机器学习第七章作业

任齐轩
重庆大学-计卓 2 班-20204154

日期：October 23, 2022

1 第一题

色泽：青绿、乌黑、浅白
1. 青绿：$P(Qinglv \mid True) = \frac{3}{8}, P(Qinglv \mid False) = \frac{3}{9}$
2. 乌黑：$P(Wuhei \mid True) = \frac{4}{8}, P(Wuhei \mid False) = \frac{2}{9}$
3. 浅白：$P(Qianbai \mid True) = \frac{1}{8}, P(Qianbai \mid False) = \frac{4}{9}$
根蒂：蜷缩、稍蜷、硬挺
1. 蜷缩：$P(Quansuo \mid True) = \frac{5}{8}, P(Quansuo \mid False) = \frac{3}{9}$
2. 稍蜷：$P(Shaoquan \mid True) = \frac{3}{8}, P(Shaoquan \mid False) = \frac{4}{9}$
3. 硬挺：$P(Yingting \mid True) = \frac{0}{8}, P(Yingting \mid False) = \frac{2}{9}$
敲声：浊响、沉闷、清脆
1. 浊响：$P(Zhuoxiang \mid True) = \frac{6}{8}, P(Zhuoxiang \mid False) = \frac{4}{9}$
2. 沉闷：$P(Chenmen \mid True) = \frac{2}{8}, P(Chenmen \mid False) = \frac{3}{9}$
3. 清脆：$P(Qingcui \mid True) = \frac{0}{8}, P(Qingcui \mid False) = \frac{2}{9}$

2 第二题

（书本 P164）
1. 对分类任务来说，只需各类别的条件概率排序正确、无须精准概率值即可导致正确分类结果。
2. 若属性间依赖对所有类别影响相同，或依赖关系的影响能够相互抵消，则属性条件独立性假设在降低计算开销的同时不会对性能产生负面影响。

3 第三题 (代码见附录)

采用拉普拉斯修正后，编程实现朴素贝叶斯分类器，对西瓜数据集 3.0 进行分类，得到的分类结果如下图所示，可以看出，使用拉普拉斯修正后，分类结果与书本 P151 所给出的结果一致。

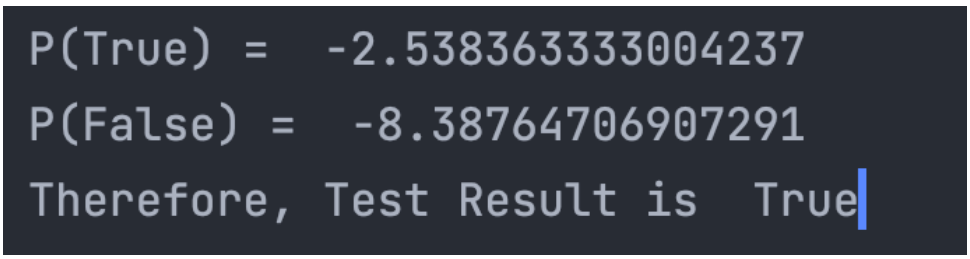

图 1: p.151 页" 测 1" 判别结果

4 第四题

使用对数似然 (log-likelihood)，即:

$$h_{nb}(x) = argmax_{c \in y}[\log(c) + \sum_{i=1}^{d} \log P(x_i \mid c)] \tag{1}$$

5 第七题

当 $d = 1$ 时，即只有一个属性时，x_2 的取值为 $0, 1$，则有 4 种情况，即 $4 * 30 = 120$，此时最好最坏情况都至少需要 120 个样本

当 $d = 2$ 时，此时多了一个属性 x_2，其取值为 $0, 1$；则最好情况下，只需要 120 个样本；而最坏情况下，x_2 都取 1，则计算 $P(c, x_2 = 0)$ 需要另外 60 个样本，即需要 180 个样本

以此类推，当 $d = n$ 时，最好情况下，需要 120 个样本，最坏情况下，需要 $60 * (n - 1) + 120$ 个样本，即 $60 * (n + 1)$ 个样本

A 第三题代码

```
import numpy as np
import Machine_Learning.Linear_Regression.datasets as datasets


def train(test_data):
    data = datasets.watermelon
    true_res, false_res = 0, 0
    test_count_true = [0 for _ in range(len(test_data) - 2)]
    test_count_false = [0 for _ in range(len(test_data) - 2)]
    data_record_true = [[], []]
    data_record_false = [[], []]
    for temp in data:
        if temp[-1] == "好瓜":
            true_res += 1
            for i in range(len(temp) - 1):
                if type(temp[i]) == str:
                    if temp[i] == test_data[i]:
                        test_count_true[i] += 1
                elif type(temp[i]) == float:
                    data_record_true[(i % 6)].append(temp[i])
        else:
            false_res += 1
            for i in range(len(temp) - 1):
                if type(temp[i]) == str:
                    if temp[i] == test_data[i]:
                        test_count_false[i] += 1
                elif type(temp[i]) == float:
                    data_record_false[(i % 6)].append(temp[i])

    p_true, p_false = ((true_res + 1) / (17.0 + 2.0)), ((false_res + 1) / (17.0
        + 2.0))
    mu_true = [np.mean(data_record_true[0]), np.mean(data_record_true[1])]
    mu_false = [np.mean(data_record_false[0]), np.mean(data_record_false[1])]
    std_true = [np.std(data_record_true[0], ddof=1), np.std(data_record_true
        [1], ddof=1)]
    std_false = [np.std(data_record_false[0], ddof=1), np.std(data_record_false
        [1], ddof=1)]
    for i in range(len(test_data)):
        if i < 6:
            p_true += np.log((test_count_true[i] + 1) / (true_res + 3))
            p_false += np.log((test_count_false[i] + 1) / (false_res + 3))
        else:
            p_true += np.log(np.exp(-np.square(test_data[i] - mu_true[i % 6]) /
                (2*np.square(std_true[i % 6]))) / (np.sqrt(2*np.pi)*std_true[i
                % 6]))
```

```
            p_false += np.log(np.exp(-np.square(test_data[i] - mu_false[i % 6])
                / (2*np.square(std_false[i % 6]))) / (np.sqrt(2*np.pi)*
                std_false[i % 6]))

    print('P(True) = ', p_true)
    print('P(False) = ', p_false)
    res = True if (p_true > p_false) else False
    print('Therefore, Test Result is ', str(res))


def main():
    test_data = ['青绿', '蜷缩', '浊响', '清晰', '凹陷', '硬滑', 0.697, 0.460]
    train(test_data)


if __name__ == '__main__':
    main()
```