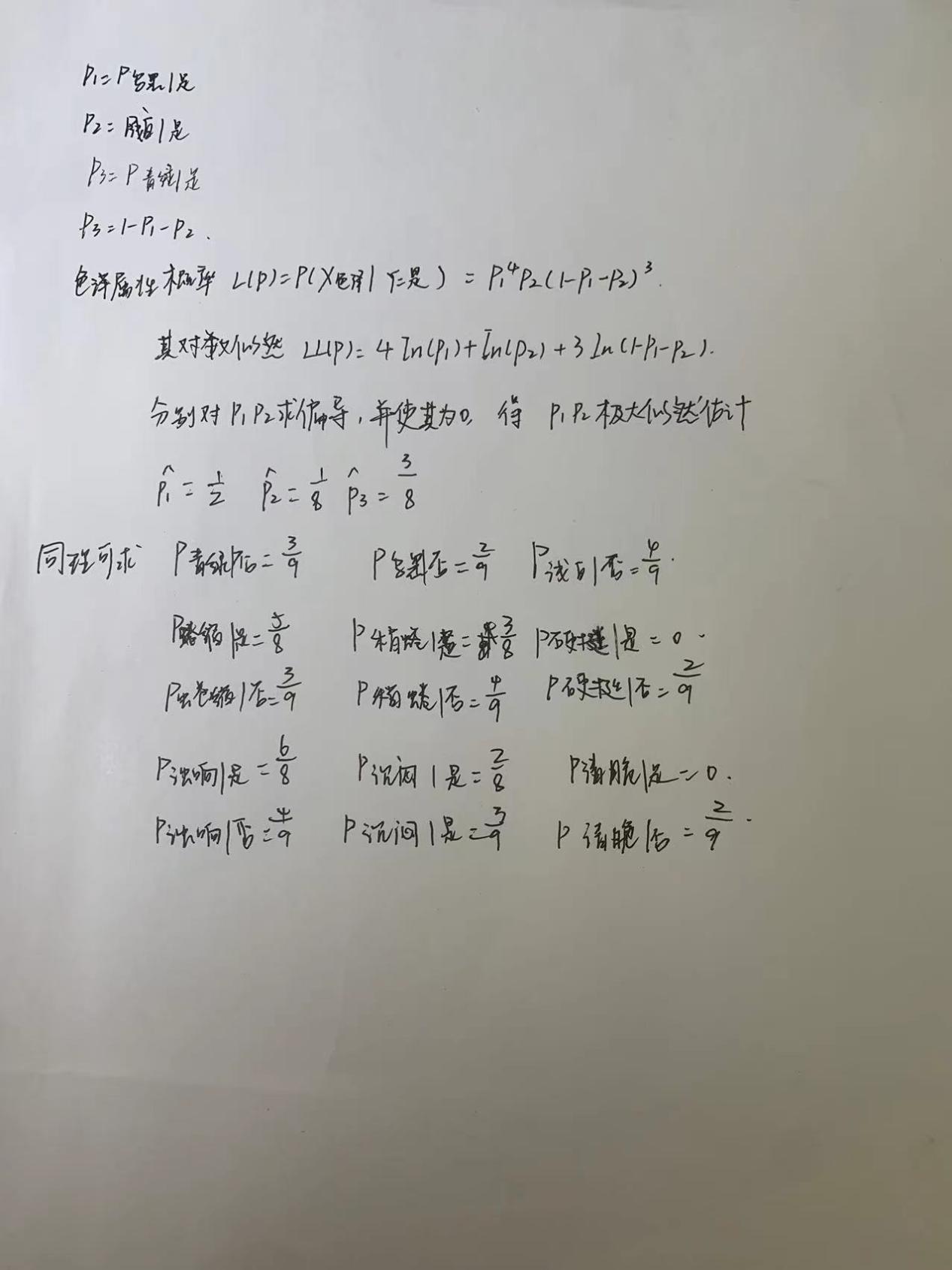
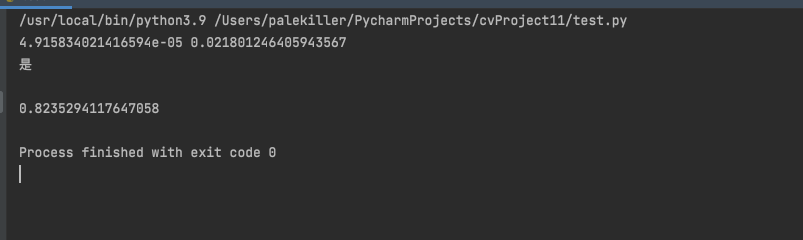
**7.1试使用极大似然法估算西瓜数据集3.0中前3个属性的类条件概率。**

西瓜数据集

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 色泽 | 根蒂 | 敲声 | 纹理 | 脐部 | 触感 | 密度 | 含糖率 | 好瓜 |
| 1 | 青绿 | 蜷缩 | 浊响 | 清晰 | 凹陷 | 硬滑 | 0.697 | 0.46 | 是 |
| 2 | 乌黑 | 蜷缩 | 沉闷 | 清晰 | 凹陷 | 硬滑 | 0.774 | 0.376 | 是 |
| 3 | 乌黑 | 蜷缩 | 浊响 | 清晰 | 凹陷 | 硬滑 | 0.634 | 0.264 | 是 |
| 4 | 青绿 | 蜷缩 | 沉闷 | 清晰 | 凹陷 | 硬滑 | 0.608 | 0.318 | 是 |
| 5 | 浅白 | 蜷缩 | 浊响 | 清晰 | 凹陷 | 硬滑 | 0.556 | 0.215 | 是 |
| 6 | 青绿 | 稍蜷 | 浊响 | 清晰 | 稍凹 | 软粘 | 0.403 | 0.237 | 是 |
| 7 | 乌黑 | 稍蜷 | 浊响 | 稍糊 | 稍凹 | 软粘 | 0.481 | 0.149 | 是 |
| 8 | 乌黑 | 稍蜷 | 浊响 | 清晰 | 稍凹 | 硬滑 | 0.437 | 0.211 | 是 |
| 9 | 乌黑 | 稍蜷 | 沉闷 | 稍糊 | 稍凹 | 硬滑 | 0.666 | 0.091 | 否 |
| 10 | 青绿 | 硬挺 | 清脆 | 清晰 | 平坦 | 软粘 | 0.243 | 0.267 | 否 |
| 11 | 浅白 | 硬挺 | 清脆 | 模糊 | 平坦 | 硬滑 | 0.245 | 0.057 | 否 |
| 12 | 浅白 | 蜷缩 | 浊响 | 模糊 | 平坦 | 软粘 | 0.343 | 0.099 | 否 |
| 13 | 青绿 | 稍蜷 | 浊响 | 稍糊 | 凹陷 | 硬滑 | 0.639 | 0.161 | 否 |
| 14 | 浅白 | 稍蜷 | 沉闷 | 稍糊 | 凹陷 | 硬滑 | 0.657 | 0.198 | 否 |
| 15 | 乌黑 | 稍蜷 | 浊响 | 清晰 | 稍凹 | 软粘 | 0.36 | 0.37 | 否 |
| 16 | 浅白 | 蜷缩 | 浊响 | 模糊 | 平坦 | 硬滑 | 0.593 | 0.042 | 否 |
| 17 | 青绿 | 蜷缩 | 沉闷 | 稍糊 | 稍凹 | 硬滑 | 0.719 | 0.103 | 否 |

****

**7.3试编程实现拉普拉斯修正的朴素贝叶斯分类器，并以西瓜数据集3.0为训练集，对p.151“测1”样本进行判别。**



# coding=gbk  
import math  
import numpy as np  
  
data\_ = [  
 ['青绿','蜷缩','浊响','清晰','凹陷','硬滑',0.697,0.460,'是'],  
 ['乌黑','蜷缩','沉闷','清晰','凹陷','硬滑',0.774,0.376,'是'],  
 ['乌黑','蜷缩','浊响','清晰','凹陷','硬滑',0.634,0.264,'是'],  
 ['青绿','蜷缩','沉闷','清晰','凹陷','硬滑',0.608,0.318,'是'],  
 ['浅白','蜷缩','浊响','清晰','凹陷','硬滑',0.556,0.215,'是'],  
 ['青绿','稍蜷','浊响','清晰','稍凹','软粘',0.403,0.237,'是'],  
 ['乌黑','稍蜷','浊响','稍糊','稍凹','软粘',0.481,0.149,'是'],  
 ['乌黑','稍蜷','浊响','清晰','稍凹','硬滑',0.437,0.211,'是'],  
 ['乌黑','稍蜷','沉闷','稍糊','稍凹','硬滑',0.666,0.091,'否'],  
 ['青绿','硬挺','清脆','清晰','平坦','软粘',0.243,0.267,'否'],  
 ['浅白','硬挺','清脆','模糊','平坦','硬滑',0.245,0.057,'否'],  
 ['浅白','蜷缩','浊响','模糊','平坦','软粘',0.343,0.099,'否'],  
 ['青绿','稍蜷','浊响','稍糊','凹陷','硬滑',0.639,0.161,'否'],  
 ['浅白','稍蜷','沉闷','稍糊','凹陷','硬滑',0.657,0.198,'否'],  
 ['乌黑','稍蜷','浊响','清晰','稍凹','软粘',0.360,0.370,'否'],  
 ['浅白','蜷缩','浊响','模糊','平坦','硬滑',0.593,0.042,'否'],  
 ['青绿','蜷缩','沉闷','稍糊','稍凹','硬滑',0.719,0.103,'否'],  
]  
  
is\_discrete = [True] \* 6 + [False] \* 2  
  
set\_list = [set() for i in range(8)]  
for d in data\_:  
 for i in range(8):  
 set\_list[i].add(d[i])  
features\_list = [[] for i in range(8)]  
for i in range(8):  
 features\_list[i] = list(set\_list[i])  
  
data = np.mat(data\_)  
  
labels = np.unique(data[:, -1].A)  
cnt\_labels = [0] \* len(labels)  
for i in range(data.shape[0]):  
 if data[i, -1] == labels[0]:  
 cnt\_labels[0] = cnt\_labels[0] + 1  
 elif data[i, -1] == labels[1]:  
 cnt\_labels[1] = cnt\_labels[1] + 1  
  
def train\_discrete(data, labels, cnt\_labels, features\_list, xi):  
 prob = np.ones([len(labels), np.unique(data[:, xi].A).shape[0]])  
 for i in range(data.shape[0]):  
 tmp = features\_list[xi].index(data[i, xi])  
 if data[i, -1] == labels[0]:  
 prob[0, tmp] = prob[0, tmp] + 1  
 elif data[i, -1] == labels[1]:  
 prob[1, tmp] = prob[1, tmp] + 1  
 for i in range(len(labels)):  
 prob[i] = prob[i] / (cnt\_labels[i] + len(features\_list[xi]))  
 return prob  
  
def train\_continuous(data, labels, xi):  
 vec0, vec1 = [], []  
 for i in range(data.shape[0]):  
 if data[i, -1] == labels[0]:  
 vec0.append(data[i, xi])  
 elif data[i, -1] == labels[1]:  
 vec1.append(data[i, xi])  
 vec0, vec1 = np.array(vec0).astype(float), np.array(vec1).astype(float)  
 u0, u1 = np.mean(vec0), np.mean(vec1)  
 s0, s1 = np.var(vec0), (np.var(vec1))  
 return np.mat([[u0, s0], [u1, s1]])  
  
param = []  
for i in range(8):  
 if is\_discrete[i]:  
 param.append(train\_discrete(data, labels, cnt\_labels, features\_list, i))  
 else:  
 param.append(train\_continuous(data, labels, i))  
  
p0 = (cnt\_labels[0] + 1) / (len(data\_) + 2)  
p1 = (cnt\_labels[1] + 1) / (len(data\_) + 2)  
d = data\_[0]  
for i in range(len(d) - 1):  
 if is\_discrete[i]:  
 ind = features\_list[i].index(d[i])  
 p0 \*= param[i][0, ind]  
 p1 \*= param[i][1, ind]  
 else:  
 p0 \*= 1 / (math.sqrt(2 \* math.pi \* param[i][0, 1])) \* math.exp(-(d[i] - param[i][0, 0])\*\*2 / (2 \* param[i][0,1]))  
 p1 \*= 1 / (math.sqrt(2 \* math.pi \* param[i][1, 1])) \* math.exp(-(d[i] - param[i][1, 0]) \*\* 2 / (2 \* param[i][1, 1]))  
print(p0, p1)  
if p0 > p1:  
 print(labels[0])  
else:  
 print(labels[1])  
  
print()  
err = 0  
for d in data\_:  
 p0 = (cnt\_labels[0] + 1) / (len(data\_) + 2)  
 p1 = (cnt\_labels[1] + 1) / (len(data\_) + 2)  
 for i in range(len(d) - 1):  
 if is\_discrete[i]:  
 ind = features\_list[i].index(d[i])  
 p0 \*= param[i][0, ind]  
 p1 \*= param[i][1, ind]  
 else:  
 p0 \*= 1 / (math.sqrt(2 \* math.pi \* param[i][0, 1])) \* math.exp(  
 -(d[i] - param[i][0, 0]) \*\* 2 / (2 \* param[i][0, 1]))  
 p1 \*= 1 / (math.sqrt(2 \* math.pi \* param[i][1, 1])) \* math.exp(  
 -(d[i] - param[i][1, 0]) \*\* 2 / (2 \* param[i][1, 1]))  
 plabel = None  
 if p0 > p1:  
 plabel = labels[0]  
 else:  
 plabel = labels[1]  
 if plabel != d[-1]:  
 err += 1  
print(1 - err / len(data\_))