第3章

朴素贝叶斯分类器

CHAPTER 3

贝叶斯分类器属于有监督的学习，它是以著名的贝叶斯定理(Bayes theorem)为基础的分类算法，其原理是通过样本的先验概率，利用误判损失来选择最优的类别进行分类。根据样本特征的分布情况，贝叶斯分类器可分为朴素贝叶斯分类器和正态贝叶斯分类器。朴素贝叶斯分类器(Naive Bayes Classifier，NBC)假设样本的特征向量的各个分量相互独立；正态贝叶斯分类器假设样本的特征向量服从正态分布。其中，朴素贝叶斯分类器实现简单，模型预测准确率高，是一种常用的分类算法。


视频讲解

3.1 贝叶斯分类器的理论基础

贝叶斯定理提供了一种计算概率的方法,可预测样本数据属于某一类的概率。要想理解贝叶斯算法的要点,我们需要先了解贝叶斯定理的相关概念。

3.1.1 贝叶斯定理的相关概念

1. 先验概率、条件概率、后验概率与类条件概率

(1) 先验概率(prior probability):在没有训练样本数据前,根据以往的经验和分析得到的概率,初始时假设样本 h 的初始概率用 $P(h)$ 表示。换句话说,先验概率是我们在未知条件下对事件发生的可能性猜测的数学表示。如果没有先验经验,可假定 $P(h)$ 为 50%。例如,如果我们对西瓜的触感、敲声、根蒂和纹路等特征一无所知,一般来说,西瓜是好瓜的概率为 65%。那么这个概率 P(好瓜)就被称为先验概率。

(2) 条件概率(conditional probability):事件 B 发生的条件下,事件 A 发生的概率,表示为 $P(A|B)$,即事件 A 的发生是由于 B 导致的概率。例如,假设上课迟到的原因可能有:①早上没有起来;②感冒发烧,需要去看病。上课迟到是由于感冒发烧引起的概率表示为 P(上课迟到|感冒发烧)。

(3) 后验概率(posterior probability):后验概率也是一种条件概率,它是根据事件结果求事件发生原因的概率。已经观测到事情已经发生了,事件发生的原因有很多,判断结果的发生是由哪个原因引起的概率,也就是说,后验概率是求导致该事件发生的原因是由某个因素引起的可能性的大小,即执果寻因。它限定了条件为观测结果,事件为隐变量取值。例如,上课又迟到了,这是事件的结果,而造成这个结果的原因可能是早上起床晚了,或感冒发烧需要先去看病,P(起床晚了|上课迟到)和 P(感冒发烧|上课迟到)就是后验概率。

(4) 类条件概率(class conditional probability):类条件概率是指把造成事件结果的原因依次列举,分别讨论,即分析并计算某类别情况下,造成此结果的原因。把一个完整的样本集合 S 通过特征进行划分,划分成 M 类 $c_1,c_2,c_3,\cdots,c_M$。假定样本的特征值 x 是一个连续的随机变量,其分布取决于类别状态 c,类条件概率函数 $P(x|c_i)$ 是指在类别 c_i 样品中,特征值 x 的分布情况。x 相对于类标签 c 的概率,也称为似然(likelihood),记作 $P(x|c)$。例如,若西瓜的类别有好瓜和坏瓜两种类别,知道一个西瓜是好瓜的情况下,估计每个属性的概率 P(纹路清晰|好瓜)、P(敲声|好瓜)就是类条件概率。

注意区分以下几个概念:

(1) 先验概率是不依赖观测数据的概率分布,在朴素贝叶斯中,类别的概率就是先验概率,记为 $p(c)$。

(2) 事情已经发生,计算这件事情发生的原因是由某个因素引起的可能性的大小,是后验概率。后验概率的计算要以先验概率为基础。

(3) "似然"描述的是在给定了特定的观测值的条件下,模型的参数的合理性。通常用于建模过程中,选取合适的参数使模型更好地拟合数据。

2. 贝叶斯公式

如果 A 和 B 是样本空间 Ω 的两个事件，在给定 A 条件下 B 的概率为

$$P(B \mid A)=\frac{P(A \cap B)}{P(A)} \tag{3-1}$$

根据式(3-1)，有

$$P(A \cap B)=P(A)P(B \mid A)=P(B)P(A \mid B) \tag{3-2}$$

结合式(3-1)和式(3-2)，有

$$P(B \mid A)=\frac{P(A \mid B)P(B)}{P(A)} \tag{3-3}$$

这就是著名的贝叶斯公式。推广到一般形式，设 A 是样本空间 Ω 上的事件，B 是样本空间 Ω 上的一个划分，则有 $\sum_{i=1}^{n} B_i=\Omega$，且 $B_i \cap B_j=\varnothing$，$(0 \leqslant i \leqslant n, 0 \leqslant j \leqslant n, i \neq j)$，则有 $A=\sum_{i=1}^{n} B_i A$，从而 $P(A)=P(\sum_{i=1}^{n} B_i A)$，因此，有

$$P(A)=\sum_{i=1}^{n} P(A \mid B_i)P(B_i) \tag{3-4}$$

这就是全概率公式。

【例 3-1】 某地区 Y 病毒的感染率为 0.05，在实际检查过程中，可能会由于技术及操作等原因使病毒携带者未必能被检查出阳性反应，同样不携带病毒的被检查者也可能会被检查出阳性。假设 P(阳性|携带病毒)=0.98，P(阳性|不携带病毒)=0.04，假设某人检查出阳性，他携带病毒的概率是多少？

由于 P(阳性|携带病毒)=0.98，P(阳性|不携带病毒)=0.04，则 P(阴性|携带病毒)=0.02，P(阴性|不携带病毒)=0.96。根据贝叶斯公式和全概率公式，有

$$\begin{aligned} P(\text{携带病毒} \mid \text{阳性}) &= \frac{P(A \mid B_j)P(B_j)}{\sum_{i=1}^{n} P(A \mid B_i)P(B_i)} \\ &= \frac{P(\text{阳性} \mid \text{携带病毒})P(\text{携带病毒})}{P(\text{阳性} \mid \text{携带病毒})P(\text{携带病毒})+P(\text{阳性} \mid \text{不携带病毒})P(\text{不携带病毒})} \\ &= \frac{0.98 \times 0.05}{0.98 \times 0.05+0.04 \times 0.95} \approx 0.563 \end{aligned}$$

3.1.2　贝叶斯决策理论

贝叶斯公式的精髓是描述了两个相关的随机事件之间的概率关系，贝叶斯分类器正是在这样的思想指导下，通过计算样本属于某一类的概率值来判定样本的分类。

假设有若干样本 x，类别标签有 M 种，即 $y=\{c_1, c_2, \cdots, c_M\}$，其后验概率为 $P(c_i \mid x)$，若将标签为 c_j 的样本误分为 c_i 类产生的损失为 λ_{ij}，则将样本 x 分类为 c_i 所产生的期望损失为

$$R(c_i \mid x)=\sum_{i=1}^{M}\lambda_{ij}P(c_j \mid x) \tag{3-5}$$

其中,误判损失 λ_{ij} 为

$$\lambda_{ij}=\begin{cases}0 & i=j\\ 1 & \text{其他}\end{cases} \tag{3-6}$$

为了最小化期望损失,需要在每个样本上选择能使期望损失最小的类别标签,即

$$h(\boldsymbol{x})=\underset{c\in y}{\operatorname{argmin}}R(c_i \mid x) \tag{3-7}$$

使分类错误率最小,也就是使分类的正确率最高,因 $R(c_i \mid x)=1-P(c_i \mid x)$,于是,最小化分类错误率的贝叶斯最优分类器变为

$$h(x)=\underset{c\in y}{\operatorname{argmax}}P(c_i \mid x) \tag{3-8}$$

一般情况下,在实际分类任务中难以直接获得后验概率 $P(c_i \mid x)$。为了利用有限的数据集准确估计后验概率 $P(c_i \mid x)$,可通过贝叶斯定理对联合概率分布 $P(x,c_i)$ 建模,再得到 $P(x \mid c_i)$。根据贝叶斯定理,有

$$P(c_i \mid x)=\frac{P(x,c_i)}{P(x)}=\frac{P(c_i)P(x \mid c_i)}{P(x)} \tag{3-9}$$

其中,$P(c_i)$ 就是上面所述的类的先验概率;$P(x \mid c_i)$ 是在类标签 c_i 下 x 的类条件概率;$P(\boldsymbol{x})$ 是样本 $\boldsymbol{x}$ 的概率分布,它对所有的类标签都是相同的。对于分类问题,只需要比较样本属于每一类的概率大小,找出概率最大的那一类即可,故分母 $P(x)$ 是可以省略的。因此,简化后的贝叶斯最优分类器为

$$h(x)=\underset{c\in y}{\operatorname{argmax}}P(c_i)P(x \mid c_i) \tag{3-10}$$

如果已经求出先验概率 $P(c_i)$ 和类条件概率 $P(x \mid c_i)$,则可以根据贝叶斯最优分类器对样本进行分类。其中,类的先验概率 $P(c_i)$ 可根据大数定律对各类样本出现的频率进行估计。但对于类条件概率 $P(x \mid c_i)$,直接根据样本出现的频率进行估计是十分困难的。

3.1.3 极大似然估计

虽然不能直接估计出类条件概率,但还是有获得类条件概率的策略的。为了估计类条件概率,可以先假设其服从某种确定的概率分布,再利用训练样本对概率分布的参数进行估计。这就是极大似然估计(Maximum Likelihood Estimation,MLE)的算法思想,极大似然估计提供了一种给定观察数据来评估模型参数的方法,即模型已定,参数未知。通过若干次实验,观察其结果,利用实验结果得到的某个参数值能够使样本出现的概率为最大,则称为极大似然估计。

假设 T_c 表示训练集 T 中第 c 类样本集合,且这些样本是独立同分布的,则参数 θ_c 对于数据集 T_c 的似然为

$$l(\theta)=P(T_c \mid \theta_c)=P(x_1,x_2,\cdots,x_N \mid \theta)=\prod_{x\in T_c}P(x \mid \theta_c) \tag{3-11}$$

找出参数空间 θ_c 中能使 $l(\theta)$ 取最大参数值的 $\hat{\theta}_c$,其实就是求解

$$\hat{\theta}_c = \underset{\theta}{\operatorname{argmax}} l(\theta) = \underset{\theta}{\operatorname{argmax}} \prod_{x \in T_c} P(x \mid \theta_c) \tag{3-12}$$

式(3-12)容易造成下溢，通常求其对数似然

$$\hat{\theta}_c = \underset{\theta}{\operatorname{argmax}} \ln \prod_{x \in T_c} P(x \mid \theta_c) = \underset{\theta}{\operatorname{argmax}} \sum_{x \in T_c} \ln P(x \mid \theta_c) \tag{3-13}$$

一旦确定了样本服从某种分布，就可以利用梯度下降法得到参数的值。

若已知数据的分布，即似然函数，则很容易利用梯度下降法求出其参数。假设样本服从均值为 μ、方差为 σ^2 的正态分布 $N(\mu, \sigma^2)$，则似然函数为

$$l(\mu, \sigma^2) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \mathrm{e}^{-\frac{(x_i-\mu)^2}{2\sigma^2}} \tag{3-14}$$

对其求对数可得

$$\ln l(\mu, \sigma^2) = -\frac{N}{2}\ln(2\pi) - \frac{N}{2}\ln\sigma^2 - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(x_i - \mu)^2 \tag{3-15}$$

对其分别求 μ 和 σ^2 的偏导可得

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{3-16}$$

$$\hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{\mu})(x_i - \hat{\mu})^{\mathrm{T}} \tag{3-17}$$

也就是说，通过最大似然估计得到的正态分布的均值就是样本的均值，那么方差就是 $(x_i - \hat{\mu})(x_i - \hat{\mu})^{\mathrm{T}}$ 的均值。

对于样本服从其他分布的情况，也是利用类似的方法求解，求最大似然估计量的一般步骤如下：

(1) 写出似然函数 $l(x; \theta)$。

(2) 对似然函数 $l(x; \theta)$取对数，并整理。

(3) 对 $\ln l(x; \theta)$的相应参数 θ 求偏导。

(4) 解似然方程，得到参数 θ 的值。

提　示

极大似然估计，也称最大似然估计，是求估计的常用方法，由德国数学家高斯(C. F. Gauss)提出，它是建立在极大似然原理上的统计方法，是概率论在统计学上的应用，具有算法思想简单、收敛性好的优势，但是实验结果会依赖于事先假设的类条件概率模型。

由于直接估计类条件概率密度函数很困难，参数估计问题只是实际问题求解过程中的一种简化方法，因此，能够使用极大似然估计方法的样本都必须满足一些前提假设：训练样本的分布能代表样本的真实分布。每个样本集中的样本都是独立同分布的随机变量，且有充分的训练样本。

下面简单介绍一下正态分布函数的数学表示及其几何意义。

一个单变量正态分布的密度函数为

$$P(x) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right) \tag{3-18}$$

其正态分布的概率密度函数如图 3-1 所示。

正态分布以 $X=\mu$ 为对称轴左右对称。μ 是正态分布的位置参数,描述正态分布的集中趋势位置。正态分布的期望、均数、中位数、众数相同,均等于 μ。与 μ 越近的值,其概率越大,反之,其概率值越小。σ 描述数据分布的离散程度,σ 越大,数据分布越分散,曲线越扁平;σ 越小,数据分布越集中,曲线越瘦高。分别服从正态分布为 $N(0,1)$、$N(0,1.5)$、$N(1,1)$、$N(1,1.5)$的概率密度函数如图 3-2 所示。

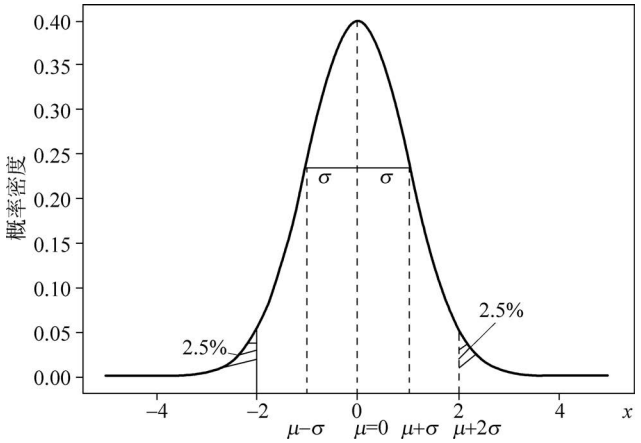


图 3-1 正态分布的概率密度函数

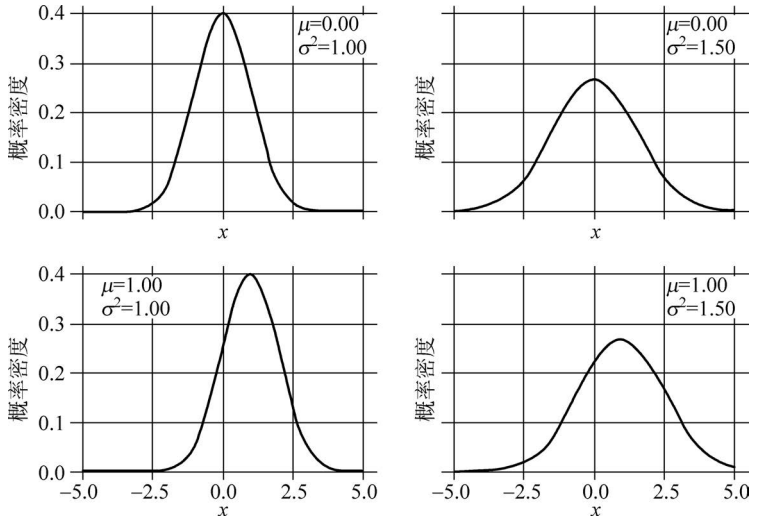


图 3-2 正态分布为 $N(0,1)$、$N(0,1.5)$、$N(1,1)$、$N(1,1.5)$的概率密度函数

对于多变量的正态分布,假设特征向量是服从均值向量为 $\boldsymbol{\mu}$、协方差矩阵为 $\boldsymbol{\Sigma}$ 的 n 维正态分布,其中类条件概率密度函数为

$$P(\boldsymbol{x} \mid c)=\frac{1}{(2\pi)^{\frac{n}{2}}|\boldsymbol{\Sigma}|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right) \tag{3-19}$$

其中,$\boldsymbol{x}=(x_1,x_2,\cdots,x_n)^{\mathrm{T}}$,$\boldsymbol{\mu}=(\mu_1,\mu_2,\cdots,\mu_n)^{\mathrm{T}}$,$\boldsymbol{\Sigma}$ 为 $n\times n$ 协方差矩阵,$|\boldsymbol{\Sigma}|$ 为 $\boldsymbol{\Sigma}$ 的行列式,$\boldsymbol{\Sigma}^{-1}$ 为 $\boldsymbol{\Sigma}$ 的逆矩阵。一个二维正态分布的概率密度函数如图 3-3 所示。

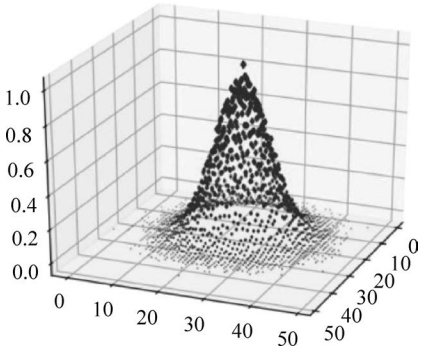


图 3-3　二维正态分布概率密度函数图像

3.2　朴素贝叶斯分类器的原理与设计


视频讲解

为了说明在机器学习中如何使用贝叶斯定理实现对数据的分类，本节根据贝叶斯定理来构造贝叶斯分类器，以此对数据进行分类。为了简化问题，下面假定数据都是独立同分布的。

3.2.1　了解数据集

下面以周志华老师《机器学习》教材中的西瓜数据集为例来讲解如何构造朴素贝叶斯分类器，并基于此数据集根据构造过程进行算法实现，西瓜数据集如表 3-1 所示。

表 3-1　西瓜数据集

序　号	色　泽	根　蒂	敲　声	纹　路	脐　部	触　感	好瓜/坏瓜
1	青绿	蜷缩	浊响	清晰	凹陷	硬滑	好瓜
2	乌黑	蜷缩	沉闷	清晰	凹陷	硬滑	好瓜
3	乌黑	蜷缩	浊响	清晰	凹陷	硬滑	好瓜
4	青绿	蜷缩	沉闷	清晰	凹陷	硬滑	好瓜
5	浅白	蜷缩	浊响	清晰	凹陷	硬滑	好瓜
6	青绿	稍蜷	浊响	清晰	稍凹	软黏	好瓜
7	乌黑	稍蜷	浊响	稍糊	稍凹	软黏	好瓜
8	乌黑	稍蜷	浊响	清晰	稍凹	硬滑	好瓜
9	乌黑	稍蜷	沉闷	稍糊	稍凹	硬滑	坏瓜
10	青绿	硬挺	清脆	清晰	平坦	软黏	坏瓜
11	浅白	硬挺	清脆	模糊	平坦	硬滑	坏瓜
12	浅白	蜷缩	浊响	模糊	平坦	软黏	坏瓜
13	青绿	稍蜷	浊响	稍糊	凹陷	硬滑	坏瓜
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	坏瓜
15	乌黑	稍蜷	浊响	清晰	稍凹	软黏	坏瓜
16	浅白	蜷缩	浊响	模糊	平坦	硬滑	坏瓜
17	青绿	蜷缩	沉闷	稍糊	稍凹	硬滑	坏瓜

表 3-1 所示的西瓜数据集中共 17 条数据，包含色泽、根蒂、敲声、纹路、脐部、触感 6 个特征，最后一列好瓜/坏瓜表示该数据的标签。

3.2.2 手工设计贝叶斯分类器

如何设计贝叶斯分类器？主要依据就是 3.1 节介绍的贝叶斯定理和极大似然估计，在训练得到贝叶斯分类器后，就可根据数据的特征去判断西瓜是好瓜还是坏瓜了。

假设我们要判断第 3 条西瓜数据是否为好瓜，即

3	乌黑	蜷缩	浊响	清晰	凹陷	硬滑	?

根据表 3-1 所示的西瓜数据集，可得好瓜和坏瓜的先验概率：

$$P(\text{好瓜})=\frac{8}{17}\approx 0.471,\quad P(\text{坏瓜})=\frac{9}{17}\approx 0.529$$

然后，为每个属性特征估计条件概率 $P(x_i \mid c)$，这里的西瓜数据集有 6 个特征，为了便于计算，我们只选取色泽、敲声、纹路 3 个特征去判定西瓜的好坏。根据表 3-1 所示的数据集，可统计得到以下类条件概率：

$$P(\text{色泽}=\text{乌黑} \mid \text{好瓜})=\frac{4}{8}=0.5$$

$$P(\text{色泽}=\text{乌黑} \mid \text{坏瓜})=\frac{2}{9}\approx 0.22$$

$$P(\text{敲声}=\text{浊响} \mid \text{好瓜})=\frac{6}{8}=0.75$$

$$P(\text{敲声}=\text{浊响} \mid \text{坏瓜})=\frac{4}{9}\approx 0.44$$

$$P(\text{纹路}=\text{清晰} \mid \text{好瓜})=\frac{7}{8}\approx 0.875$$

$$P(\text{纹路}=\text{清晰} \mid \text{坏瓜})=\frac{2}{9}\approx 0.22$$

假设各特征是相互独立的，则根据式(3-11)有

$$\begin{aligned}h(\text{好瓜})&=P(\text{好瓜})\times P(\text{色泽}=\text{乌黑} \mid \text{好瓜})\times\\&\quad P(\text{敲声}=\text{浊响} \mid \text{好瓜})\times P(\text{纹路}=\text{清晰} \mid \text{好瓜})\\&=0.471\times 0.5\times 0.75\times 0.875\approx 0.1545\\h(\text{坏瓜})&=P(\text{坏瓜})\times P(\text{色泽}=\text{乌黑} \mid \text{坏瓜})\times\\&\quad P(\text{敲声}=\text{浊响} \mid \text{坏瓜})\times P(\text{纹路}=\text{清晰} \mid \text{坏瓜})\\&=0.529\times 0.22\times 0.44\times 0.22\approx 0.0113\end{aligned}$$

显然有 $h(\text{好瓜})>h(\text{坏瓜})$，因此判断为好瓜。

3.2.3 贝叶斯分类器的 Python 实现

根据以上手工设计贝叶斯分类器的过程，利用 Python 语言模拟手工计算的过程就可以实现贝叶斯分类器，从而实现分类。

1. 计算先验概率

首先加载数据集，统计样本中好瓜和坏瓜的数量，计算好瓜和坏瓜的先验概率：

```
dataTrain = [['青绿', '蜷缩', '浊响', '清晰', '凹陷', '硬滑', '好瓜'],
             ['乌黑', '蜷缩', '沉闷', '清晰', '凹陷', '硬滑', '好瓜'],
             ['乌黑', '蜷缩', '浊响', '清晰', '凹陷', '硬滑', '好瓜'],
             ['青绿', '蜷缩', '沉闷', '清晰', '凹陷', '硬滑', '好瓜'],
             ['浅白', '蜷缩', '浊响', '清晰', '凹陷', '硬滑', '好瓜'],
             ['青绿', '稍蜷', '浊响', '清晰', '稍凹', '软黏', '好瓜'],
             ['乌黑', '稍蜷', '浊响', '稍糊', '稍凹', '软黏', '好瓜'],
             ['乌黑', '稍蜷', '浊响', '清晰', '稍凹', '硬滑', '好瓜'],
             ['乌黑', '稍蜷', '沉闷', '稍糊', '稍凹', '硬滑', '坏瓜'],
             ['青绿', '硬挺', '清脆', '清晰', '平坦', '软黏', '坏瓜'],
             ['浅白', '硬挺', '清脆', '模糊', '平坦', '硬滑', '坏瓜'],
             ['浅白', '蜷缩', '浊响', '模糊', '平坦', '软黏', '坏瓜'],
             ['青绿', '稍蜷', '浊响', '稍糊', '凹陷', '硬滑', '坏瓜'],
             ['浅白', '稍蜷', '沉闷', '稍糊', '凹陷', '硬滑', '坏瓜'],
             ['乌黑', '稍蜷', '浊响', '清晰', '稍凹', '软黏', '坏瓜'],
             ['浅白', '蜷缩', '浊响', '模糊', '平坦', '硬滑', '坏瓜'],
             ['青绿', '蜷缩', '沉闷', '稍糊', '稍凹', '硬滑', '坏瓜']]
dataTrain = np.array(dataTrain)
y = dataTrain[:, -1]
good = np.sum(y == '好瓜')                    #好瓜的数量
bad = np.sum(y == '坏瓜')                     #坏瓜的数量
#好瓜和坏瓜的先验概率
prior_good = good/len(y)
prior_bad = bad/len(y)
```

2. 计算类条件概率

为了计算各特征(色泽、敲声、纹路)在好瓜和坏瓜基础上的条件概率,需要分别统计这些特征在好瓜和坏瓜中出现的频率。

```
#统计各特征在好瓜和坏瓜情况下出现的频率
def featureFrequency(feature,flag,X,y):
    c_good = c_bad = 0
    if flag == 'c':                           #颜色
        for index in range(len(X)):
            if X[index,6] == '好瓜' and feature == X[index,0]:
                c_good += 1
            elif X[index,6] == '坏瓜' and feature == X[index,0]:
                c_bad += 1
        return c_good,c_bad
    if flag == 's':                           #声音
        for index in range(len(X)):
            if X[index, 6] == '好瓜' and feature == X[index, 2]:
                c_good += 1
            elif X[index, 6] == '坏瓜' and feature == X[index, 2]:
                c_bad += 1
        return c_good, c_bad
    if flag == 'l':#纹路
        for index in range(len(X)):
            if X[index, 6] == '好瓜' and feature == X[index, 3]:
                c_good += 1
            elif X[index, 6] == '坏瓜' and feature == X[index, 3]:
                c_bad += 1
        return c_good, c_bad
```

根据得到的不同特征出现的频率,计算它们在好瓜和坏瓜情况下的条件概率。

```
#根据频率计算不同特征的条件概率
def feaConProbability(c_good,c_bad,X,y):
    p_good = c_good/good
    p_bad = c_bad/bad
    return p_good,p_bad
```

3. 计算后验概率和分类准确率

由前面计算出的各类样本的先验概率和类条件概率,在各类样本相互独立的情况下,根据朴素贝叶斯定理,可以得到 h(好瓜)和 h(坏瓜)。若 h(好瓜)$>h$(坏瓜),则判定该类样本的标签为"好瓜",否则该类样本的标签为"坏瓜"。最后对预测结果与真实结果进行比较,得到分类准确率。

```
pre = []#存储预测结果
count_good = count_bad = 0
for index in range(len(dataTrain)):
    color = dataTrain[index,0]
    sound = dataTrain[index, 2]
    lines = dataTrain[index, 3]
    #统计在好瓜和坏瓜的情况下不同特征的概率
    c_good,c_bad = featureFrequency(color,'c',dataTrain,y)
    p_c_good,p_c_bad = feaConProbability(c_good,c_bad,dataTrain,y)
    print('颜色概率', p_c_good, p_c_bad)

    s_good, s_bad = featureFrequency(sound, 's', dataTrain, y)
    p_s_good, p_s_bad = feaConProbability(s_good, s_bad, dataTrain, y)
    print('敲声概率', p_s_good, p_s_bad)

    line_good, line_bad = featureFrequency(lines, 'l', dataTrain, y)
    p_l_good, p_l_bad = feaConProbability(line_good, line_bad, dataTrain, y)
    print('纹路概率', p_l_good, p_l_bad)
    if p_c_good*p_s_good*p_l_good*prior_good>
            p_c_bad*p_s_bad*p_l_bad*prior_bad:
        pre.append('好瓜')
    else:
        pre.append('坏瓜')

print('准确率%.2f%%'%(100*np.sum(y==pre)/len(y)))     #输出准确率
```

程序运行结果如下:

```
颜色概率 0.375 0.333
敲声概率 0.75 0.444
纹路概率 0.875 0.222
颜色概率 0.5 0.222
敲声概率 0.25 0.333
纹路概率 0.875 0.222
颜色概率 0.5 0.222
敲声概率 0.75 0.444
纹路概率 0.875 0.222
准确率 88.24%
```

为了验证前面手工计算的第 3 条样本的预测结果，这里输出了前 3 条记录的颜色、敲声、纹路 3 个概率值。

3.2.4　平滑方法

在计算属性特征的条件概率时，可能会出现某属性特征在训练集中没有出现过的情况，按照 3.2.2 节的方法直接计算，会出现概率结果为 0。这会导致在对样本分类时直接认为是 h(好瓜)或 h(坏瓜)。这显然是不合理的，不能因为一个事件没有被观察到就武断地判断该事件的概率是 0。平滑技术就是用来解决在实际数据处理过程中出现零概率的问题，“平滑”处理的基本思想是“劫富济贫”，即提高低概率(零概率)，降低高概率，尽量使概率的分布趋于实际水平。

为了解决零概率的问题，法国数学家拉普拉斯最早提出用加 1 的方法估计没有出现过的现象的概率，因此这种平滑(Smoothing)方法也被称为拉普拉斯平滑(Laplacian Smoothing)。

引入拉普拉斯平滑技术后，修正后的类先验概率和类条件概率可表示为

$$\hat{p}(c)=\frac{|T_c|+1}{|T|+M} \tag{3-20}$$

$$\hat{p}(x_i \mid c)=\frac{|T_{c,x_i}|+1}{|T_c|+M_i} \tag{3-21}$$

其中，T_c 表示类别 c 的训练集，$|T_c|$ 表示集合的个数，M 表示训练集 T 中的类别个数，T_{c,x_i} 表示类别 c 中取值为 x_i 的样本个数，M_i 表示具有第 i 个属性的可能的取值数。

例如，该西瓜数据集中好瓜和坏瓜的类先验概率为

$$\hat{P}(\text{好瓜})=\frac{8+1}{17+2}\approx 0.474$$

$$\hat{P}(\text{坏瓜})=\frac{9+1}{17+2}\approx 0.526$$

修正后，色泽、敲声、纹路 3 个特征的类先验概率为

$$P(\text{色泽}=\text{乌黑} \mid \text{好瓜})=\frac{4+1}{8+3}\approx 0.455$$

$$P(\text{色泽}=\text{乌黑} \mid \text{坏瓜})=\frac{2+1}{9+3}=0.25$$

$$P(\text{敲声}=\text{浊响} \mid \text{好瓜})=\frac{6+1}{8+3}\approx 0.636$$

$$P(\text{敲声}=\text{浊响} \mid \text{坏瓜})=\frac{4+1}{9+3}\approx 0.417$$

$$P(\text{纹路}=\text{清晰} \mid \text{好瓜})=\frac{7+1}{8+3}\approx 0.667$$

$$P(\text{纹路}=\text{清晰} \mid \text{坏瓜})=\frac{2+1}{9+3}=0.25$$

当训练样本很大时，每个分量 x 的计数加 1 造成的估计概率变化可以忽略不计，却可以方便有效地避免零概率问题。

学习要点

朴素贝叶斯分类器的优点：①对小规模数据表现很好，能处理多分类任务。②算法比较简单，常用于文本分类。③有稳定的分类效率，对缺失数据不太敏感。④适合增量式训练，当数据量超出内存时，可一批一批读取数据进行增量训练。缺点是：①使用朴素贝叶斯算法有一个重要的前提条件：样本的特征属性之间是相互独立的，在满足这一条件的数据集上，其分类效果会非常好，而在不满足独立性条件的数据集上，效果欠佳。虽然在理论上，朴素贝叶斯模型与其他分类方法相比，有最小的误差率，但这一结果仅限于满足独立性条件的数据集上。而在实际应用中，属性特征之间不太可能完全独立，在属性特征个数非常多，且属性之间的相关性较大时，朴素贝叶斯的分类效果不佳。②需要知道先验概率，且先验概率很多时候取决于假设，会由于假设的先验模型的原因导致预测效果不佳。


视频讲解

3.3 朴素贝叶斯分类算法的实现——鲈鱼和三文鱼的分类系统

本节将介绍如何利用朴素贝叶斯分类算法的思想，实现鲈鱼和三文鱼的分类。通过贝叶斯分类算法的实现过程，掌握如何得到条件概率、后验概率及可视化的方法，从而学会针对任意给定的样本数据，构造贝叶斯分类器进行分类。

3.3.1 算法实现

利用朴素贝叶斯分类算法对存储在 fish.xls 文件中的鱼类二维特征数据进行分类。这些二维特征包括鲈鱼和三文鱼的长度、亮度特征，鲈鱼和三文鱼各 1000 条数据，前 5 条数据如表 3-2 所示。

表 3-2 鲈鱼和三文鱼二维特征的前 5 条数据

long(数据)	light(属性)	label(类别)
4.5	1	0
3.1	2.6	0
1.1	4.1	0
3.9	2	0
2.6	3.2	0
5.6	9.1	1
5.1	5.3	1
5.9	3.5	1
4.5	6.8	1
3.5	7.6	1

(1) 将随机读取的鲈鱼和三文鱼的长度和亮度特征数据作为训练样本，画出散点图。

(2) 假设鲈鱼和三文鱼的长度、亮度服从正态分布，根据训练样本计算出鲈鱼和三文鱼

的均值和方差，画出估计的类条件概率密度图。

（3）假设先验概率相同，均为 0.5，利用贝叶斯公式，分别计算长度和亮度的分类结果，画出后验概率密度图，计算分类结果的正确率和错误率。

（4）假设长度和亮度是互相完全独立的，利用贝叶斯公式和联合概率密度公式构造鲈鱼和三文鱼的二维分类器，计算分类结果的正确率和错误率。

1. 读取数据

从 fish.xlsx 文件中读取鲈鱼和三文鱼的长度、亮度数据，其中，前 n/2 条数据为鲈鱼，后 n/2 条数据为三文鱼，分别从鲈鱼和三文鱼数据中随机取出 50%作为训练集，其余的 50%作为测试集。

```
#三文鱼数据
readbook = xlrd.open_workbook(r'...\dataset\Fish.xlsx')
sheet = readbook.sheet_by_index(3)              #取出第 3 个工作表中的数据，索引从 0 开始
nrows,ncols = sheet.nrows,sheet.ncols           #行数和列数
fish = []
for i in range(1,nrows):
    v1 = sheet.cell(i,0).value                  #获取第 i 行第 3 列的表格值
    v2 = sheet.cell(i, 1).value                 #获取第 i 行第 3 列的表格值
    fish.append([v1,v2,0,0,0])

n = len(fish)
perch_fish = fish[0:n//2]
salmon_fish = fish[n//2:n]
perch_train = random.sample(list(enumerate(perch_fish)),n//4)
perch_train_index = [i[0] for i in perch_train]
perch_train = [perch_fish[v] for v in perch_train_index]
perch_test_index = [i for i in list(range(n//2)) if i not in perch_train_index]
perch_test = [perch_fish[v] for v in perch_test_index]
#绘制鲈鱼和三文鱼的散点图
salmon_train = np.array(salmon_train)
perch_train = np.array(perch_train)
plt.scatter(perch_train[:,0], perch_train[:,1], marker = 'o', c = 'r',label = '鲈鱼')
plt.scatter(salmon_train[:,0], salmon_train[:,1], marker = 's',c = 'b',label = '三文鱼')
plt.xlabel('长度')
plt.ylabel('亮度')
plt.legend(loc = 'upper left')
plt.rcParams['font.sans-serif'] = ['SimHei'] #显示中文
plt.rcParams['axes.unicode_minus'] = False
plt.show()
```

鲈鱼和三文鱼长度和亮度特征的散点图如图 3-4 所示。

2. 计算鲈鱼和三文鱼的均值和方差

假设鲈鱼和三文鱼的长度、亮度特征数据是服从正态分布的，为了对测试样本进行分类，需要知道正态分布的模型参数，即均值和方差。利用读取出的鲈鱼和三文鱼的长度、亮度特征数据，根据式(3-11)和式(3-12)计算长度和亮度特征的均值和方差。

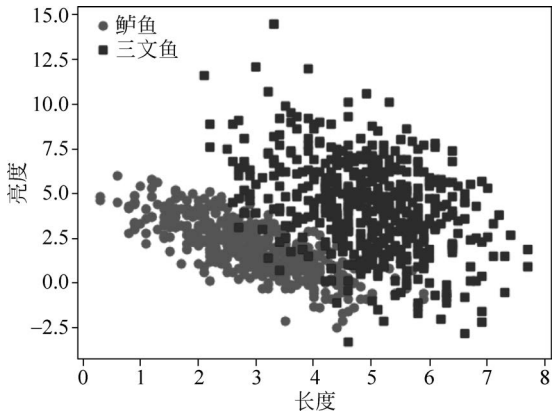


图 3-4　鲈鱼和三文鱼的长度和亮度特征的散点图

```
#估计鲈鱼的长度、亮度特征对应的均值和方差
perch_Mean_Length = np.mean(perch_train[:,0])
perch_Mean_Light = np.mean(perch_train[:,1])
perch_Variance_Length = np.var(perch_train[:,0])
perch_Variance_Light = np.var(perch_train[:,1])
print('鲈鱼长度均值:',perch_Mean_Length)
print('鲈鱼亮度均值:',perch_Mean_Light)
print('鲈鱼长度方差:',perch_Variance_Length)
print('鲈鱼亮度方差:',perch_Variance_Light)
```

与计算鲈鱼的长度、亮度特征的均值和方差类似,估计三文鱼的长度、亮度特征的均值和方差,实现代码如下。

```
#估计三文鱼的长度和亮度特征对应的均值和方差
salmon_Mean_Length = np.mean(salmon_train[:,0])
salmon_Mean_Light = np.mean(salmon_train[:,1])
salmon_Variance_Length = np.var(salmon_train[:,0])
salmon_Variance_Light = np.var(salmon_train[:,1])

print('三文鱼长度均值:',salmon_Mean_Length)
print('三文鱼亮度均值:',salmon_Mean_Light)
print('三文鱼长度方差:',salmon_Variance_Length)
print('三文鱼亮度方差:',salmon_Variance_Light)
```

计算出的鲈鱼和三文鱼的长度、亮度的均值和方差如下。

```
鲈鱼长度均值: 2.984
鲈鱼亮度均值: 2.016
鲈鱼长度方差: 0.961
鲈鱼亮度方差: 1.856
三文鱼长度均值: 4.976
三文鱼亮度均值: 4.241
三文鱼长度方差: 1.092
三文鱼亮度方差: 6.976
```

3. 求解并绘制鲈鱼和三文鱼的长度、亮度特征的概率密度

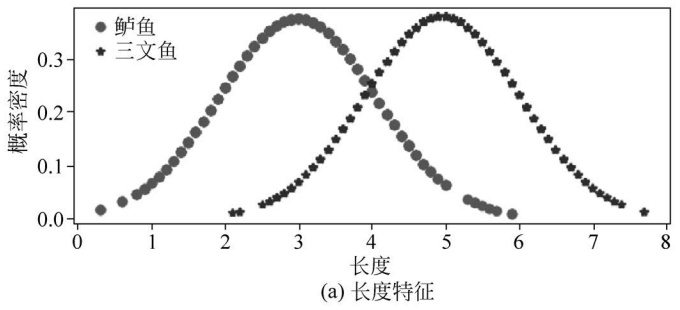
计算出训练集中的鲈鱼和三文鱼的长度、亮度特征均值和方差。

```
#根据估计的鲈鱼和三文鱼长度、亮度特征的均值和方差,绘制类条件概率密度函数
p_c1 = np.zeros((n//4,2))
p_c2 = np.zeros((n//4,2))
for i in range(n//4):
    p_c1[i, 0] = stats.norm(perch_Mean_Length,
                            perch_Variance_Length).pdf(perch_train[i, 0])
    p_c1[i, 1] = stats.norm(perch_Mean_Light,
                            perch_Variance_Light).pdf(perch_train[i, 1])
    p_c2[i,0] = stats.norm(salmon_Mean_Length,
                            salmon_Variance_Length).pdf(salmon_train[i,0])
    p_c2[i,1] = stats.norm (salmon_Mean_Light,
                           salmon_Variance_Light).pdf(salmon_train[i,1])
```

绘制类条件概率密度函数。

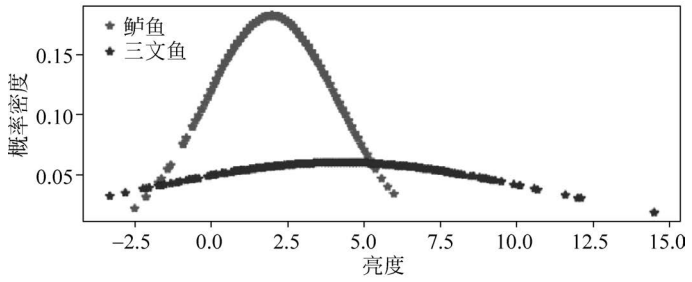
```
plt.subplot(2,1,1)
plt.plot(perch_train[:,0],p_c1[:,0],'ro',label = '鲈鱼')
plt.plot(salmon_train[:,0],p_c2[:,0],'b*',label = '三文鱼')
plt.xlabel('长度')
plt.ylabel('概率密度')
plt.legend(loc = 'upper left')
plt.title('鲈鱼和三文鱼的概率密度')
plt.subplot(2,1,2)
plt.plot(perch_train[:,1],p_c1[:,1],'r*',label = '鲈鱼')
plt.plot(salmon_train[:,1],p_c2[:,1],'b*',label = '三文鱼')
plt.xlabel('亮度')
plt.ylabel('概率密度')
plt.legend(loc = 'upper left')
plt.rcParams['font.sans-serif'] = ['SimHei'] #显示中文
plt.rcParams['axes.unicode_minus'] = False
plt.show()
```

鲈鱼和三文鱼的长度、亮度特征的概率密度如图 3-5 所示。



(a) 长度特征

图 3-5　鲈鱼和三文鱼的概率密度



(b) 亮度特征

图 3-5 (续)

SciPy 是一个基于 NumPy 的开源的 Python 算法库和数学工具包,包含最优化、线性代数、积分、插值、特殊函数、快速傅里叶变换、信号处理和图像处理、常微分方程求解和其他科学与工程中常用的计算。SciPy 中的 stats 用于统计计算的函数库,scipy. stats. norm 函数可以实现正态分布,scipy. stats. norm. pdf 用于求概率密度,它有以下两种调用方式。

(1) gauss1=stats. norm(loc=0,scale=2)　　#loc 为均值,scale 为标准差
　　y=gauss1. pdf(x)　　#x 为特征,y 为条件概率密度

(2) y=norm. pdf(x, loc, scale)

4. 计算鲈鱼和三文鱼的后验概率

根据得到的鲈鱼和三文鱼的训练集的长度、亮度特征的条件概率,利用朴素贝叶斯公式可计算出鲈鱼和三文鱼的测试集中样本的后验概率,即对测试样本进行分类。

```
#计算鲈鱼和三文鱼的后验概率密度
prior = np.zeros((2,1))
prior[0] = 0.5
prior[1] = 0.5
p_post = np.zeros((n // 4, 2))
p_post2 = np.zeros((n // 4, 2))
salmon_test = np.array(salmon_test)
perch_test = np.array(perch_test)
for i in range(n // 4):
    p_post[i, 0] = stats.norm( salmon_Mean_Length,
           salmon_Variance_Length).pdf(salmon_test[i, 0])     #三文鱼的长度概率密度
    p_post[i, 1] = stats.norm(perch_Mean_Length,
         perch_Variance_Length).pdf(salmon_test[i, 0]) #三文鱼划分为鲈鱼的概率密度

    p_post2[i, 0] = stats.norm(perch_Mean_Light,
         perch_Variance_Light).pdf(perch_test[i, 1])   #鲈鱼的亮度概率密度
    p_post2[i, 1] = stats.norm( salmon_Mean_Light,
        salmon_Variance_Light).pdf(perch_test[i, 1])   #鲈鱼的亮度划分为三文鱼的概率密度

for i in range(n// 4):
    p_post[i, 0] = prior[0] * p_post[i, 0] / (p_post[i, 0] * prior[0] +
                      p_post[i, 1] * prior[1])
    p_post[i, 1] = prior[1] * p_post[i, 1] / (p_post[i, 0] * prior[0] +
                      p_post[i, 1] * prior[1])
```

```
    p_post2[i, 0] = prior[1] * p_post2[i, 0] / (p_post2[i, 0] * prior[1] +
                    p_post2[i, 1] * prior[0])
    p_post2[i, 1] = prior[1] * p_post2[i, 1] / (p_post2[i, 0] * prior[1] +
                    p_post2[i, 1] * prior[0])
```

绘制后验概率密度。

```
 plt.subplot(2, 1, 1)
plt.plot(salmon_test[: , 0], p_post[:, 0], 'b*')
plt.plot(salmon_test[: , 0], p_post[:, 1], 'r*')
plt.subplot(2, 1, 2)
plt.plot(perch_test[: , 1], p_post2[:, 0], 'b*')
plt.plot(perch_test[: , 1], p_post2[:, 1], 'r*')
plt.show()
```

后验概率密度如图 3-6 所示。

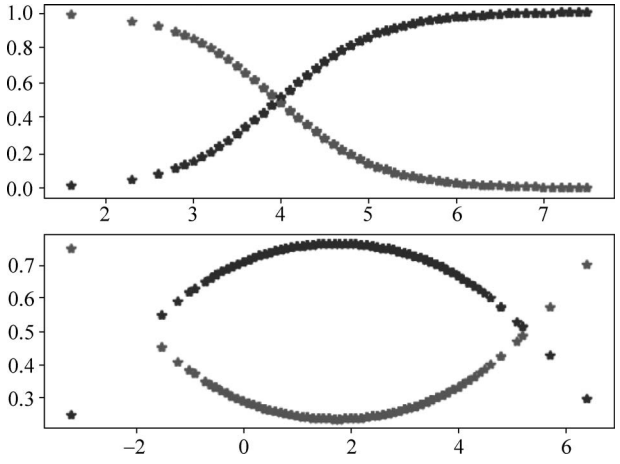


图 3-6　三文鱼和鲈鱼的后验概率密度(见彩插)

5. 计算分类正确率

若长度和亮度是互相完全独立的,根据朴素贝叶斯公式和联合概率密度公式,得出鲈鱼和三文鱼的二维分类器,并计算它们的分类正确率和错误率。

```
#假设长度和亮度是互相完全独立的,根据朴素贝叶斯公式和联合概率密度公式计算出鲈鱼和三文
#鱼的类条件概率,计算分类的正确率和错误率
count1 = 0
count2 = 0
for i in range(n//4):                          #长度特征
    post_length_pred1 = stats.norm(perch_Mean_Length,perch_Variance_Length).pdf(perch_test
[i,0])                                         #将鲈鱼分为鲈鱼
    post_length_pred2 = stats.norm(salmon_Mean_Length,salmon_Variance_Length).pdf(perch_
test[i,0])                                     #将鲈鱼分为三文鱼
    #亮度
    post_light_pred1 = stats.norm(perch_Mean_Light,perch_Variance_Light).pdf(perch_test[i,
1])                                            #将鲈鱼分为鲈鱼
```

```
    post_light_pred2 = stats.norm(salmon_Mean_Length,salmon_Variance_Length).pdf(perch_
test[i,1])                              #将鲈鱼分为三文鱼
    if post_length_pred1 * post_light_pred1 > post_length_pred2 * post_light_pred2:
        count1 += 1
    #长度特征
    post_length_pred1 = stats.norm(salmon_Mean_Length,salmon_Variance_Length).pdf(salmon_
test[i,0])                              #将三文鱼分为三文鱼
    post_length_pred2 = stats.norm(perch_Mean_Length,perch_Variance_Length).pdf(salmon_
test[i,0])                              #将三文鱼分为鲈鱼
    #亮度特征
    post_light_pred1 = stats.norm(salmon_Mean_Light,salmon_Variance_Light).pdf(salmon_test
[i,1])                                  #将三文鱼分为三文鱼
    post_light_pred2 = stats.norm(perch_Mean_Length,perch_Variance_Length).pdf(salmon_test
[i,1])                                  #将三文鱼分为鲈鱼
    if post_length_pred1 * post_light_pred1 > post_length_pred2 * post_light_pred2:
        count2 += 1

precision_salmon = count1/(n//4)
precision_perch = count2/(n//4)
precision_bayes = (count1 + count2)/(n/2)
print('precision_salmon:',precision_salmon)
print('precision_perch:',precision_perch)
print('precision_bayes:',precision_bayes)
```

程序运行结果如下。

```
precision_salmon: 1.0
precision_perch: 0.834
precision_bayes: 0.917
```

若长度和亮度不独立,如何对鲈鱼和三文鱼进行分类呢?可以根据它们的两个特征(长度和亮度)计算出协方差矩阵,根据协方差矩阵计算概率密度,完成朴素贝叶斯分类。

3.3.2 调用系统函数实现

正态朴素贝叶斯分类算法在 scikit-learn 中的实现类如下。

```
sklearn.native_bayes.GaussianNB(priors = None, var_smoothing = 1e - 09)
```

主要参数如表 3-3 所示。

表 3-3 主要参数

参 数	说 明
priors	类别的先验概率。一经指定,不会根据数据进行调整

主要属性如表 3-4 所示。

表 3-4 主要属性

属 性	说 明
class_count_	ndarray of shape(n_classes,) 每个类别中保留的训练样本数量

续表

属　　性	说　　明
class_prior_	ndarray of shape(n_classes,) 每个类别的概率
classes_	ndarray of shape(n_classes,) 分类器已知的类别标签
theta_	ndarray of shape(n_classes, n_features) 每个类中每个特征的均值

主要方法如表 3-5 所示。

表 3-5　主要方法

方　　法	说　　明
fit(X, y[, sample_weight])	根据 X 和 y 拟合高斯朴素贝叶斯分类器
get_params([deep])	获取这个估计器的参数
predict(X)	对测试向量 X 进行分类
predict_log_proba(X)	返回针对测试向量 X 的对数概率估计
predict_proba(X)	返回针对测试向量 X 的概率估计
score(X, y[, sample_weight])	返回给定测试数据和标签上的平均准确率

使用正态朴素贝叶斯算法对鲈鱼和三文鱼进行分类的算法如下。

```
X_train = np.array([[4.5,1],[3.1,2.6],[1.1,4.1],[3.9,2],[2.6,3.2],[2.9,2.9],[4.3,0.3],
[5.6,9.1],[5.1,5.3],[5.9,3.5],[4.5,6.8],[3.5,7.6], [6.7,2.4],[5.1,4.8]])
    y = [0,0,0,0,0,0,0,1,1,1,1,1,1,1]
    model = GaussianNB()                    #构造一个正态朴素贝叶斯分类器
    model.fit(X_train, y)                   #调用训练方法
    #测试数据
    X_test = [[3.9, 4], [2.3, 5.4],[3,3.1],[5.1,9.1],[5,4.4],[3.5,8],[6.6,3.7],
[3.7,6.9],[6,3],[4.5,1.1],[2.8,3],[3,2.3],[6.7,2.4],[5.4,2.6]]
    y_label = [1,1,0,1,1,1,1,1,1,0,0,0,1,1]
    predicted = model.predict(X_test)   #调用预测方法
    print(predicted)
    acc = model.score(X_test,y_label)
    print("预测准确率:",acc)
```

程序运行结果如下：

```
[0 0 0 1 1 1 1 1 1 0 0 0 1 1]
预测准确率: 0.8571428571428571
```

3.4　正态贝叶斯分类器

假设样本的特征向量服从正态分布，则这样的贝叶斯分类器就称为正态贝叶斯分类器或高斯贝叶斯分类器。更一般地，样本的特征并不是相互独立的。

根据分类判决规则，在预测时需要寻找具有最大条件概率值的那个类，即最大化后验概率，等价于求每个类中 $P(\boldsymbol{x}|c)$最大的那个。对 $P(\boldsymbol{x}|c)$取对数，公式为

$$l(\boldsymbol{x};\mu,\boldsymbol{\Sigma})=\ln P(\boldsymbol{x}\mid\boldsymbol{c})=\ln\frac{1}{(2\pi)^{2}\mid\boldsymbol{\Sigma}\mid^{\frac{1}{2}}}-\frac{1}{2}(\boldsymbol{x}-\mu)^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\mu) \tag{3-22}$$

对式(3-22)简化为

$$l(\boldsymbol{x};\mu,\boldsymbol{\Sigma})=\ln P(\boldsymbol{x}\mid\boldsymbol{c})=-\frac{n}{2}\ln(2\pi)-\frac{1}{2}\ln\mid\boldsymbol{\Sigma}\mid-\frac{1}{2}(\boldsymbol{x}-\mu)^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\mu) \tag{3-23}$$

其中，$-\frac{n}{2}\ln(2\pi)$ 与样本所属类别无关，将其从判别函数中消去，不会影响分类结果。此时判别函数可进一步简化为

$$l(\boldsymbol{x};\mu,\boldsymbol{\Sigma})=-\frac{1}{2}\ln\mid\boldsymbol{\Sigma}\mid-\frac{1}{2}(\boldsymbol{x}-\mu)^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\mu) \tag{3-24}$$

其中，μ 和 $\boldsymbol{\Sigma}$ 可根据样本数据得到

$$\mu=-\frac{1}{M}\sum_{i=1}^{M}x_i \tag{3-25}$$

$$\boldsymbol{\Sigma}=\frac{1}{M}\sum_{i=1}^{M}(x_i-\mu)(x_i-\mu)^{\mathrm{T}}=\begin{bmatrix}\sigma_{11}^2 & \sigma_{12}^2 & \cdots & \sigma_{1M}^2\\ \sigma_{21} & \sigma_{22}^2 & \cdots & \sigma_{2N}^2\\ \vdots & \vdots & \ddots & \vdots\\ \sigma_{M1}^2 & \sigma_{M2}^2 & \cdots & \sigma_{MM}^2\end{bmatrix} \tag{3-26}$$

特别地，如果有协方差矩阵为对角阵 $\sigma^2\boldsymbol{I}$，则样本特征相互独立且方差相等，判别函数为

$$l(\boldsymbol{x};\mu,\sigma)=-2n\ln\sigma-\frac{1}{2}\left(\frac{1}{\sigma^2}(\boldsymbol{x}-\mu)^{\mathrm{T}}(\boldsymbol{x}-\mu)\right) \tag{3-27}$$

其中，$\ln\mid\boldsymbol{\Sigma}\mid=\ln\sigma^{2n}=2n\ln\sigma$，$\boldsymbol{\Sigma}^{-1}=\frac{1}{\sigma^2}\boldsymbol{I}$。

3.5 贝叶斯网络

贝叶斯网络(Bayesian network)，又称信念网络(Belief network)，是一种概率图模型(Probabilistic Graphical Model，PGD)，它是一种模拟人类推理过程中因果关系的不确定性处理模型，可通过有向无环图(Directed Acyclic Graph，DAG)来表示。在贝叶斯网络中，用节点表示随机变量，有向边表示变量之间有因果关系，这些随机变量有些是可以观测到的，有些是无法观测到的，两个节点通过箭头连接会产生一个条件概率值，这种概率通常表达的是因果关系。a、b、c 的依赖关系构成一个贝叶斯网络，如图 3-7 所示。

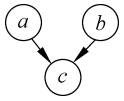


图 3-7 贝叶斯网络

一个贝叶斯网络 BN=(G，Θ)由结构 G 和参数 Θ 构成，G 表示有向无环图，图中每个节点对应一个属性，Θ 描述节点之间的依赖关系。假设属性 x_i 在 G 中的父节点集合为 $\mathrm{pa}(x_i)$，则 Θ 就表示每个属性的条件概率表，即 $P(x_i\mid\mathrm{pa}(x_i))$。若 x_i 表示有向无环图中某一节点 x_i 代表的随机变量，x_i 的概率可表示为

$$P(x_i)=\prod_{i=1}^{k}P(x_i \mid \mathrm{pa}(x_i)) \tag{3-28}$$

以图 3-7 为例，联合概率分布为

$$P(a,b,c)=p(a)p(b)p(c \mid a)p(c \mid b) \tag{3-29}$$

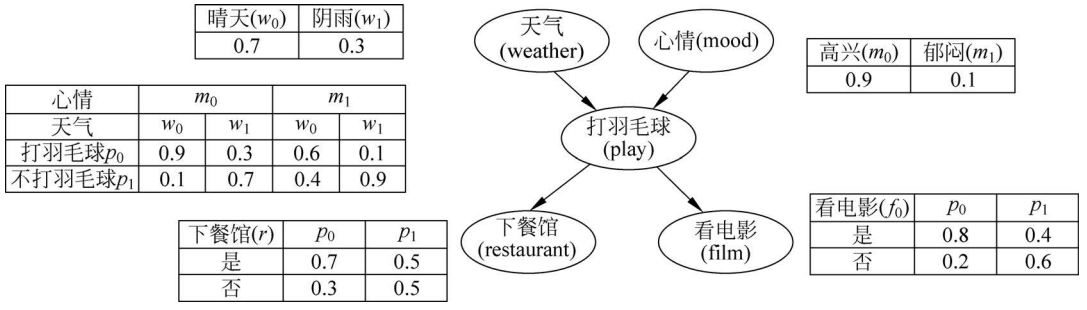
若已知网络结构，也就是属性之间的关系确定，则贝叶斯网络的学习过程就只需要通过对样本进行统计（训练），估计每个节点的条件概率即可。但实际上，网络结构在很多情况下是未知的，那首要任务就是先确定最佳的网络结构，而这是一个 NP 难问题，可通过贪心策略或增加约束条件减少搜索空间。

下面通过一个具体的实例（根据天气和心情好坏决定是否打羽毛球）来说明贝叶斯网络的构造。

假设随机变量 w（weather）表示天气；随机变量 m（mood）表示心情；随机变量 p（play）表示打羽毛球；随机变量 r（restaurant）表示下餐馆吃饭；随机变量 f（film）表示看电影。

变量 w 和变量 m 对变量 p 有因果影响，而变量 p 对变量 r 和变量 f 也有因果影响。

变量之间的关系可描绘成贝叶斯网络，各节点之间的概率表如图 3-8 所示。根据天气和心情的好坏决定是否打羽毛球，打完羽毛球可能会去餐馆吃饭或者看电影。



晴天(w_0)	阴雨(w_1)
0.7	0.3

心情	m_0		m_1	
天气	w_0	w_1	w_0	w_1
打羽毛球p_0	0.9	0.3	0.6	0.1
不打羽毛球p_1	0.1	0.7	0.4	0.9

高兴(m_0)	郁闷(m_1)
0.9	0.1

下餐馆(r)	p_0	p_1
是	0.7	0.5
否	0.3	0.5

看电影(f_0)	p_0	p_1
是	0.8	0.4
否	0.2	0.6

图 3-8　节点间的概率关系

基于 Python 的 pgmpy 库构建贝叶斯网络，其步骤是先建立网络结构，然后填入相关参数。

（1）针对已知结构及参数，先采用 BayesianModel 构造贝叶斯网结构。

```
from pgmpy.models import BayesianModel
from pgmpy.factors.discrete import TabularCPD
from pgmpy.inference import VariableElimination
import pandas as pd
#建立模型

model = BayesianModel([
                       ('w', 'p'),
                       ('m', 'p'),
                       ('p', 'r'),
                       ('p', 'f')])
```

（2）通过 TabularCPD 构造条件概率分布（Condition Probability Distribution，CPD）表格，最后将 CPD 数据添加到贝叶斯网络结构中，完成贝叶斯网络的构造。

```
#定义条件概率分布
   #variable:变量
   #variable_card:基数
   #values:变量值
   #evidence:依据
   #evidence_card:依据基数
cpd_A = TabularCPD(variable='w',
                   variable_card=2,
                   values=[[0.7],[0.3]])
cpd_B = TabularCPD(variable='m',
                   variable_card=2,
                   values=[[0.9],[0.1]])
cpd_C = TabularCPD(variable='p',
                variable_card=2,
                values=[[0.9,0.3,0.6,0.1],
                        [0.1,0.7,0.4,0.9]],
                evidence_card=[2,2],
                evidence=['m','w'])
cpd_D = TabularCPD(variable='r',
                  variable_card=2,
                  values=[[0.7,0.5], [0.3,0.5]],
                 evidence=['p'],evidence_card=[2])
cpd_W = TabularCPD(variable='f',
                 variable_card=2,
                 values=[[0.8,0.4],
                         [0.2,0.6]],
                 evidence=['p'],
                 evidence_card=[2])

#将有向无环图与条件概率分布表关联
model.add_cpds(cpd_A,cpd_B,cpd_C,cpd_D,cpd_W)
```

(3) 验证模型并进行推断。检查网络结构和 CPD,并验证 CPD 是否正确定义和总和为 1。

```
#验证模型:检查网络结构和 CPD,并验证 CPD 是否正确定义和总和为 1
model.check_model()
#获取节点"w(天气情况)"的概率表:
print(model.get_cpds("w"))

#获取整个贝叶斯网络的局部依赖:
print(model.local_independencies(['p','r','f']))

#推测"f(是否看电影)"的节点概率,在 pgmpy 中我们只需要省略额外参数即可计算出条件分布概率
infer = VariableElimination(model)
print(infer.query(['f'], evidence={'p':1,'p':0}))

#变量消除法是精确推断的一种方法
 asia_infer = VariableElimination(model)
 q = asia_infer.query(variables=['r'], evidence={'p': 0})
 print(q)
 q = asia_infer.query(variables=['r'], evidence={'m': 0})
 print(q)
```

程序运行结果如下所示。

```
+------+-----+
| w(0) | 0.7 |
+------+-----+
| w(1) | 0.3 |
+------+-----+
(r ⊥ w, m, f | p)
(f ⊥ w, m, r | p)
+------+----------+
| f | phi(f) |
+====== +========== +
| f(0) | 0.8000 |
+------+----------+
| f(1) | 0.2000 |
+------+----------+
+------+----------+
| r | phi(r) |
+====== +========== +
| r(0) | 0.7000 |
+------+----------+
| r(1) | 0.3000 |
+------+----------+
+------+----------+
| r | phi(r) |
+====== +========== +
| r(0) | 0.6440 |
+------+----------+
| r(1) | 0.3560 |
+------+----------+
```

思政元素

鸢尾花在古代据说是治疗脾脏病痛的良药，也是骨折的外敷药。大多数人对“鸢尾花”这个名字会感到陌生，然而一说“爱丽丝”就很熟悉了，其实它们是同一种花卉的不同叫法而已。爱丽丝(Iris)是希腊语音译过来的名字，是“彩虹”的意思，意味着它的色彩绚烂，像天上的彩虹一样美丽。鸢尾花在中国常用以象征爱情和友谊，有鹏程万里、前途无量、明察秋毫之意。欧洲人认为它象征光明和自由。在古代埃及，鸢尾花是力量与雄辩的象征。此外，鸢尾花还是属羊的人的生命之花，代表着使人生更美好。

3.6　本章小结

贝叶斯分类是以贝叶斯定理为基础的分类方法。朴素贝叶斯分类是贝叶斯分类中最简单、最常见的一种分类方法，它假设样本特征之间是相互独立的。理论上，朴素贝叶斯分类与其他分类方法相比具有最小的误差率，但在实际应用中样本特征个数比较多或者特征之间相关性较大时，分类效果表现较差。为了避免零概率情况，在构造朴素贝叶斯分类器时，还需要利用数据平滑的方法进行处理。

3.7 习题

在线测试

一、选择题

1. 朴素贝叶斯分类器属于(　　)。

A. 判别模型　　B. 生成模型　　C. 预算模型　　D. 统计模型

2. (　　)算法在数据量较少的情况下,仍然能较准确地对数据进行分类。

A. k 近邻　　B. 支持向量机　　C. 朴素贝叶斯　　D. 人工神经网络

3. 以下关于朴素贝叶斯分类算法的说法,正确的是(　　)。

A. 朴素贝叶斯需要大量数据训练才能较准确地对测试样本进行分类

B. 朴素贝叶斯分类算法只能处理非连续型数据

C. 朴素贝叶斯假设样本特征之间相互独立

D. 朴素贝叶斯在分类时需要计算各种类别的概率

4. (　　)不属于朴素贝叶斯分类算法的优点。

A. 对小规模的数据表现很好,能处理多分类任务,适合增量式训练

B. 对缺失数据不太敏感,算法也比较简单,常用于文本分类

C. 具有可解释性

D. 适合处理样本属性有关联的数据

5. 根据西瓜数据集的第一条记录特征,预测其是好瓜还是坏瓜时,先验概率 P(好瓜)$=\frac{8}{17}\approx 0.471$,$P$(坏瓜)$=\frac{9}{17}\approx 0.529$,请计算类条件概率 P(色泽=青绿 | 好瓜)、P(纹路=清晰 | 坏瓜)。下列计算结果正确的是(　　)。

1	青绿	蜷缩	浊响	清晰	凹陷	硬滑	?

A. 0.375,0.625　　B. 0.375,0.222

C. 0.222,0.750　　D. 0.333,0.875

二、算法分析题

1. 编写算法,编写一个朴素贝叶斯分类器,并对鸢尾花数据进行分类。

2. 对于表 3-6 中的样本,假设这些样本特征都是相互独立的,请训练一个朴素贝叶斯分类器,并预测样本 $\boldsymbol{x}=(2,S)^{\mathrm{T}}$ 属于哪一类?其中,X_1、X_2 为样本数据特征,y 为标签。

表 3-6　训练样本

指标	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
X_1	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3
X_2	S	S	M	S	M	S	S	M	L	L	M	L	M	L	L
y	−1	−1	1	1	−1	1	−1	−1	1	1	1	1	1	−1	1