    return h

def gets(s):
    h = np.var(s, axis=0)
    return h
```

```
# 计算先验概率
def getPxc (num, u, s):
    p = (1/(c*math.sqrt(s)))*pow((math.e),(-(num-u)*(num-u)/(2*s)))
    return p

# 计算后验概率
def getPcx(c,u1,u2,s1,s2,num1,num2):
    p1 = getPxc(num1, u1, s1)
    p2 = getPxc(num2, u2, s2)
    if c == 1:
        return (1/3)*p1
    else:
        return (2/3)*p2

# 最小错误率计算和返回判断
def getminfau(num1, num2, u, s):
    p1 = getPcx(1, u[0], u[1], s[0], s[1], num1, num2)
    p2 = getPcx(2, u[2], u[3], s[2], s[3], num1, num2)
    print("数据为正样本的概率为{}".format(p1))
    print("数据为负样本的概率为{}".format(p2))
    if p1>p2:
        return 1
    else:
        return 2

# 最小风险判断
def getminrisk(num1, num2, u, s, risk):
    p1 = getPcx(1, u[0], u[1], s[0], s[1], num1, num2)*risk[0]
    p2 = getPcx(2, u[2], u[3], s[2], s[3], num1, num2)*risk[1]
    print("数据为正样本的风险为{}".format(p2))
    print("数据为负样本的风险为{}".format(p1))
    if p1>p2:
        return 1
    else:
        return 2

if __name__ == "__main__":
    u = []
    s = []
    for i in range(2):
        for i1 in range(2):
            if i==0:
                uuu = input("请输入正样本第{}维的均值".format(i1+1))
                u.append(int(uuu))
```

```
            u1,u2 = input("请输入正样本第{}维的方差".format(i1 + 1)).split()
            s.append(int(u1))
            s.append(int(u2))
        if i==1:
            uuu = input("请输入负样本第{}维的均值".format(i1+1))
            u.append(int(uuu))
            u1, u2 = input("请输入负样本第{}维的方差".format(i1 + 1)).split()
            s.append(int(u1))
            s.append(int(u2))
risk = [0.6, 0.4]
print("**********产生样本并训练中**********")
data1 = makepodata(
    u[0], u[1], s[0], s[1], s[2], s[3]
)
data2 = makeantidata(
    u[2], u[3], s[4], s[5], s[6], s[7]
)
U1 = getu(data1)
U2 = getu(data2)
S1 = gets(data1)
S2 = gets(data2)
print("**********训练完成**********")
while 1:
    a, b = input("请输入一个二维向量(输入a a停止)\n").split()
    a = int(a)
    b = int(b)
    UUU = []
    for i in U1:
        UUU.append(i)
    for i in U2:
        UUU.append(i)
    SSS = []
    for i in S1:
        SSS.append(i)
    for i in S2:
        SSS.append(i)
    print("最小风险贝叶斯分类器所作出判断为{}".format(getminrisk(a, b, UUU, SSS, risk)))
    print("最小化错误率贝叶斯分类器做出判断为{}".format(getminfau(a, b, UUU, SSS)))
```

第二问相关代码：

```
# 编写日期 2019/10/21
# 编写人 尹成林
# 学号 201700301166
# 实验 贝叶斯实验第 2 问
```

```
import csv
import random
import numpy as np
# import pandas
# 数据导入及分成两份
def loadcsv(name):
    f = csv.reader(open(name,'r'))
    dataset = list(f)
    return dataset

def randDivision(dataset , trainSize):
    copy = list(dataset)
    train = []
    while len(train)<trainSize:
        index = random.randrange(len(copy))
        train.append(copy.pop(index))
    return [train, copy]


#初始化一些数据

data1 = [[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0]]
dataunacc = [[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0]]
dataacc = [[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0]]
datagood = [[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0]]
dataVgood = [[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0]]
datavip = [0,0,0,0]

# 统计函数，将具体个数的多少进行统计
'''
****************************************
训练器部分
****************************************
'''

def stat(dataset):
    for i in dataset:
        count = i[0]
        maint = i[1]
        door = i[2]
        persons = i[3]
        lug = i[4]
        safty = i[5]
        vip = i[6]
```

```
        addcount(count, data1)
        addmaint(maint, data1)
        adddoor(door, data1)
        addperson(persons, data1)
        addlug(lug, data1)
        addsafty(safty, data1)
        if vip == 'unacc':
            datavip[0] = datavip[0]+1
            addcount(count, dataunacc)
            addmaint(maint, dataunacc)
            adddoor(door, dataunacc)
            addperson(persons, dataunacc)
            addlug(lug, dataunacc)
            addsafty(safty, dataunacc)
        elif vip == 'acc':
            datavip[1] = datavip[1] + 1
            addcount(count, dataacc)
            addmaint(maint, dataacc)
            adddoor(door, dataacc)
            addperson(persons, dataacc)
            addlug(lug, dataacc)
            addsafty(safty, dataacc)
        elif vip == 'good':
            datavip[2] = datavip[2] + 1
            addcount(count, datagood)
            addmaint(maint, datagood)
            adddoor(door, datagood)
            addperson(persons, datagood)
            addlug(lug, datagood)
            addsafty(safty, datagood)
        elif vip == 'vgood':
            datavip[3] = datavip[3] + 1
            addcount(count, dataVgood)
            addmaint(maint, dataVgood)
            adddoor(door, dataVgood)
            addperson(persons, dataVgood)
            addlug(lug, dataVgood)
            addsafty(safty, dataVgood)


##上面函数的仔函数

def addcount(count,data):
    if count == 'vhigh':
```

```
        data[0][0] = data[0][0]+1
    elif count == 'high':
        data[0][1] = data[0][1]+1
    elif count == 'med':
        data[0][2] = data[0][2] + 1
    elif count == 'low':
        data[0][3] = data[0][3] + 1
def addmaint(maint,data):
    if maint == 'vhigh':
        data[1][0] = data[1][0]+1
    elif maint == 'high':
        data[1][1] = data[1][1] + 1
    elif maint == 'med':
        data[1][2] = data[1][2] + 1
    elif maint == 'low':
        data[1][3] = data[1][3] + 1
def adddoor(door,data):
    if door == '2':
        data[2][0] = data[2][0] + 1
    elif door == '3':
        data[2][1] = data[2][1] + 1
    elif door == '4':
        data[2][2] = data[2][2] + 1
    elif door == '5more':
        data[2][3] = data[2][3] + 1
def addperson(persons,data):
    if persons == '2':
        data[3][0] = data[3][0] + 1
    elif persons == '4':
        data[3][1] = data[3][1] + 1
    elif persons == 'more':
        data[3][2] = data[3][2] + 1
def addlug(lug,data):
    if lug == 'small':
        data[4][0] = data[4][0] + 1
    elif lug == 'med':
        data[4][1] = data[4][1] + 1
    elif lug == 'big':
        data[4][2] = data[4][2] + 1
def addsafty(safty,data):
    if safty == 'low':
        data[5][0] = data[5][0] + 1
    elif safty == 'med':
        data[5][1] = data[5][1] + 1
```

```
    elif safty == 'high':
        data[5][2] = data[5][2] + 1

# 将具体的个数转化为概率
def getP(num, data):
    for k in range(len(data)):
        for ii in range(len(data[k])):
            data[k][ii] = data[k][ii]/num


'''
**************************************
测试部分
**************************************
'''

# 以下的函数为概率返回函数，再检验时起到查表的作用
def getR0(data, t):
        if t == 'vhigh':
            return data[0][0]
        elif t == 'high':
            return data[0][1]
        elif t == 'med':
            return data[0][2]
        elif t == 'low':
            return data[0][3]


def getR1(data, maint):
    if maint == 'vhigh':
        return data[1][0]
    elif maint == 'high':
        return data[1][1]
    elif maint == 'med':
        return data[1][2]
    elif maint == 'low':
        return data[1][3]
    return 0


def getR2(data, door):
    if door == '2':
        return data[2][0]
    elif door == '3':
```

```
        return data[2][1]
    elif door == '4':
        return data[2][2]
    elif door == '5more':
        return data[2][3]


def getR3(data, persons):
    if persons == '2':
        return data[3][0]
    elif persons == '4':
        return data[3][1]
    elif persons == 'more':
        return data[3][2]


def getR4(data, lug):
    if lug == 'small':
        return data[4][0]
    elif lug == 'med':
        return data[4][1]
    elif lug == 'big':
        return data[4][2]


def getR5(data, safty):
    if safty == 'low':
        return data[5][0]
    elif safty == 'med':
        return data[5][1]
    elif safty == 'high':
        return data[5][2]


def getR6(num):
    if num == 0:
        return "unacc"
    if num == 1:
        return "acc"
    if num == 2:
        return "good"
    if num == 3:
        return "vgood"
```

```
rate = [0,0]

# 检验函数
def test(testset):
    for line in testset:
        rate0 =
datavip[0]*getR0(dataunacc,line[0])*getR1(dataunacc,line[1])*getR2(dataunacc,line[2])*getR3(dataunacc,line[3])*g
etR4(dataunacc,line[4])*getR5(dataunacc,line[5])
        rate1 =
datavip[1]*getR0(dataacc,line[0])*getR1(dataacc,line[1])*getR2(dataacc,line[2])*getR3(dataacc,line[3])*getR4(dat
aacc,line[4])*getR5(dataacc,line[5])
        rate2 =
datavip[2]*getR0(datagood,line[0])*getR1(datagood,line[1])*getR2(datagood,line[2])*getR3(datagood,line[3])*getR4
(datagood,line[4])*getR5(datagood,line[5])
        rate3 =
datavip[3]*getR0(dataVgood,line[0])*getR1(dataVgood,line[1])*getR2(dataVgood,line[2])*getR3(dataVgood,line[3])*g
etR4(dataVgood,line[4])*getR5(dataVgood,line[5])
        k = getbig(rate0,rate1,rate2,rate3)
        if line[6] != getR6(k):
            rate[0] = rate[0]+1
        else:
            rate[1] = rate[1]+1

'''
****************************************
总控函数和一些轮子函数
****************************************
'''

##返回四个数中的最大值的下表
def getbig(r1,r2,r3,r4):
    k = max(r1, r2, r3, r4)
    if k == r1:
        return 0
    if k == r2:
        return 1
    if k == r3:
        return 2
    if k == r4:
        return 3

# 初始化函数，初始化数组中的值
def renum():
    data1 = [[0.0, 0.0, 0.0, 0.0], [0.0, 0.0, 0.0, 0.0], [0.0, 0.0, 0.0, 0.0], [0.0, 0.0, 0.0], [0.0, 0.0, 0.0],
```

```
            [0.0, 0.0, 0.0]]
    dataunacc = [[0.0, 0.0, 0.0, 0.0], [0.0, 0.0, 0.0, 0.0], [0.0, 0.0, 0.0, 0.0], [0.0, 0.0, 0.0], [0.0, 0.0, 0.0],
                 [0.0, 0.0, 0.0]]
    dataacc = [[0.0, 0.0, 0.0, 0.0], [0.0, 0.0, 0.0, 0.0], [0.0, 0.0, 0.0, 0.0], [0.0, 0.0, 0.0], [0.0, 0.0, 0.0],
               [0.0, 0.0, 0.0]]
    datagood = [[0.0, 0.0, 0.0, 0.0], [0.0, 0.0, 0.0, 0.0], [0.0, 0.0, 0.0, 0.0], [0.0, 0.0, 0.0], [0.0, 0.0, 0.0],
                [0.0, 0.0, 0.0]]
    dataVgood = [[0.0, 0.0, 0.0, 0.0], [0.0, 0.0, 0.0, 0.0], [0.0, 0.0, 0.0, 0.0], [0.0, 0.0, 0.0], [0.0, 0.0, 0.0],
                 [0.0, 0.0, 0.0]]
    datavip = [0, 0, 0, 0]

##总函数，调用上面的所有函数
def training(size):
    renum()
    trainSet, testSet = randDivision(dataset, size)
    stat(trainSet)
    getP(datavip[0], dataunacc)
    getP(datavip[1], dataacc)
    getP(datavip[2], datagood)
    getP(datavip[3], dataVgood)
    test(testSet)
    Rate = rate[1] / (rate[0] + rate[1])
    print("当训练数据大小为{0}个时，剩余数据的检测正确率为{1}".format(size, Rate))

# 返回合并后的训练集
def getTrainTest(kkk, datas):
    kkkk = 0
    traindata = []
    for i in datas:
        if kkkk == kkk:
            continue
        kkkk = kkkk+1
        traindata = traindata + i
    return traindata

# 求平均值函数

# 十折交叉验证函数
def TenFordCV(size,nums):
    renum()
    s = int(size/nums)
    dataForTfcv = []
    kkk = 0
    Rate = []
```

```
    testSet = dataset
    for i in range(nums-1):
        trainSet, testSet = randDivision(testSet, s)
        dataForTfcv.append(trainSet)
    dataForTfcv.append(testSet)
    for i in dataForTfcv:
        renum()
        testSetForTen = getTrainTest(kkk, dataForTfcv)
        kkk = kkk+1
        stat(testSetForTen)
        getP(datavip[0], dataunacc)
        getP(datavip[1], dataacc)
        getP(datavip[2], datagood)
        getP(datavip[3], dataVgood)
        test(i)
        Rate.append(rate[1] / (rate[0] + rate[1]))
        #print("第{}次正确率为{}".format(kkk,rate[1] / (rate[0] + rate[1])))
    return np.mean(Rate)



if __name__=="__main__":
    name = "..\..\data\car.csv"
    dataset = loadcsv(name)
    training(100)
    training(200)
    training(500)
    training(700)
    training(1350)
    ppp = input("想要进行多少层十折交叉验证")
    r = []
    for i in range(int(ppp)):
        r.append(TenFordCV(1728, 10))
    print("经{}层十折交叉验证，该朴素贝叶斯分类器平均分类正确率为{}".format(ppp, np.mean(r)))
```