实验一 设计贝叶斯分类器对 iris 数据进行分类

一、实验目的:

1.通过熟悉鸢尾花数据集，理解特征、样本、类别等基本概念。

2.通过对鸢尾花数据集进行分类，掌握并理解统计模式识别的原理及实现方法。

二、实验内容及要求:

1.利用所提供的训练数据，完成基本最小错误率的贝叶斯分类器的设计，并用测试数据进行测试，计算出错误率。

2.再使用最小风险判别准则进行分类，实验中假设风险参数矩阵为 L，该数据可根据实际损失的情况需要进行修改。

这里给定损失参数矩阵为:[0,2,1;

3,0,4;

1,2,0]　损失参数矩阵可以调整.

3.改变损失矩阵对分类结果是否会有影响？给出不同的两组损失矩阵得到的分类结果。

4.使用 python 语言来完成实验

三、实验原理

实验数据：IRIS 数据。分为三种类型，每种类型中包括 50 个四维的向量。

实验模型：假设 IRIS 数据是正态分布的。

实验准备：在每种类型中，选择 45 个向量作为训练样本，估计未知的均值和方差的参数。

实验方法：最小错误判别准则；最小风险判别准则。

实验原理：

1.贝叶斯公式

已知共有 M 类别 $\omega_i, i=1,2,\cdots M$，统计分布为正态分布，已知先验概率 $P(\omega_i)$ 及类条件概率密度函数 $P(X|\omega_i)$，对于待测样品，贝叶斯公式可以计算出该样品分属各类别的概率，叫做后验概率；看 X 属于哪个类的可能性最大，就把 X 归于可能性最大的那个类，后验概率即为识别对象归属的依据。贝叶斯公式为

$$P(\omega_i \mid X) = \frac{P(X \mid \omega_i) P(\omega_i)}{\sum_{j=1}^{M} P(X \mid \omega_j) P(\omega_j)}, i = 1, 2, \cdots M$$

该公式体现了先验概率、类条件概率、后验概率三者的关系。

其中，**类条件概率密度函数** $P(X \mid \omega_i)$ 为正态密度函数，用大量样本对其中未知参数进行估计，多维正态密度函数为

$$P(X) = \frac{1}{(2\pi)^{n/2} |S|^{1/2}} \exp[-\frac{1}{2}(X-\mu)^T S^{-1}(X-\mu)]$$

式中，$X = (x_1, x_2, \cdots x_n)$ 为 n 维向量；

$\mu = (\mu_1, \mu_2, \cdots \mu_n)$ 为 n 维均值向量；

$S = E[(X-\mu)(X-\mu)^T]$ 为 n 维协方差矩阵；

S^{-1} 是 S 的逆矩阵；

$|S|$ 是 S 的行列式。

大多数情况下，类条件密度可以采用多维变量的正态密度函数来模拟。

$$P(X \mid \omega_i) = \ln\{\frac{1}{(2\pi)^{n/2} |S_i|^{1/2}} \exp[-\frac{1}{2}(X - \overline{X^{(\omega_i)}})^T S_i^{-1} (X - \overline{X^{(\omega_i)}})]\}$$

$$= -\frac{1}{2}(X - \overline{X^{(\omega_i)}})^T S_i^{-1} (X - \overline{X^{(\omega_i)}}) - \frac{n}{2}\ln 2\pi - \frac{1}{2}\ln|S_i|$$

$\overline{X^{(\omega_i)}}$ 为 ω_i 类的均值向量。

2. 最小错误判别准则

① 两类问题

有两种形式，似然比形式：

$$l(X) = \frac{P(X \mid \omega_1)}{P(X \mid \omega_2)} \begin{cases} > \\ < \end{cases} \frac{P(\omega_2)}{P(\omega_1)} \Rightarrow X \in \begin{cases} \omega_1 \\ \omega_2 \end{cases}$$

其中，$l(X)$ 为似然比，$\frac{P(\omega_2)}{P(\omega_1)}$ 为似然比阈值。

对数形式：

$$\ln P(X \mid \omega_1) - \ln P(X \mid \omega_2) \begin{cases} > \\ < \end{cases} \ln P(\omega_2) - \ln P(\omega_1) \Rightarrow X \in \begin{cases} \omega_1 \\ \omega_2 \end{cases}$$

② 多类问题

本实验采取针对多累问题的解决方法。在待测向量 X 的条件下，看哪个类的概率最大，应该把 X 归于概率最大的那个类。因此，可以通过比较各个判别函数来确定 X 的类型。

$$P(\omega_i)P(X\mid\omega_i)=\max_{1\le j\le M}\{P(\omega_j)P(X\mid\omega_j)\}\Rightarrow X\in\omega_i, i=1,2,\cdots M$$

对数形式为：

$$\ln P(\omega_i)+\ln P(X\mid\omega_i)=\max_{1\le j\le M}\{\ln P(\omega_j)+\ln P(X\mid\omega_j)\}\Rightarrow X\in\omega_i, i=1,2,\cdots M$$

所以此时正态分布的贝叶斯分类器判别函数为

$$h_i(X)=\ln P(X\mid\omega_i)P(\omega_i)=\ln\{\frac{1}{(2\pi)^{n/2}\left|S_i\right|^{1/2}}\exp[-\frac{1}{2}(X-\overline{X^{(\omega_i)}})^T S_i^{-1}(X-\overline{X^{(\omega_i)}})]\}P(\omega_i)$$

$$=-\frac{1}{2}(X-\overline{X^{(\omega_i)}})^T S_i^{-1}(X-\overline{X^{(\omega_i)}})-\frac{n}{2}\ln 2\pi-\frac{1}{2}\ln\left|S_i\right|+\ln P(\omega_i)$$

3. 最小风险判别准则

对观测值 X 条件下，各状态后验概率求加权和的方式，表示风险如下：

$$R_i(X)=\sum_{j=1}^{M}L(i,j)P(\omega_j\mid X)$$

其中，$L(i,j)$ 为将第 j 类判为第 i 类的损失。若判对 i=j，则 $L(i,j)$ 取负值或零值，表示没有损失；若判对 i≠j，则 $L(i,j)$ 取正值，数值大小表示损失多少。

对得到的 M 个类型的风险值 $R_i(X), i=1,2,\cdots M$ 进行比较，得到使条件风险最小的类别，判别 X 属于该类别。

四、实验代码和结果

实验代码:

```
import numpy as np
# 将列表中的数据切片读入矩阵
def Read(lines,m,n):
    A = np.zeros((m, n), dtype=float)
    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


'''
1.读取训练集和测试集
注:在此处统计行数是为了兼容不同的样本集,因为理论上说我们事先不会知晓有多少组数据
'''
f1 = open('F:\\Code\\Mode Regonization\\Iris\\train.txt') # 打开训练集
f2 = open('F:\\Code\\Mode Regonization\\Iris\\test.txt') # 打开测试集
lines1 = f1.readlines() # 把全部数据文件读到一个列表 lines 中
lines2 = f2.readlines()


Line1 = len(lines1) # 读取训练集行数
Line2 = len(lines2) # 读取测试集列数
print(Line1, Line2)
A = Read(lines1,Line1,5)
B = Read(lines2,Line2,5)

'''
2.将三类训练样本拆分
这里为了提高效率默认我们已知哪几行是第一类、第二类、第三类
若放在普遍场景,则需进行遍历逐行分类拆分

Axy 代表第 x 类样本的第 y 个特性
三类样本分别为:Setosa、Versicolour、Virginica
四类特性分别为:花萼长度、花萼宽度、花瓣长度、花瓣宽度
'''
A1 = A[0:25]
```

```python
A1 = np.delete(A1,0, axis=1) # 删除第一列(类别号)
IIII
A2 = np.delete(A2,0, axis=1) # 删除第一列(类别号)
A3 = A[50:75]
A3 = np.delete(A3,0, axis=1) # 删除第一列(类别号)


'''
3.计算样本期望μ
按列求得均值，得到三类的期望值
'''
# 将数据集矩阵输入求解均值，并输出均值向量
def Mean(A):
    mean = np.average(A, axis=0) # 按列求均值
    mean = mean.transpose() # 将矩阵转置
    return mean

mean1 = Mean(A1)
mean2 = Mean(A2)
mean3 = Mean(A3)
'''此时每一个 mean 都是对应类别的均值列向量'''


'''
4.计算样本协方差矩阵
rowvar=False 代表把每一列看做一组变量
本实验中有四组变量因此返回值必为 4*4 矩阵
'''
cov1 = np.cov(A1,rowvar=False)
cov2 = np.cov(A2,rowvar=False)
cov3 = np.cov(A3,rowvar=False)



'''
5.
基于最小错误率:
构建正态分布的概率密度函数，计算测试集的类条件概率
因为所有类别的全概率和先验概率相同，所以仅需比较类条件概率大小
用以替代后验概率的比较
在本题目中
f(x)=e^( -0.5 * (x-μ)^T * (Σ^-1) * (x-μ))/(4π²*|Σ|^0.5)
两边取对数可得
ln[f(x)] = -0.5 * (x-μ)^T * (Σ^-1) * (x-μ) - ln(4π² * |Σ|^0.5 )
         = -0.5 * (x-μ)^T * (Σ^-1) * (x-μ) - 0.5*ln(|Σ|) - ln(4π²)
```

```
可以看出不论是哪一类的概率密度函数都有一个- ln(4π²)，可以省略成
g(x) = -0.5 * (x-μ)^T * (Σ^-1) * (x-μ) - 0.5*ln(|Σ|)
两边同乘 2 可得
p(x) = -(x-μ)^T * (Σ^-1) * (x-μ) - ln(|Σ|)
p(x)越大,g(x)越大,从而 f(x)越大
'''
def get_pdf(x,mean,cov):
    # 计算协方差的行列式
    det = np.linalg.det(cov)
    # 计算协方差的逆矩阵
    # numpy 中的 linalg 模块包含大量线性代数中的函数方法
    cov_inv = np.linalg.inv(cov)
    # 也可以使用**-1 幂运算代表逆矩阵
    # cov_inv = cov**-1
    '''用 t 代表 x-μ'''
    t = x-mean
    p = np.dot( np.dot(-t.transpose(),cov_inv),t ) - np.log(det)
    return p

true = 0
false = 0
for i in range(0,Line2):
    B_row = B[[i]] # 获取第 i 行
    id = B_row[0,0] # 获取文件中的标号
    B_row = np.delete(B_row,0,axis=1) # 删除第一列(类别号)
    B_row = B_row.flatten() # 平铺成列向量


    res1 = get_pdf(B_row,mean1,cov1)
    res2 = get_pdf(B_row,mean2,cov2)
    res3 = get_pdf(B_row,mean3,cov3)
    # print(res1)
    if max(res1,res2)==res1:
        if max(res1,res3)==res1:
            Id = 1.0
        else:
            Id = 3.0
    else:
        if max(res2,res3)==res2:
            Id = 2.0
        else:
            Id = 3.0
    if(id==Id):
        true+=1
    else:
```

```python
        false+=1
print("基于最小错误率：")
print("正确个数：",true)
print("错误个数：",false)
print("准确率：",true/(true+false))


'''
6.基于最小风险率
在本题目中，R(i) = L[i,1]*P(w1|x)+L[i,2]*P(w2|x)+L[i,3]*P(w3|x)
而对所有类别的全概率和先验概率相同，所以上式的 P(w1|x)可以等价替换为 P(x|w1)，以此类推
而每一类的概率密度函数均有(2π)^(n/2)这一项，因此这一部分也可以忽略
f(x)=e^( -0.5 * (x-μ)^T * (∑^-1) * (x-μ))/(4π²*|∑|^0.5)
最终式子变为：
R(i) = L[i,1] * e^( -0.5 * (x-μ1)^T * (∑1^-1) * (x-μ1))/(|∑1|^0.5) +
       L[i,2] * e^( -0.5 * (x-μ2)^T * (∑2^-1) * (x-μ1))/(|∑2|^0.5) +
       L[i,3] * e^( -0.5 * (x-μ3)^T * (∑3^-1) * (x-μ1))/(|∑3|^0.5)
这里将给出不同的两组损失矩阵得到的分类结果。
'''

true1 = false1 = 0
true2 = false2 = 0
# 计算上式的等价后验概率
def get_pdf2(x,mean,cov):
    # 计算协方差的行列式
    det = np.linalg.det(cov)
    # 计算协方差的逆矩阵
    # numpy 中的 linalg 模块包含大量线性代数中的函数方法
    cov_inv = np.linalg.inv(cov)
    # 也可以使用**-1 幂运算代表逆矩阵
    # cov_inv = cov**-1
    '''用 t 代表 x-μ'''
    t = x-mean
    p = np.exp(-
0.5*np.dot( np.dot(t.transpose(),cov_inv),t ))/pow(det,0.5)
    return p
# 找到风险最小的类别号
def find_min_risk(R1,R2,R3):
    if min(R1,R2)==R1:
        if min(R1,R3)==R1:
            Id = 1.0
        else:
            Id = 3.0
    else:
```

```
        if min(R2,R3)==R2:
            Id = 2.0
        else:
            Id = 3.0
    return Id
# 计算准确率
def count_accuracy(true,false,id,Id):
    if id==Id:
        true+=1
    else:
        false+=1
    return true,false

L1 = np.array([[0,2,1],[3,0,4],[1,2,0]]) # 导入第一组损失参数矩阵
L2 = np.array([[0,1,1],[1,0,6],[1,2,0]]) # 将选错的代价损失减小
for i in range(0,Line2):
    B_row = B[[i]] # 获取第 i 行
    id = B_row[0,0] # 获取文件中的标号
    B_row = np.delete(B_row,0,axis=1) # 删除第一列(类别号)
    B_row = B_row.flatten() # 平铺成列向量
    res1 , res2 , res3 = get_pdf2(B_row,mean1,cov1) ,
get_pdf2(B_row,mean2,cov2) , get_pdf2(B_row,mean3,cov3)
    res = [res1,res2,res3]
    # 第一组的相关数据
    R11 , R21 , R31 = sum(res*L1[0]) , sum(res*L1[1]) , sum(res*L1[2])
    Id1 = find_min_risk(R11,R21,R31)
    true1 , false1 = count_accuracy(true1,false1,id,Id1)
    # 第二组的相关数据
    R12 , R22 , R32 = sum(res*L2[0]) , sum(res*L2[1]) , sum(res*L2[2])
    Id2 = find_min_risk(R12,R22,R32)
    true2 , false2 = count_accuracy(true2,false2,id,Id2)

print("基于最小风险率: ")
print("第一组损失参数矩阵结果: ")
print("正确个数: ",true1)
print("错误个数: ",false1)
print("准确率: ",true1/(true1+false1))
print("第二组损失参数矩阵结果: ")
print("正确个数: ",true2)
print("错误个数: ",false2)
print("准确率: ",true2/(true2+false2))
```

结果:

```
75 75
基于最小错误率:
正确个数:  74
错误个数:  1
准确率:  0.9866666666666667
基于最小风险率:
第一组损失参数矩阵结果:
正确个数:  74
错误个数:  1
准确率:  0.9866666666666667
第二组损失参数矩阵结果:
正确个数:  73
错误个数:  2
准确率:  0.9733333333333334
```

①贝叶斯分类器能够对绝大多数的数据成功分类，偶尔会有错误出现

②当风险矩阵的损失量减小时，使用最小风险的决策方法会增加错误率，因为当损失量减小时，我们选错的代价就会减小，因此风险值就会变小，导致部分数据的异常。