朴素贝叶斯法

首先朴素贝叶斯法属于**生成模型**，其是基于特征条件独立假设学习输入/输出的联合概率分布，然后基于此模型，对给定输入 x 用贝叶斯定理求出后验概率最大的输出 y。

这里的“朴素”，是因为此方法对条件概率分布作了条件独立性的假设，这是一种很强的假设，所以是 朴素（naive）。

使用训练数据学习的是 `P（X|Y）` 和 `P（Y）`的估计，最后得到联合分布概率：`P（X,Y）=P(X|Y)*P(Y)`。

这里假设`P（X|Y）`满足条件独立性：

$$\begin{aligned}P(X=x\mid Y=c_k)&=P(X^{(1)}=x^{(1)},\cdots,X^{(n)}=x^{(n)}\mid Y=c_k)\\&=\prod_{j=1}^{n}P(X^{(j)}=x^{(j)}\mid Y=c_k)\end{aligned}$$

于是可以得到：

$$P(Y=c_k\mid X=x)=\frac{P(X=x\mid Y=c_k)P(Y=c_k)}{\sum_k P(X=x\mid Y=c_k)P(Y=c_k)}$$

最终学习目的是后验概率最大化：

$$y=f(x)=\arg\max_{c_k}\frac{P(Y=c_k)\prod_j P(X^{(j)}=x^{(j)}\mid Y=c_k)}{\sum_k P(Y=c_k)\prod_j P(X^{(j)}=x^{(j)}\mid Y=c_k)}$$

后验概率最大化

0-1损失函数时对期望风险最小化等同于后验概率最大化：

$$\begin{aligned}f(x)&=\arg\min_{y\in\mathcal{Y}}\sum_{k=1}^{K}L(c_k,y)P(c_k\mid X=x)\\&=\arg\min_{y\in\mathcal{Y}}\sum_{k=1}^{K}P(y\neq c_k\mid X=x)\\&=\arg\min_{y\in\mathcal{Y}}(1-P(y=c_k\mid X=x))\\&=\arg\max_{y\in\mathcal{Y}}P(y=c_k\mid X=x)\end{aligned}$$

参数估计

- 极大似然估计

- 贝叶斯估计

  用极大似然估计可能会出现所要估计的概率值为 0 的情况，这会影响到后验概率的计算结果，使分类产生偏差，解决方法就是采用贝叶斯估计。

代码实现

这部分主要实现模型 高斯朴素贝叶斯：

特征的可能性假设为高斯分布，其概率密度函数为：

$$P(x_i|y_k) = \frac{1}{\sqrt{2\pi\sigma_{yk}^2}}exp(-\frac{(x_i-\mu_{yk})^2}{2\sigma_{yk}^2})$$

数学期望(mean)：μ，方差：$\sigma^2 = \frac{\sum(X-\mu)^2}{N}$

```
# 数学期望
@staticmethod
def mean(x):
    return sum(x)/float(len(x))

# 标准差
def std(self,x):
    avg = self.mean(x)
    return math.sqrt(sum(math.pow(x_i-avg,2) for x_i in x)/float(len(x)))

# 概率密度函数
def gaussian_prob(self,x,mean,std):
    exp = math.pow(math.e, -1*(math.pow(x - mean,2))/(2*std))
    return (1/(math.sqrt(2*math.pi*std)))*exp
```

最后基于上面构建的模型进行贝斯斯方法的实现，并对数据做出预测。详情见完整代码naive_bayes.py