朴素贝叶斯

课前准备

- 下载Anaconda软件，请点击这里进行下载。
- 复习条件概率，联合概率，事件独立性。
- 复习全概率公式与贝叶斯公式。

本节要点

- 朴素贝叶斯算法原理。
- 平滑系数的意义。
- 几种常用的朴素贝叶斯。

朴素贝叶斯算法

朴素贝叶斯算法是基于概率的分类算法，之所以称为“朴素”，是因为其假设特征之间是独立的，该算法设计比较简单，实际上使用的就是全概率公式与贝叶斯公式。朴素贝叶斯算法在文本场景中效果非常好，例如垃圾邮件过滤，新闻分类，情感分析等。

实际案例

之前，我们已经介绍了全概率公式与贝叶斯公式，现在，我们将这两个公式应用于朴素贝叶斯算法中，实现分类任务。假设我们有如下的数据集：

序号	天气（X_1）	上课距离（X_2）	学生成绩（X_3）	课程类别（X_4）	上课情况（Y）
1	晴	远	差	选修	逃课
2	晴	近	差	必修	上课
3	晴	近	好	必修	上课
4	阴	远	差	选修	逃课
5	阴	近	好	选修	上课
6	阴	近	好	必修	上课
7	雨	远	差	选修	逃课
8	雨	近	好	必修	上课
9	雨	近	差	必修	逃课
10	雨	远	好	选修	逃课
11	阴	近	差	选修	?
12	晴	远	好	选修	?

该数据集展示在不同的条件下，学生的上课与逃课情况。现在问题是，对于第11条记录，学生会上课还是逃课?

算法原理

计算概率

之前我们提过，朴素贝叶斯是基于概率的分类算法，因此，想要预测未知样本X所属的类别，只需要计算X属于每个类别（y）的概率是多少，预测结果就是概率最大的那个类别。以上例来讲，就是比较：

$$P(y = \text{上课} \mid X)$$
$$P(y = \text{逃课} \mid X)$$

哪个概率大。

贝叶斯转换

假设X含有n个特征（上面的示例中含有4个特征），即我们要计算：

$$P(y \mid x_1, \ldots, x_n)$$

然而，以上的概率我们并不容易求解，不过，根据贝叶斯公式：

$$P(B_i \mid A) = \frac{P(B_i A)}{P(A)} = \frac{P(AB_i)}{P(A)} = \frac{P(A|B_i)P(B_i)}{P(A)}$$
$$= \frac{P(A|B_i)P(B_i)}{\sum_{j=1}^{n} P(A|B_j)P(B_j)}$$

- A：实验E的事件，$P(A) > 0$。
- B_1，B_2，...，B_n：样本空间S的一个划分，$P(B_i) > 0 (i = 1, 2, \cdots, n)$。

我们可以进行如下的转换：

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots x_n|y)}{P(x_1, \ldots, x_n)} \quad \ldots (1)$$

特征独立

因为朴素贝叶斯算法的前提假设为，各个特征之间都是独立的，因此有：

$$P(x_1,\dots,x_n \mid y) = P(x_1 \mid y)P(x_2 \mid y)\dots P(x_n \mid y)$$
$$= \prod_{i=1}^{n} P(x_i \mid y) \quad \dots(2)$$

因此，将（2）式代入（1）式，可得：

$$P(y \mid x_1,\dots,x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1,\dots,x_n)}$$

类别判定

我们发现，无论是计算样本属于哪个类别的概率，分母部分都是不变的，因此，比较概率的大小，只需要比较分子部分就可以了。

$$P(y \mid x_1,\dots,x_n) \propto P(y)\prod_{i=1}^{n} P(x_i \mid y)$$

- $\propto$：正比于。

故算法最终预测的类别，就是能够使得分子部分最大的那个类别，即：

$$\hat{y} = \arg\max_y P(y)\prod_{i=1}^{n} P(x_i \mid y)$$

从公式中，我们容易发现，若要预测样本的类别，只需要求解$P(y)$与$P(x_i \mid y)$即可。而这两个概率，都可以从训练集中获取。

案例求解

对于样本11，我们就可以分别计算下，在上课与逃课的情况下，各自的概率值是多少。

$$
\begin{aligned}
&P(y=\text{上课})\prod_{i=1}^{n}P(x_i \mid y=\text{上课})\\
&=P(y=\text{上课})P(x_1=\text{阴} \mid y=\text{上课})P(x_2=\text{近} \mid y=\text{上课})\\
&P(x_3=\text{差} \mid y=\text{上课})P(x_4=\text{选修} \mid y=\text{上课})\\
&=0.5\times 0.4\times 1\times 0.2\times 0.2\\
&=0.008\\
&P(y=\text{逃课})\prod_{i=1}^{n}P(x_i \mid y=\text{逃课})\\
&=P(y=\text{逃课})P(x_1=\text{阴} \mid y=\text{逃课})P(x_2=\text{近} \mid y=\text{逃课})\\
&P(x_3=\text{差} \mid y=\text{逃课})P(x_4=\text{选修} \mid y=\text{逃课})\\
&=0.5\times 0.2\times 0.2\times 0.8\times 0.8\\
&=0.0128
\end{aligned}
$$

由此，我们可知，学生逃课的概率，大于上课的概率，因此，最终算法对样本11预测的结果类别为：逃课。

上述的概率之和不为1，我们得出这样的结论还正确吗？

A 正确。

B 不正确。

平滑改进

当我们以同样的方式，试图计算样本12所属的类别时，会出现一点小问题。那就是：

$$P(x_i=\text{远} \mid y=\text{上课})$$

该概率的值为0。由于最终概率是使用各个概率的乘积计算的，因此，一旦有一个概率为0，即使其他的概率值较大，也一律会得到0值。这会严重影响预测的准确性，为了避免这种情况的发生，我们在计算概率时，采用平滑改进。

$$P(x_i \mid y) = \frac{\text{类别}y\text{中}x_i\text{出现的次数}+\alpha}{\text{类别}y\text{的总数量}+k*\alpha}$$

- k：特征x_i可能的取值数。
- α：平滑系数。（$\alpha \geqslant 0$）
  - 当$\alpha = 1$时，称为拉普拉斯平滑（Laplace smoothing）。
  - 当$\alpha < 1$时，称为Lidstone smoothing平滑。

算法优点

相对于其他算法，朴素贝叶斯算法具有如下优势：

- 即使训练集数据较少，也能够实现预测，并且效果不错。
- 算法的训练速度非常快。

这是因为，算法假设特征之间是独立的，这意味着每个特征可以单独当成一维分布而进行评估，无需考虑与其他特征之间的关联性。反之，如果特征之间不独立，则为了获得较准确的数据分布，就需要更多的训练样本。假设训练集中的含有N个特征，每个特征需要M个样本来训练，则总共需要的样本数为各自样本之间的笛卡尔积，即M^N，这在N很大时，训练会非常缓慢，然而，如果特征之间独立，对于每个特征，就可以单独进行考虑，总共只需要$N * M$个样本就可以正常训练。

关于朴素贝叶斯算法，说法正确的是（ ）【不定项】
A 朴素贝叶斯算法是基于概率的分类算法。
B 朴素贝叶斯算法之所以“朴素”，是因为该算法前提假设各个特征是独立的。
C 朴素贝叶斯算法训练速度较快。
D 朴素贝叶斯算法在较少的训练样本集上，也可能工作的很好。

常用朴素贝叶斯

在sklearn中，提供了若干种朴素贝叶斯的实现算法，不同的朴素贝叶斯算法，主要是对$P(x_i \mid y)$的分布假设不同，进而采用不同的参数估计方式。实际上，通过之前的介绍，我们也应该能够发现，朴素贝叶斯算法，主要就是计算$P(x_i \mid y)$，一旦$P(x_i \mid y)$确定，最终属于每个类别的概率，自然也就迎刃而解了。

sklearn中提供的朴素贝叶斯为：

- 分类朴素贝叶斯
- 高斯朴素贝叶斯
- 伯努利朴素贝叶斯
- 多项式朴素贝叶斯
- 互补朴素贝叶斯

分类朴素贝叶斯

分类朴素贝叶斯适用于类别变量。对于特征i，$P(x_i \mid y)$计算如下：

$$P(x_i = t \mid y) = \frac{N_{yit} + \alpha}{N_y + \alpha n_i}$$

- N_{yit}：第i个特征中，属于类别y，类别值为t的样本个数。
- N_y：属于类别y的所有样本个数。
- α：平滑系数。
- n_i：第i个特征的类别取值数量。

```
import numpy as np
import pandas as pd

data = pd.read_csv("data.csv")
display(data)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	天气	上课距离	学生成绩	课程类别	上课情况
0	晴	远	差	选修	逃课
1	晴	近	差	必修	上课
2	晴	近	好	必修	上课
3	阴	远	差	选修	逃课
4	阴	近	好	选修	上课
5	阴	近	好	必修	上课
6	雨	远	差	选修	逃课
7	雨	近	好	必修	上课
8	雨	近	差	必修	逃课
9	雨	远	好	选修	逃课
10	阴	近	差	选修	?
11	晴	远	好	选修	?

然而，目前数据的类型不是数值类型，我们需要进行数据转换（特征工程）。

```
from sklearn.preprocessing import OrdinalEncoder, LabelEncoder

X, y = data.iloc[:10, :-1], data.iloc[:10, -1]
oe = OrdinalEncoder()
X = oe.fit_transform(X)
print(X)
# 输出每个特征的类别信息。
print(oe.categories_)
```

```
[[0. 1. 1. 1.]
 [0. 0. 1. 0.]
 [0. 0. 0. 0.]
 [1. 1. 1. 1.]
 [1. 0. 0. 1.]
 [1. 0. 0. 0.]
 [2. 1. 1. 1.]
 [2. 0. 0. 0.]
 [2. 0. 1. 0.]
 [2. 1. 0. 1.]]
[array(['晴', '阴', '雨'], dtype=object), array(['近', '远'], dtype=object),
array(['好', '差'], dtype=object), array(['必修', '选修'], dtype=object)]
```

```
le = LabelEncoder()
y = le.fit_transform(y)
print(y)
# 输出标签的类别信息。
print(le.classes_)
```

```
[1 0 0 1 0 0 1 0 1 1]
['上课' '逃课']
```

```python
from sklearn.naive_bayes import CategoricalNB

cnb = CategoricalNB()
cnb.fit(X, y)
# 每个类别中，每个特征各个取值出现的次数。
# 该值为列表类型，列表中元素形状依次为(类别数量，对应特征的类别取值数量）。
print(cnb.category_count_)
# 每个类别的样本数量。
print(cnb.class_count_)
# 每个类别的对数概率，如果想查看原始概率，需要使用指数还原。
print(np.exp(cnb.class_log_prior_))
# 类别的标签值。
print(cnb.classes_)
# 计算P(x_i|y)的概率。
print([np.exp(item) for item in cnb.feature_log_prob_])
# 特征的数量。
print(cnb.n_features_)
```

```
[array([[2., 2., 1.],
       [1., 1., 3.]]), array([[5., 0.],
       [1., 4.]]), array([[4., 1.],
       [1., 4.]]), array([[4., 1.],
       [1., 4.]])]
[5. 5.]
[0.5 0.5]
[0 1]
[array([[0.375, 0.375, 0.25 ],
       [0.25 , 0.25 , 0.5  ]]), array([[0.85714286, 0.14285714],
       [0.28571429, 0.71428571]]), array([[0.71428571, 0.28571429],
       [0.28571429, 0.71428571]]), array([[0.71428571, 0.28571429],
       [0.28571429, 0.71428571]])]
4
```

```python
# 对最后两条记录进行预测。注意，测试集的数据需要与训练集
# 执行相同的转换。
test = data.iloc[10:, :-1]
test = oe.transform(test)
y_hat = cnb.predict(test)
print(y_hat)
pro = cnb.predict_proba(test)
print(pro)
```

```
[1 1]
[[0.41860465 0.58139535]
 [0.23076923 0.76923077]]
```

现在，我们可以算一下样本11（未知数据集中第一条记录）的真实概率值。

```
a = 0.008
b = 0.0128
c = a + b
print(a / c, b / c)
```

```
0.38461538461538464 0.6153846153846154
```

我们自己计算的概率，与sklearn计算的概率不同，这是为什么呢?
A 我们算错了。
B sklearn算错了。
C 都没有错。
D 都错了。

```
# alpha：指定平滑系数，默认为1（拉普拉斯平滑）。
cnb = CategoricalNB(alpha=0)
cnb.fit(X, y)
pro = cnb.predict_proba(test)
print(pro)
```

```
[[3.84615385e-01 6.15384615e-01]
 [5.00000000e-11 1.00000000e+00]]
```

```
D:\software\anaconda3\lib\site-packages\sklearn\naive_bayes.py:507: UserWarning: alpha too small will result in numeric errors, setting alpha = 1.0e-10
  'setting alpha = %.1e' % _ALPHA_MIN)
```

高斯朴素贝叶斯

高斯朴素贝叶斯，适用于连续变量，其假定各个特征x_i在各个类别y下是服从正态分布的，即$x_i \sim N(\mu_y, \sigma_y^2)$，算法内部使用正态分布的概率密度函数来计算概率$P(x_i \mid y)$，如下：

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

- μ_y：在类别为y的样本中，特征x_i的均值。
- σ_y：在类别为y的样本中，特征x_i的标准差。

```
from sklearn.naive_bayes import GaussianNB

np.random.seed(0)
```

```
X = np.random.randint(0, 10, size=(6, 2))
y = np.array([0, 0, 0, 1, 1, 1])
data = pd.DataFrame(np.concatenate([X, y.reshape(-1, 1)], axis=1), columns=["x1", "x2", "y"])
display(data)

gnb = GaussianNB()
gnb.fit(X, y)
# 每个类别的先验概率。P(y)
print("概率：", gnb.class_prior_)
# 每个类别样本的数量。
print("样本数量：", gnb.class_count_)
# 每个类别的标签。
print("标签", gnb.classes_)
# 每个特征在每个类别下的均值。
print("均值：", gnb.theta_)
# 每个特征在每个类别下的方差。
print("方差：", gnb.sigma_)

# 测试集
X_test = np.array([[6, 3]])
print("预测结果：", gnb.predict(X_test))
print("预测结果概率：", gnb.predict_proba(X_test))
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	x1	x2	y
0	5	0	0
1	3	3	0
2	7	9	0
3	3	5	1
4	2	4	1
5	7	6	1

```
概率： [0.5 0.5]
样本数量： [3. 3.]
标签 [0 1]
均值： [[5. 4.]
 [4. 5.]]
方差： [[ 2.66666667 14.00000001]
 [ 4.66666667  0.66666667]]
预测结果： [0]
预测结果概率： [[0.87684687 0.12315313]]
```

伯努利朴素贝叶斯

设实验E只有两个可能的结果：A与$\bar{A}$，则称E为**伯努利试验。**

伯努利朴素贝叶斯，适用于离散变量，其假设各个特征x_i在各个类别y下是服从n重伯努利分布（二项分布）的，因为伯努利试验仅有两个结果，因此，算法会首先对特征值进行二值化处理（假设二值化的结果为1与0）。

$P(x_i \mid y)$计算方式如下：

$$P(x_i \mid y) = P(x_i = 1 \mid y)x_i + (1 - P(x_i = 1 \mid y))(1 - x_i)$$

在训练集中，会进行如下的估计：

$$P(x_i = 1 \mid y) = \frac{N_{yi} + \alpha}{N_y + 2 * \alpha}$$
$$P(x_i = 0 \mid y) = 1 - P(x_i = 1 \mid y)$$

- N_{yi}：第i个特征中，属于类别y，数值为1的样本个数。
- N_y：属于类别y的所有样本个数。
- α：平滑系数。

因为在数据集中，只有两种取值（1与0），因此，对于给定的类别与特征，只需要计算$P(x_i = 1 \mid y)$就可以了，而$P(x_i = 0 \mid y)$的概率，用1减去$P(x_i = 1 \mid y)$即可得出。

```
from sklearn.naive_bayes import BernoulliNB

np.random.seed(0)
X = np.random.randint(-5, 5, size=(6, 2))
y = np.array([0, 0, 0, 1, 1, 1])
data = pd.DataFrame(np.concatenate([X, y.reshape(-1, 1)], axis=1), columns=["x1",
"x2", "y"])
display(data)
```

```
bnb = BernoulliNB()
bnb.fit(X, y)
# 每个特征在每个类别下发生（出现）的次数。因为伯努利分布只有两个值，
# 我们只需要计算出现的概率P(x=1|y)，不出现的概率P(x=0|y)使用1减去P(x=1|y)即可。
print("数值1出现次数：", bnb.feature_count_)
# 每个类别样本所占的比重，即P(y)。注意，该值为概率取对数之后的结果，如果需要查看
# 原有的概率，需要使用指数还原。
print("类别占比p(y)：", np.exp(bnb.class_log_prior_))
# 每个类别下，每个特征（值为1）所占的比例（概率），即P(x|y)。注意，该值为概率
# 取对数之后的结果，如果需要查看原有的概率，需要使用指数还原。
print("特征概率：", np.exp(bnb.feature_log_prob_))
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	x1	x2	y
0	0	-5	0
1	-2	-2	0
2	2	4	0
3	-2	0	1
4	-3	-1	1
5	2	1	1

```
数值1出现次数： [[1. 1.]
 [1. 1.]]
类别占比p(y)： [0.5 0.5]
特征概率： [[0.4 0.4]
 [0.4 0.4]]
```

多项式朴素贝叶斯

多项式朴素贝叶斯，适用于离散变量，其假设各个特征x_i在各个类别y下是服从多项式分布的，故每个特征值不能是负数。

$P(x_i \mid y)$计算如下：

$$P(x_i \mid y) = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

- N_{yi}：特征i在类别y的样本中发生（出现）的次数。
- N_y：类别y的样本中，所有特征发生（出现）的次数。

- n：特征数量。
- α：平滑系数。

```
from sklearn.naive_bayes import MultinomialNB

np.random.seed(0)
X = np.random.randint(0, 4, size=(6, 2))
y = np.array([0, 0, 0, 1, 1, 1])
data = pd.DataFrame(np.concatenate([X, y.reshape(-1, 1)], axis=1), columns=["x1", "x2", "y"])
display(data)

mnb = MultinomialNB()
mnb.fit(X, y)
# 每个类别的样本数量。
print(mnb.class_count_)
# 每个特征在每个类别下发生（出现）的次数。
print(mnb.feature_count_)
# 每个类别下，每个特征所占的比例（概率），即P(x|y)。注意，该值为概率
# 取对数之后的结果，如果需要查看原有的概率，需要使用指数还原。
print(np.exp(mnb.feature_log_prob_))
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	x1	x2	y
0	0	3	0
1	1	0	0
2	3	3	0
3	3	3	1
4	1	3	1
5	1	2	1

```
[3. 3.]
[[4. 6.]
 [5. 8.]]
[[0.41666667 0.58333333]
 [0.4        0.6       ]]
```

互补朴素贝叶斯

互补朴素贝叶斯与多项式朴素贝叶斯类似，我们可以认为是对多项式朴素贝叶斯的一种改进。在互补朴素贝叶斯中，计算的不再是每个特征在对应类别中出现的概率，而是计算每个特征不在对应类别中出现的概率，这也正是互补朴素贝叶斯命名的由来。

在互补朴素贝叶斯中，计算公式如下：

$$\hat{\theta}_{yi} = \frac{N_{\tilde{y}i} + \alpha}{N_{\tilde{y}} + \alpha n}$$

$$w_{yi} = log\hat{\theta}_{yi}$$

$$w_{yi} = \frac{w_{yi}}{\sum_i w_{yi}}$$

$$\hat{y} = \arg\min_y \sum_i t_i w_{yi}$$

- $N_{\tilde{y}i}$：特征i在不属于类别y的样本中发生（出现）的次数。
- $N_{\tilde{y}}$：不属于类别y的样本中，所有特征发生（出现）的次数。
- $\hat{\theta}_{yi}$：特征i不在类别y中发生的概率。
- n：特征数量。
- α：平滑系数。
- t_i：样本中特征i的具体值（发生次数）。

互补朴素贝叶斯更适合于样本不均衡的数据集中。在文本分类的任务中，互补朴素贝叶斯的表现往往优于多项式朴素贝叶斯。

```
from sklearn.naive_bayes import ComplementNB

np.random.seed(0)
X = np.random.randint(0, 4, size=(6, 2))
y = np.array([0, 0, 0, 1, 1, 1])
data = pd.DataFrame(np.concatenate([X, y.reshape(-1, 1)], axis=1), columns=["x1",
"x2", "y"])
display(data)

cnb = ComplementNB()
cnb.fit(X, y)
# 每个类别的样本数量。
print(cnb.class_count_)
# 每个特征在每个类别下发生（出现）的次数。
print(cnb.feature_count_)
# 特征i不在指定类别中发生的概率（对数概率的相反数）。
print(cnb.feature_log_prob_)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	x1	x2	y
0	0	3	0
1	1	0	0
2	3	3	0
3	3	3	1
4	1	3	1
5	1	2	1

```
[3. 3.]
[[4. 6.]
 [5. 8.]]
[[0.91629073 0.51082562]
 [0.87546874 0.5389965 ]]
```

程序示例

如果对鸢尾花数据集进行分类，哪种朴素贝叶斯算法效果可能会更好?
A 分类朴素贝叶斯
B 高斯朴素贝叶斯
C 伯努利朴素贝叶斯
D 多项式朴素贝叶斯
E 互补朴素贝叶斯
F 差不多

```python
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

X, y = load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=0)
models = [("分类朴素贝叶斯", CategoricalNB()),
          ("高斯朴素贝叶斯：", GaussianNB()),
          ("伯努利朴素贝叶斯：", BernoulliNB()),
          ("多项式朴素贝叶斯：", MultinomialNB()),
          ("互补朴素贝叶斯：", ComplementNB())
           ]
for name, m in models:
    m.fit(X_train, y_train)
    print(name, m.score(X_test, y_test))
```

```
分类朴素贝叶斯 0.8947368421052632
高斯朴素贝叶斯： 1.0
伯努利朴素贝叶斯： 0.23684210526315788
多项式朴素贝叶斯： 0.5789473684210527
互补朴素贝叶斯： 0.5789473684210527
```

拓展点

- 几种不同的数据分布。
  - 高斯分布（正态分布）
  - n重伯努利分布（二项分布）
  - 多项式分布
- 不同分布下，朴素贝叶斯的参数估计计算方式，可参考scikit-learn官方文档。

作业

1. 设盒子中有5个球，其中1个球内部含有中奖信息。现在5个人按顺序从盒子中拿1个球，请问是先拿球的中奖率高，还是后拿球的中奖率高?
2. 在我们获取概率时，获取的是概率的对数，而不是概率本身，为什么要这么设计呢?
3. 总结朴素贝叶斯在特征分布上的局限性。