山东大学__计算机科学与技术__学院

__机器学习__课程实验报告

学号:201900130059	姓名: 孙奇	班级: 2019 级 1 班
实验题目: Naive Bayes		
实验学时: 2	实验日期: 2021/11/08	

实验目的:

1. 使用 Naive Bayes 在数据集上对结果进行推断;
2. 切分数据集，观察不同大小的 training_data 在 Naive Bayes 模型下结果的变化;

硬件环境:

CPU: Intel i5-9300H

GPU: UHD630

软件环境:

Python3.8

PyCharm CE

实验步骤与内容:

1. Naive Bayes 的多变量分类问题:

（1） Training_data:

- Each training sample involves a different number of features

$$x^{(i)} = [x_1^{(i)}, x_2^{(i)}, \cdots, x_{n_i}^{(i)}]^{\mathrm{T}}$$

- The j-th feature of $x^{(i)}$ takes a finite set of values, $x_j^{(i)} \in \{1, 2, \cdots, v\}$

每一个 training_data 的自变量 x 是一个 n^i维向量, x 的第 j 个 feature x_j有 v 种不同的取值;

（2） 多分类问题的空间为:

$$(\Omega = \{p(y), p(t \mid y)\}_{y\in\{0,1\}, t\in\{1,\cdots,v\}})$$

（3） 极大似然函数:

$$\begin{aligned}
\ell(\Omega) &= \log \prod_{i=1}^{m} p(x^{(i)}, y^{(i)}) \\
&= \sum_{i=1}^{m} \log p(x^{(i)} \mid y^{(i)}) p(y^{(i)}) \\
&= \sum_{i=1}^{m} \log p(y^{(i)}) \prod_{j=1}^{n_i} p(x_j^{(i)} \mid y^{(i)}) \\
&= \sum_{i=1}^{m} \sum_{j=1}^{n_i} \log p(x_j^{(i)} \mid y^{(i)}) + \sum_{i=1}^{m} \log p(y^{(i)})
\end{aligned}$$

（4） 分类问题可建模为:

- Problem formulation

$$\max \quad \ell(\Omega)=\sum_{i=1}^{m}\sum_{j=1}^{n_i}\log p(x_j^{(i)}\mid y^{(i)})+\sum_{i=1}^{m}\log p(y^{(i)})$$

$$s.t. \quad \sum_{y\in\{0,1\}} p(y)=1,$$

$$\sum_{t=1}^{v} p(t\mid y)=1,\ \forall y=0,1$$

$$p(y)\geq 0,\ \forall y=0,1$$

$$p(t\mid y)\geq 0,\ \forall t=1,\cdots,v,\ \forall y=0,1$$

（5） 根据拉格朗日乘数法求的最优化结果：

$$p(t\mid y)=\frac{\sum_{i=1}^{m}\mathbf{1}(y^{(i)}=y)count^{(i)}(t)}{\sum_{i=1}^{m}\mathbf{1}(y^{(i)}=y)\sum_{t=1}^{v}count^{(i)}(t)}$$

$$p(y)=\frac{\sum_{i=1}^{m}\mathbf{1}(y^{(i)}=y)}{m}$$

$$\text{where} \quad count^{(i)}(t)=\sum_{j=1}^{n_i}\mathbf{1}(x_j^{(i)}=t)$$

p(t|y)表示对于第i个training_data，y值(label) = y的情况下，对应的x值(n维向量)，x值的所有feature，即$x_j=t(1<=j<=n_i)$的个数占x的所有可能取值之和$\sum_{t=1}^{v}count^{(i)}(t)$的比例

（6） 引入 Laplace smoothing：

对于n维向量x中的一个feature x_j，可能对于所有的y值，training_data中都不存在该x_j，因此导致：

$$p_{j^*}(x_{j^*}=1\mid y)=\frac{\sum_{i=1}^{m}\mathbf{1}(y^{(i)}=y\wedge x_{j^*}^{(i)}=1)}{\sum_{i=1}^{m}\mathbf{1}(y^{(i)}=y)}=0,\ \forall y=0,1$$

这样会导致后面test过程中，如果出现test_data的n维x向量中$x_j=1$，则：

$$p(y\mid x)=\frac{p(y)\prod_{j=1}^{n}p_j(x_j\mid y)}{\sum_y\prod_{j=1}^{n}p_j(x_j\mid y)p(y)}=\frac{0}{0},\ \forall y=0,1$$

无法得到正确结果，因此引入**Laplace smoothing**

（7） 本问题下 Laplace smoothing 后得到：

- Laplace smoothing

$$\psi(t\mid y)=\frac{\sum_{i=1}^{m}\mathbf{1}(y^{(i)}=y)count^{(i)}(t)+1}{\sum_{i=1}^{m}\mathbf{1}(y^{(i)}=y)\sum_{t=1}^{v}count^{(i)}(t)+v}$$

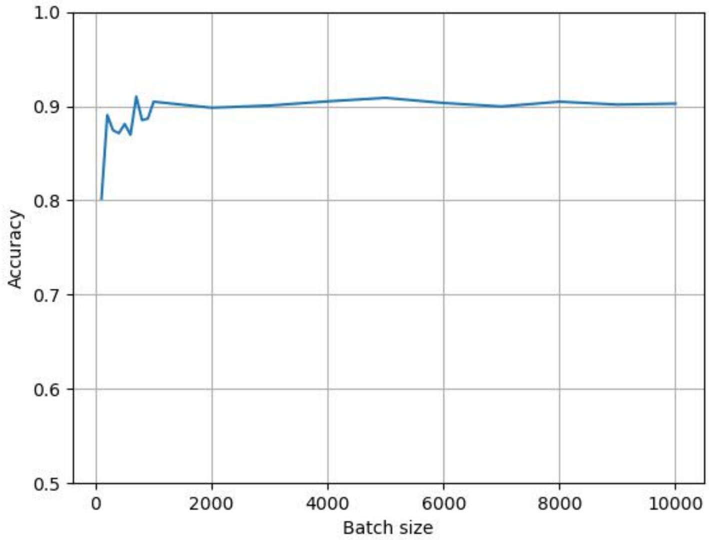
$$\psi(y)=\frac{\sum_{i=1}^{m}\mathbf{1}(y^{(i)}=y)+1}{m+k}$$

2. 训练&预测过程：

（1） 在 training_data 上建立 Naive Bayes 模型，计算出相应的 p(y)和$p_j(x/y)$

（2） 基于计算得到的 p(y)和$p_j(x/y)$，在 test_data 上面对 y 各种取值求解概率大小，保留最大概率的 y 作为对应 x 取值下的预测结果，将其与真实 y 值进行比较，得到预测准确率；

（3） 基于不同的数据大小得到的预测结果如下：



结论分析与体会：

1. Naive Bayes 假设了 x 的各个 feature 是独立同分布的，因此才能够利用该模型进行预测，预测效果不错；

2. 数据集大小会影响预测效果，当数据集十分小时，模型不具有普遍性，因此预测的准确率会降低，当数据集足够大时（此问题中超过 2000 个数据），继续增大数据集大小，预测的准确度不会大幅提升，而是在轻微波动；

附录：程序源代码

```
import numpy as np
import matplotlib.pyplot as plt

class NaiveBayes:
    def __init__(self, training_data, test_data):
        self.training_x = training_data[:, :-1]
        self.training_y = training_data[:, -1]
        self.test_x = test_data[:, :-1]
        self.test_y = test_data[:, -1]
        self.py = np.zeros(5)
        self.pjxy = np.zeros((8, 5, 5))

    def get_py_with_ls(self, count_y, batch_size, number_of_value_y):
        self.py = np.zeros(5)
        for i in range(5):
            self.py[i] = (count_y[i] + 1) / (batch_size + number_of_value_y)

    def get_pjxy_with_ls(self, count_jxy, count_y, number_of_value_x):
        self.pjxy = np.zeros((8, 5, 5))
        for j in range(8):
```

```
            for x in range(5):
                for y in range(5):
                    self.pjxy[j][x][y] += (count_jxy[j][x][y] + 1) / (count_y[y] + number_of_value_x[x])


    def max_likelihood(self, x):
        pred_y = self.py.copy()
        for y in range(5):
            for j in range(8):
                pred_y[y] *= self.pjxy[j][x[j]][y]

        return np.argmax(pred_y)

    def train(self, training_data):
        self.training_x = training_data[:, :-1]
        self.training_y = training_data[:, -1]

        # count_x[j][x]：第 j 个 feature, xj = x 的个数
        count_x = np.zeros((8, 5))
        # count_y[y]: y_label = y 的个数
        count_y = np.zeros(5)
        # count_jxy[j][x][y]：第 j 个 feature，xj = x and y = y 的个数
        count_jxy = np.zeros((8, 5, 5))

        # 遍历所有 training_data，记录每一行中 count_x 和 count_y
        for data_x, data_y in zip(self.training_x, self.training_y):
            count_y[data_y] += 1
            for j in range(8):
                count_x[j][data_x[j]] += 1
                count_jxy[j][data_x[j]][data_y] += 1

        number_of_value_x = np.array([np.sum(count_x[j] > 0) for j in range(8)])
        number_of_value_y = np.sum(count_y > 0)

        self.get_py_with_ls(count_y, training_data.shape[0], number_of_value_y)
        self.get_pjxy_with_ls(count_jxy, count_y, number_of_value_x)

    def predict(self, batch_size):
        right_count = 0
        m = self.test_x.shape[0]
        for data_x, data_y in zip(self.test_x, self.test_y):
            pred_y = self.max_likelihood(data_x)
            if data_y == pred_y:
                right_count += 1
```

```
        print(f'accuracy of batch size = {batch_size}: {right_count / m}')

        return right_count / m

if __name__ == "__main__":
    # load data & initial model
    training_data = np.loadtxt("data4/training_data.txt", dtype=int)
    test_data = np.loadtxt("data4/test_data.txt", dtype=int)

    nb = NaiveBayes(training_data, test_data)
    batch_size = np.arange(100, 1000, 100)
    batch_size = np.concatenate((batch_size, np.arange(1000, 11000, 1000)))

    test_acc_list = []
    for size in batch_size:
        print(size)
        np.random.shuffle(training_data)
        nb.train(training_data[:size, :])
        test_acc_list.append(nb.predict(size))

    plt.figure(1)
    plt.grid()
    plt.ylim([0.5, 1])
    plt.xlabel('Batch size')
    plt.ylabel('Accuracy')
    plt.plot(batch_size, test_acc_list)
    plt.show()
```