

法律声明

■ 本课件包括演示文稿、示例、代码、题库、视频和声音等内容，北风网和讲师拥有完全知识产权；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或者机构不得盗版、复制、仿造其中的创意和内容，我们保留一切通过法律手段追究违反者的权利。

■ 课程详情请咨询

◆ 微信公众号：北风教育

◆ 官方网址：<http://www.ibeifeng.com/>



人工智能之机器学习

朴素贝叶斯 (Naive Bayes)

主讲人：赵翌臣

上海育创网络科技有限公司



数学回顾：先验概率、后验概率（条件概率）引例



- 想象有 A、B、C 三个不透明的碗倒扣在桌面上，已知其中一个瓷碗下面有鸡蛋。此时请问，鸡蛋在 A 碗下面的概率是多少？答曰 $1/3$ 。
- 现在发生一件事：有人揭开了 C 碗，发现 C 碗下面没有蛋。此时再问：鸡蛋在 A 碗下面的概率是多少？答曰 $1/2$ 。注意，由于有“揭开C碗发现鸡蛋不在C碗下面”这个新情况，对于“鸡蛋在 A 碗下面”这件事的主观概率由原来的 $1/3$ 上升到了 $1/2$ 。这里的先验概率就是 $1/3$ ，后验概率（条件概率）是 $1/2$ 。

数学回顾：乘法公式、全概率公式、贝叶斯公式

■ 条件概率

- ◆ 设A, B为任意两个事件, 若 $P(A) > 0$, 我们称在已知事件A发生的条件下, 事件B发生的概率为条件概率, 记为 $P(B|A)$, 并定义

$$P(B | A) = \frac{P(AB)}{P(A)}$$

■ 乘法公式

- ◆ 如果 $P(A) > 0$, 则 $P(AB) = P(A)P(B|A)$
- ◆ 如果 $P(A_1 \dots A_{n-1}) > 0$, 则 $P(A_1 \dots A_n) = P(A_1) P(A_2|A_1) P(A_3|A_1 A_2) \dots P(A_n|A_1 \dots A_{n-1})$

数学回顾：乘法公式、全概率公式、贝叶斯公式

■ 全概率公式

◆ 如果 $\bigcup_{i=1}^n A_i = \Omega$, $A_i A_j = \phi$ (对一切 $i \neq j$), $P(A_i) > 0$, 则对任一事件B, 有

$$P(B) = \sum_{i=1}^n P(A_i)P(B | A_i)$$

- ◆ 全概率公式是用于计算某个“结果”B发生的可能性大小。如果一个结果B的发生总是与某些前提条件 A_i 相联系, 那么在计算 $P(B)$ 时, 我们就要用 A_i 对B作分解, 应用全概率公式计算 $P(B)$, 我们常称这种方法为**全集分解法**。
- ◆ 根据小偷们的资料, 计算村子今晚失窃概率的问题 (今晚有且仅有一个小偷作案):
 $P(A_i)$ 表示小偷*i*作案的概率, $P(B|A_i)$ 表示小偷*i*作案成功的概率, 那么 $P(B)$ 就是村子失窃的概率

数学回顾：乘法公式、全概率公式、贝叶斯公式

■ 贝叶斯公式（又称逆概公式）

◆ 如果 $\bigcup_{i=1}^n A_i = \Omega$, $A_i A_j = \phi$ (对一切 $i \neq j$), $P(A_i) > 0$, 则对任一事件 B , 只要 $P(B) > 0$, 有

$$P(A_j | B) = \frac{P(A_j B)}{P(B)} = \frac{P(A_j)P(B | A_j)}{\sum_{i=1}^n P(A_i)P(B | A_i)} \quad (i, j = 1, 2, \dots, n)$$

◆ 如果在 B 发生的条件下探求导致这一结果的各种“原因” A_i 发生的可能性大小 $P(A_i | B)$, 则要应用贝叶斯公式

◆ 若村子今晚失窃, 计算哪个小偷嫌疑最大的问题（嫌疑最大就是后验概率最大）

数学回顾：乘法公式、全概率公式、贝叶斯公式

- 假设小偷1和小偷2在某村庄的作案数量比为3:2，前者偷窃成功的概率为0.02，后者为0.01，现村庄失窃，求这次失窃是小偷1作案的概率。
- 【分析】 $A_1 = \{\text{小偷1作案}\}$, $A_2 = \{\text{小偷2作案}\}$, $B = \{\text{村庄失窃}\}$

$$P(A_1) = 3/5, \quad P(A_2) = 2/5$$

$$P(B | A_1) = 0.02, \quad P(B | A_2) = 0.01$$

$$P(A_1 | B) = \frac{P(A_1)P(B | A_1)}{\sum_{i=1}^2 P(A_i)P(B | A_i)} = \frac{3/5 \times 0.02}{3/5 \times 0.02 + 2/5 \times 0.01} = 3/4$$

朴素贝叶斯直观理解

- 肤色 $x_1 = \{\text{黑}, \text{黄}\}$, 发型 $x_2 = \{\text{卷}, \text{直}\}$; 地区 $\text{label} = \{\text{亚}, \text{非}\}$
- 比如告诉你一个人, 其肤色=黑, 发型=卷, 那么你会预测这个人的地区为亚洲还是非洲?

朴素贝叶斯直观理解

■ 模型构建：根据资料计算模型参数

- ◆ 亚洲人的比例
- ◆ 非洲人的比例
- ◆ 亚洲人中肤色=黑的比例
- ◆ 亚洲人中肤色=黄的比例
- ◆ 非洲人中肤色=黑的比例
- ◆ 非洲人中肤色=黄的比例
- ◆ 亚洲人中发型=卷的比例
- ◆ 亚洲人中发型=直的比例
- ◆ 非洲人中发型=卷的比例
- ◆ 非洲人中发型=直的比例



朴素贝叶斯直观理解

■ 例子

◆ 有一个训练集包含100个人，其中有60个非洲人（黑卷*47, 黑直*1, 黄卷*11, 黄直*1），有40个亚洲人（黑卷*1, 黄卷*4, 黄直*35），请训练朴素贝叶斯模型

■ 先计算先验概率：

$$P(\text{非洲}) = \frac{60}{100}, P(\text{亚洲}) = \frac{40}{100}$$

■ 再计算每一个特征的条件概率：

$$P(\text{黑} | \text{非洲}) = \frac{48}{60}, P(\text{黄} | \text{非洲}) = \frac{12}{60}, P(\text{直} | \text{非洲}) = \frac{2}{60}, P(\text{卷} | \text{非洲}) = \frac{58}{60}$$

$$P(\text{黑} | \text{亚洲}) = \frac{1}{40}, P(\text{黄} | \text{亚洲}) = \frac{39}{40}, P(\text{直} | \text{亚洲}) = \frac{35}{40}, P(\text{卷} | \text{亚洲}) = \frac{5}{40}$$

朴素贝叶斯直观理解

- 假设新来了一个人【[黑, 卷], 地区=? 】, 请用朴素贝叶斯模型预测这个人的地区。

Y表示地区, X表示特征向量, 根据贝叶斯公式, 并假设特征间独立的假设有:

$$P(Y | X) = \frac{P(Y) \times P(X | Y)}{P(X)} \xrightarrow{\text{条件独立性假设}} P(Y | X) = \frac{P(Y) \times P(X^{(1)} | Y) \times P(X^{(2)} | Y)}{P(X)}$$

- 和特征间独立的假设 (朴素), 得

$$P(\text{非洲} | \text{黑卷}) = P(\text{非洲})P(\text{黑} | \text{非洲})P(\text{卷} | \text{非洲}) = \frac{60}{100} \cdot \frac{48}{60} \cdot \frac{58}{60}$$

$$P(\text{亚洲} | \text{黑卷}) = P(\text{亚洲})P(\text{黑} | \text{亚洲})P(\text{卷} | \text{亚洲}) = \frac{40}{100} \cdot \frac{1}{40} \cdot \frac{5}{40}$$

$$P(X = x | Y = c_k) = P(X^{(1)} = x^{(1)}, \dots, X^{(n)} = x^{(n)} | Y = c_k) \\ = \prod_{j=1}^n P(X^{(j)} = x^{(j)} | Y = c_k)$$

- 根据计算结果, 模型会将这个人的地区预测为非洲。

朴素的意义

- 朴素：假设特征间是独立的（忽略肤色和发型的联系）。从而变成了“低配版的贝叶斯模型”，称为“朴素贝叶斯”。优点是可以减少需要估计的参数的个数；缺点是会牺牲一定的分类准确率。

- 如果是贝叶斯模型的话，模型参数总数为：

$$\text{len}\{\text{亚洲, 非洲}\} \cdot \text{len}\{\text{黑, 黄}\} \cdot \text{len}\{\text{富, 穷}\} = 2 * 2 * 2$$

是指数增长的，实际是不可行的；而朴素贝叶斯模型参数总数为：

$$\text{len}\{\text{亚洲, 非洲}\} \cdot (\text{len}\{\text{黑, 黄}\} + \text{len}\{\text{富, 穷}\}) = 2 * (2 + 2)$$

是线性增长的，可行

朴素贝叶斯的工作原理

- 训练：先根据数据集，计算标记（地区）的先验概率，再计算每一个特征（肤色和发型）的条件概率，这些概率值就是模型参数，因此朴素贝叶斯的训练成本很低。
- 预测：当一个【黑，卷】来报道时，假设特征间是独立的，朴素贝叶斯模型会预测他的老家是非洲的，原理就是“非洲人的概率 * 非洲人里肤色为黑的比例 * 非洲人里发型为卷的比例 > 亚洲人的概率 * 亚洲人里肤色为黑的比例 * 亚洲人里发型为卷的比例”。朴素贝叶斯模型会将实例预测为后验概率最大的类。

朴素贝叶斯有一个问题

- 继续上文的引例，考虑一个这样的问题：
- 假设某人的地区完全依靠其肤色的就能确定，发型是一个对判断地区没有参考价值的特征，假设 $P(\text{卷}|\text{非洲})=0$ ， $P(\text{卷}|\text{亚洲})=0.001$ ，当来了一个【黑，卷】人的时候，我们算出

$$P(\text{非洲})P(\text{黑}|\text{非洲})P(\text{卷}|\text{非洲}) = 0$$

$$P(\text{亚洲})P(\text{黑}|\text{亚洲})P(\text{卷}|\text{亚洲}) = 0.00001$$

- 然后被预测为亚洲人，傻了吧？
- 原因：出现某个模型参数为0时，0乘任何数都=0，直接影响到后验概率的计算结果。

拉普拉斯平滑

- 解决这一问题的方法是使用平滑操作，改造先验概率公式：

$$P(\text{非洲}) = \frac{60 + \lambda}{100 + \text{len}\{\text{亚洲}, \text{非洲}\} \cdot \lambda} = \frac{60 + \lambda}{100 + 2 \cdot \lambda}$$

- 改造每个特征的条件概率公式（这里只列举了2个）：

$$P(\text{黑} | \text{非洲}) = \frac{48 + \lambda}{60 + \text{len}\{\text{黑}, \text{白}\} \cdot \lambda} = \frac{48 + \lambda}{60 + 2 \cdot \lambda}$$

$$P(\text{直} | \text{非洲}) = \frac{2 + \lambda}{60 + \text{len}\{\text{直}, \text{卷}\} \cdot \lambda} = \frac{2 + \lambda}{60 + 2 \cdot \lambda}$$

- 在随机变量各个取值的频数上赋予一个正数，当时 $\lambda = 1$ ，称为拉普拉斯平滑

	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
$X^{(2)}$	<i>S</i>	<i>M</i>	<i>M</i>	<i>S</i>	<i>S</i>	<i>S</i>	<i>M</i>	<i>M</i>	<i>L</i>	<i>L</i>	<i>L</i>	<i>M</i>	<i>M</i>	<i>L</i>	<i>L</i>
Y	-1	-1	1	1	-1	-1	-1	1	1	1	1	1	1	1	-1

$$P(Y=1)=\frac{9}{15}, \quad P(Y=-1)=\frac{6}{15}$$

$$P(X^{(1)}=1|Y=1)=\frac{2}{9}, \quad P(X^{(1)}=2|Y=1)=\frac{3}{9}, \quad P(X^{(1)}=3|Y=1)=\frac{4}{9}$$

$$P(X^{(2)}=S|Y=1)=\frac{1}{9}, \quad P(X^{(2)}=M|Y=1)=\frac{4}{9}, \quad P(X^{(2)}=L|Y=1)=\frac{4}{9}$$

$$P(X^{(1)}=1|Y=-1)=\frac{3}{6}, \quad P(X^{(1)}=2|Y=-1)=\frac{2}{6}, \quad P(X^{(1)}=3|Y=-1)=\frac{1}{6}$$

$$P(X^{(2)}=S|Y=-1)=\frac{3}{6}, \quad P(X^{(2)}=M|Y=-1)=\frac{2}{6}, \quad P(X^{(2)}=L|Y=-1)=\frac{1}{6}$$

对于给定的 $x=(2,S)^T$ 计算:

$$P(Y=1)P(X^{(1)}=2|Y=1)P(X^{(2)}=S|Y=1)=\frac{9}{15} \cdot \frac{3}{9} \cdot \frac{1}{9}=\frac{1}{45}$$

$$P(Y=-1)P(X^{(1)}=2|Y=-1)P(X^{(2)}=S|Y=-1)=\frac{6}{15} \cdot \frac{2}{6} \cdot \frac{3}{6}=\frac{1}{15}$$

因为 $P(Y=-1)P(X^{(1)}=2|Y=-1)P(X^{(2)}=S|Y=-1)$ 最大, 所以 $y=-1$.

	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
$X^{(2)}$	<i>S</i>	<i>M</i>	<i>M</i>	<i>S</i>	<i>S</i>	<i>S</i>	<i>M</i>	<i>M</i>	<i>L</i>	<i>L</i>	<i>L</i>	<i>M</i>	<i>M</i>	<i>L</i>	<i>L</i>
Y	-1	-1	1	1	-1	-1	-1	1	1	1	1	1	1	1	-1

$$P(Y=1)=\frac{10}{17}, \quad P(Y=-1)=\frac{7}{17}$$

$$P(X^{(1)}=1|Y=1)=\frac{3}{12}, \quad P(X^{(1)}=2|Y=1)=\frac{4}{12}, \quad P(X^{(1)}=3|Y=1)=\frac{5}{12}$$

$$P(X^{(2)}=S|Y=1)=\frac{2}{12}, \quad P(X^{(2)}=M|Y=1)=\frac{5}{12}, \quad P(X^{(2)}=L|Y=1)=\frac{5}{12}$$

$$P(X^{(1)}=1|Y=-1)=\frac{4}{9}, \quad P(X^{(1)}=2|Y=-1)=\frac{3}{9}, \quad P(X^{(1)}=3|Y=-1)=\frac{2}{9}$$

$$P(X^{(2)}=S|Y=-1)=\frac{4}{9}, \quad P(X^{(2)}=M|Y=-1)=\frac{3}{9}, \quad P(X^{(2)}=L|Y=-1)=\frac{2}{9}$$

对于给定的 $x=(2,S)^T$ 计算:

$$P(Y=1)P(X^{(1)}=2|Y=1)P(X^{(2)}=S|Y=1)=\frac{10}{17} \cdot \frac{4}{12} \cdot \frac{2}{12}=\frac{5}{153}=0.0327$$

$$P(Y=-1)P(X^{(1)}=2|Y=-1)P(X^{(2)}=S|Y=-1)=\frac{7}{17} \cdot \frac{3}{9} \cdot \frac{4}{9}=\frac{28}{459}=0.0610$$

由于 $P(Y=-1)P(X^{(1)}=2|Y=-1)P(X^{(2)}=S|Y=-1)$ 最大, 所以 $y=-1$.

编程——朴素贝叶斯



	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
$X^{(2)}$	<i>S</i>	<i>M</i>	<i>M</i>	<i>S</i>	<i>S</i>	<i>S</i>	<i>M</i>	<i>M</i>	<i>L</i>	<i>L</i>	<i>L</i>	<i>M</i>	<i>M</i>	<i>L</i>	<i>L</i>
Y	-1	-1	1	1	-1	-1	-1	1	1	1	1	1	1	1	-1

对于给定的 $x = (2, S)^T$ 计算：

$$P(Y=1)P(X^{(1)}=2|Y=1)P(X^{(2)}=S|Y=1)=\frac{10}{17} \cdot \frac{4}{12} \cdot \frac{2}{12} = \frac{5}{153} = 0.0327$$

$$P(Y=-1)P(X^{(1)}=2|Y=-1)P(X^{(2)}=S|Y=-1)=\frac{7}{17} \cdot \frac{3}{9} \cdot \frac{4}{9} = \frac{28}{459} = 0.061$$

由于 $P(Y=-1)P(X^{(1)}=2|Y=-1)P(X^{(2)}=S|Y=-1)$ 最大，所以 $y=-1$ 。

朴素贝叶斯

- 在机器学习中，朴素贝叶斯分类器是一系列以假设特征之间强独立（朴素）下运用贝叶斯定理为基础的简单概率分类器。
- 高度可扩展的，求解过程只需花费线性时间
- **目前来说**，朴素贝叶斯在文本分类（text classification）的领域的应用多，无论是sklearn还是Spark Mllib中，都只定制化地实现了在文本分类领域的算法



例 13-1 对于表 13-1 中的例子，文档分类所需要的多项式参数包括先验概率 $\hat{P}(c)$
 $= 3/4$ ， $\hat{P}(\bar{c}) = 1/4$ 。其他参数为：

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

$$\hat{P}(\text{Chinese}|c) = (5+1)/(8+6) = 6/14 = 3/7$$

$$\hat{P}(\text{Tokyo}|c) = \hat{P}(\text{Japan}|c) = (0+1)/(8+6) = 1/14$$

$$\hat{P}(\text{Chinese}|\bar{c}) = (1+1)/(3+6) = 2/9$$

$$\hat{P}(\text{Tokyo}|\bar{c}) = \hat{P}(\text{Japan}|\bar{c}) = (1+1)/(3+6) = 2/9$$



表13-1 用于参数估计的数据

	文档ID	文档中的词	属于 $c=\text{China}$ 类?
训练集	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
测试集	5	Chinese Chinese Chinese Tokyo Japan	?

上述计算中的分母分别是 $(8+6)$ 和 $(3+6)$ ，这是因为 c 类文档的总长度为 8，而非 c 类文档的总长度为 3，而按照公式 (13-7)，常数 B 为词汇表大小 6。因此，我们有

$$\hat{P}(c|d_5) \propto 3/4 \cdot (3/7)^3 \cdot 1/14 \cdot 1/14 \approx 0.0003,$$

$$\hat{P}(\bar{c}|d_5) \propto 1/4 \cdot (2/9)^3 \cdot 2/9 \cdot 2/9 \approx 0.0001.$$

于是，分类器会将测试文档归于 $c=\text{China}$ 类。这是因为 d_5 中的正特征 Chinese 的权重比负特征 Japan 和 Tokyo 的权重都要大。

特征工程——TF

■ Tf means **term-frequency**

```
from sklearn.feature_extraction.text import CountVectorizer
X = ['我 爱 你','我 恨 你 恨 你']
y = [0,1]
countCoder = CountVectorizer(token_pattern="[a-zA-Z|\u4e00-\u9fa5]+")
X = countCoder.fit_transform(X)
print(countCoder.get_feature_names())
print(X.toarray())
# ['你', '恨', '我', '爱']
# [[1 0 1 1]
#  [2 2 1 0]]
```


特征工程——TF-IDF

原始的词项频率会面临这样一个严重问题，即在和查询进行相关度计算时，所有的词项都被认为是同等重要的^①。实际上，某些词项对于相关度计算来说几乎没有或很少有区分能力。例如，在一个有关汽车工业的文档集中，几乎所有的文档都会包含 auto，此时，auto就没有区分能力。为此，下面我们提出一种机制来降低这些出现次数过多的词项在相关性计算中的重要性。一个很直接的想法就是给文档集频率（collection frequency）较高的词项赋予较低的权重，其中文档集频率指的是词项在文档集中出现的次数。这样，便可以降低具有较高文档集频率的词项的权重。

$$\text{tf-idf}(t,d) = \text{tf}(t,d) \times \text{idf}(t)$$

特征工程——TF-IDF

- In a large text corpus, some words will be very present (e.g. “the” , “a” , “is” in English) hence carrying very little meaningful information about the actual contents of the document. If we were to feed the direct count data directly to a classifier those very frequent terms would shadow the frequencies of rarer yet more interesting terms.
- In order to re-weight the count features into floating point values suitable for usage by a classifier it is very common to use the tf-idf transform.
- Tf means **term-frequency** while tf-idf means term-frequency times **inverse document-frequency**:

$$\text{tf-idf}(t,d) = \text{tf}(t,d) \times \text{idf}(t)$$

特征工程——TF-IDF

- while tf-idf means term-frequency times **inverse document-frequency**:

$$\text{idf}(t) = \log \frac{1+n_d}{1+\text{df}(d,t)} + 1,$$

where n_d is the total number of documents, and $\text{df}(d, t)$ is the number of documents that contain term t .

- 以 “爱” 为例，其tf (爱, document1) =1

- $\text{Idf}(\text{爱}) = \log \frac{1+2}{1+1} + 1 = 1.405$

- 那么，TF-IDF (爱, document1) =1*1.405

['我爱你','我恨你恨你']

特征工程——TF-IDF

- while tf-idf means term-frequency times **inverse document-frequency**:

```
from sklearn.feature_extraction.text import TfidfVectorizer
X = ['我 爱 你','我 恨 你 恨 你']
y = [0,1]
tiCoder = TfidfVectorizer(norm=None,token_pattern="[a-zA-Z|\u4e00-\u9fa5]+")
X = tiCoder.fit_transform(X)
print(tiCoder.get_feature_names())
print(X.toarray())
# ['你', '恨', '我', '爱']
# [[ 1.      0.      1.      1.40546511]
# [ 2.      2.81093022  1.      0.      ]]
```

朴素贝叶斯模型在文档分类领域的应用

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
df = pd.read_csv("datas/bayes2.txt", header=None)
X = df[1]
Y = df[0]
tfCoder = TfidfVectorizer(token_pattern="[a-zA-Z|\u4e00-\u9fa5]+")
X = tfCoder.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0, random_state=42)
from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB()
model.fit(X_train, y_train)
print(model.predict(X_train))
print(y_train.values)
a = ["残血的安琪拉打不过鲁班", "这一波大龙别再被抢了",
      "你在石头那不要动，我去买几个橘子"]
# print(tfCoder.transform(a).todense())
print(model.predict(tfCoder.transform(a)))
```

编程——朴素贝叶斯文档分类

■ 数据集：

◆ <http://www.nlp.ir.org/download/tc-corpus-answer.rar>

■ 解释：

◆ 由复旦大学李荣陆提供。answer.rar为测试语料，共9833篇文档；train.rar为训练语料，共9804篇文档，分为20个类别。

- C3-Art
- C4-Literature
- C5-Education
- C6-Philosophy
- C7-History
- C11-Space
- C15-Energy
- C16-Electronics
- C17-Communication
- C19-Computer
- C23-Mine
- C29-Transport
- C31-Environment
- C32-Agriculture
- C34-Economy
- C35-Law
- C36-Medical
- C37-Military
- C38-Politics
- C39-Sports



THANK YOU

上海育创网络科技有限公司

朴素贝叶斯的三种事件模型*

- 多项式朴素贝叶斯：当特征是离散变量时，使用多项式模型
- 高斯朴素贝叶斯：当特征是连续变量时，使用高斯模型
- 伯努利朴素贝叶斯：伯努利模型和多项式模型是一致的，但要求特征是二值化的
(1, 0)
- 注意：当特征中既有连续变量又有离散变量时，一般将连续变量离散化后使用多项式模型

高斯朴素贝叶斯*

Person	height (feet)	weight (lbs)	foot size(inches)
male	6	180	12
male	5.92 (5' 11")	190	11
male	5.58 (5' 7")	170	12
male	5.92 (5' 11")	165	10
female	5	100	6
female	5.5 (5' 6")	150	8
female	5.42 (5' 5")	130	7
female	5.75 (5' 9")	150	9

Person	height (feet)	weight (lbs)	foot size(inches)
sample	6	130	8

Person	mean (height)	variance (height)	mean (weight)	variance (weight)	mean (foot size)	variance (foot size)
male	5.855	3.5033*10-02	176.25	1.2292*10+02	11.25	9.1667*10-01
female	5.4175	9.7225*10-02	132.5	5.5833*10+02	7.5	1.6667

$$P(\text{male}) = 0.5$$

$$p(\text{height} \mid \text{male}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(6 - \mu)^2}{2\sigma^2}\right) \approx 1.5789,$$

$$p(\text{weight} \mid \text{male}) = 5.9881 \cdot 10^{-6}$$

$$p(\text{foot size} \mid \text{male}) = 1.3112 \cdot 10^{-3}$$

$$\text{posterior numerator (male)} = \text{their product} = 6.1984 \cdot 10^{-9}$$

$$P(\text{female}) = 0.5$$

$$p(\text{height} \mid \text{female}) = 2.2346 \cdot 10^{-1}$$

$$p(\text{weight} \mid \text{female}) = 1.6789 \cdot 10^{-2}$$

$$p(\text{foot size} \mid \text{female}) = 2.8669 \cdot 10^{-1}$$

$$\text{posterior numerator (female)} = \text{their product} = 5.3778 \cdot 10^{-4}$$

$$\text{posterior (male)} = \frac{P(\text{male}) p(\text{height} \mid \text{male}) p(\text{weight} \mid \text{male}) p(\text{foot size} \mid \text{male})}{\text{evidence}}$$

$$\text{posterior (female)} = \frac{P(\text{female}) p(\text{height} \mid \text{female}) p(\text{weight} \mid \text{female}) p(\text{foot size} \mid \text{female})}{\text{evidence}}$$

1.9.2. Multinomial Naive Bayes

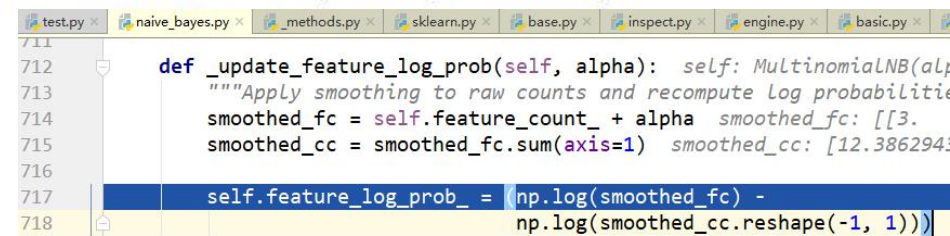
`MultinomialNB` implements the naive Bayes algorithm for multinomially distributed data, and is one of the two classic naive Bayes variants used in text classification (where the data are typically represented as word vector counts, although tf-idf vectors are also known to work well in practice). The distribution is parametrized by vectors $\theta_y = (\theta_{y1}, \dots, \theta_{yn})$ for each class y , where n is the number of features (in text classification, the size of the vocabulary) and θ_{yi} is the probability $P(x_i | y)$ of feature i appearing in a sample belonging to class y . 每一个词就是一个特征

The parameters θ_y is estimated by a smoothed version of maximum likelihood, i.e. relative frequency counting:

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

where $N_{yi} = \sum_{x \in T} x_i$ is the number of times feature i appears in a sample of class y in the training set T , and $N_y = \sum_{i=1}^{|T|} N_{yi}$ is the total count of all features for class y .

The smoothing priors $\alpha \geq 0$ accounts for features not present in the learning samples and prevents zero probabilities in further computations. Setting $\alpha = 1$ is called Laplace smoothing, while $\alpha < 1$ is called Lidstone smoothing.



```

711
712 def _update_feature_log_prob(self, alpha): self: MultinomialNB(alpha)
713 """Apply smoothing to raw counts and recompute log probabilities
714 smoothed_fc = self.feature_count_ + alpha smoothed_fc: [[3.
715 smoothed_cc = smoothed_fc.sum(axis=1) smoothed_cc: [12.386294:
716
717 self.feature_log_prob_ = (np.log(smoothed_fc) -
718                          np.log(smoothed_cc.reshape(-1, 1)))
719

```

```
696 smoothed_fc = self.feature_count_ + self.alpha smoothed_cc:
+ {ndarray} [[ 3. 2.69314718 1. 3. 2.69314718] \n [ 3. 1. 4.38629436 2. 1. ]]
```

```
697 smoothed_cc = smoothed_fc.sum(axis=1)
698 + {ndarray} [ 12.38629436 11.38629436]
699 prob = (np.log(smoothed
```

```
self.feature_log_prob_ = (np.log(smoothed_fc) -
np.log(smoothed_cc.reshape(-1, 1)))
+ {ndarray} [[-1.41797828 -1.5258801 -2.51659057 -1.41797828 -1.5258801] \n [-1.33379809 -2.43241038 -0.95392562 -1.7392632 -2.43241038]]
```

```
self.class_log_prior_ = (np.log(self.class_count_) -
np.log(self.class_count_.sum()))
else:
+ {ndarray} [ 2. 1.]
```



```
return (safe_sparse_dot(X, self.feature_log_prob_.T) +
        self.class_log_prior_)
```

```
(safe_sparse_dot(X, self.feature_log_prob_.T) +
self.class_log_prior_)
+ {csr_matrix} (0, 3)\t1.0\n (0, 2)\t1.69314718056\n (0, 0)\t1.0
```

```
return (safe_sparse_dot(X, self.feature_log_prob_.T) +
        self.class_log_prior_)
+ {ndarray} [[-1.41797828 -1.5258801 -2.51659057 -1.41797828 -1.5258801 ]\n [-1.33379809 -2.43241038 -0.95392562 -1.7392632 -2.43241038]]
```

```
self.class_log_prior_)
```

```
+ {ndarray} [-0.40546511 -1.09861229]
```

```
jll = self._joint_log_likelihood(X) jll: [[-7.50237989 -5.78681006]]
return self.classes_[np.argmax(jll, axis=1)]
```

```
+ {ndarray} [[-7.50237989 -5.78681006]]
```