机器学习 **Lecture 8**

贝叶斯

一、 贝叶斯决策论

Defination 1 (模型)**.**

$$R(c_i|x) = \sum_{j=1}^{k} \lambda_{ij} P(c_j|x)$$

其中，λ_{ij} 表示将 c_j 类样本误分为 c_i 类的风险。$P(c_j|x)$ 将样本 x 出现在 c_j 类的概率。

$$h^*(x) = \arg\min_{c \in \mathcal{Y}} R(c|x)$$

Defination 2 (贝叶斯定理)**.**

$$P(c|x) = \frac{P(c)P(x|c)}{P(x)}$$

其中，$P(x|c)$ 表示 x 在 c 类中出现的概率

二、 朴素贝叶斯分类器

假设 x 的属性间相互独立，则

$$P(x|c) = \prod_{i=1}^{d} P(x_i|c)$$

$$h(x) = \arg\max P(c) \prod_{i=1}^{d} P(x_i|c)$$

三、 算法

四、 代码实现

在垃圾邮件检测数据集上运行了朴素贝叶斯，准确率为 0.986，f1-score 为 0.99，classification report 如表1所示

```python
import numpy as np


class NaiveBayes:
    def __init__(self):
```

Algorithm 1 朴素贝叶斯

Input: $X \in \mathbb{N}_{+}^{\mathrm{m} \times \mathrm{vocab\_size}}, y \in \mathbb{N}_{+}^{m}$

1: 计算拉普拉斯平滑后每个 label 的个数

$$y\_count_k = A\lambda + \sum_{i=1}^{m} \mathbb{I}(y_i = c_k)$$

其中 A 为类别 c_k 中词集的大小

2: 计算 c_k 下 $x = x_v$ 的概率

$$p_{vk} = \frac{y = c_k \text{下} x_v \text{出现的次数}}{y\_count}$$

表 1: classification report

	precision	**recall**	**f1-score**	**support**
0	1.00	0.99	0.99	4827
1	0.93	0.97	0.95	747
accuracy			0.99	5574
macro avg	0.96	0.98	0.97	5574
weighted avg	0.99	0.99	0.99	5574

```
        pass

    def train(self, X, y):
        self.X = X
        self.y = y
        self.y_count = self.count_y()
        self.num_classes = len(self.y_count)
        self.X_count = np.ones(shape=(self.X.shape[1], self.num_classes))
        self.count_X()

    def predict(self, x):
        idxs = np.where(x > 0)[0]
        p = []
        for i in range(self.num_classes):
            pi = 1
            for idx in idxs:
                pi *= self.X_count[idx, i]
            p.append(pi)
        return np.argmax(p)

    def count_y(self):
        count = [0 for i in range(np.max(self.y) + 1)]
        for i in range(self.X.shape[0]):
            count[self.y[i]] += np.sum(self.X[i, :])
        count = np.array(count)
```

```
31          count = count + self.X.shape[1]
32          return count
33
34      def count_X(self):
35          for i in range(self.X.shape[0]):
36              self.X_count[:, self.y[i]] += self.X[i, :]
37          for i in range(self.X_count.shape[1]):
38              self.X_count[:, i] = self.X_count[:, i] / self.y_count[i]
```