

机器学习实战教程（四）：朴素贝叶斯基础篇之言论过滤器

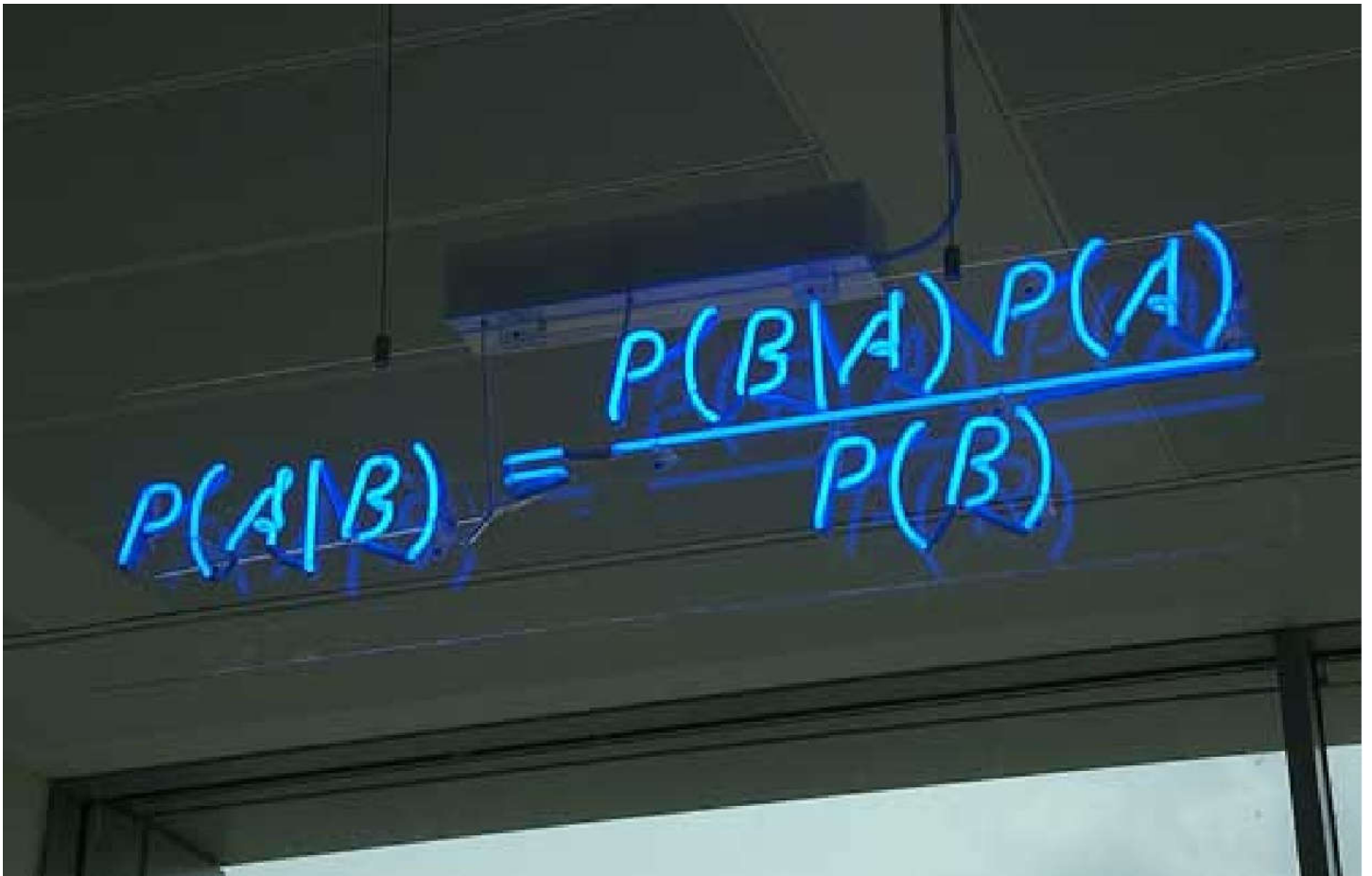
🕒 2017年11月7日 10:38:18 💬 102 🌡 20,163 °C 🛠 编辑



一、前言

朴素贝叶斯算法是有监督的学习算法，解决的是分类问题，如客户是否流失、是否值得投资、信用等级评定等多分类问题。**该算法的优点在于简单易懂、学习效率高、在某些领域的分类问题中能够与决策树、神经网络相媲美。**但由于该算法以自变量之间的独立（条件特征独立）性和连续变量的正态性假设为前提，就会导致算法精度在某种程度上受影响。

本篇文章将从朴素贝叶斯推断原理开始讲起，通过实例进行辅助讲解。最后，使用Python3编程实现一个简单的言论过滤器。

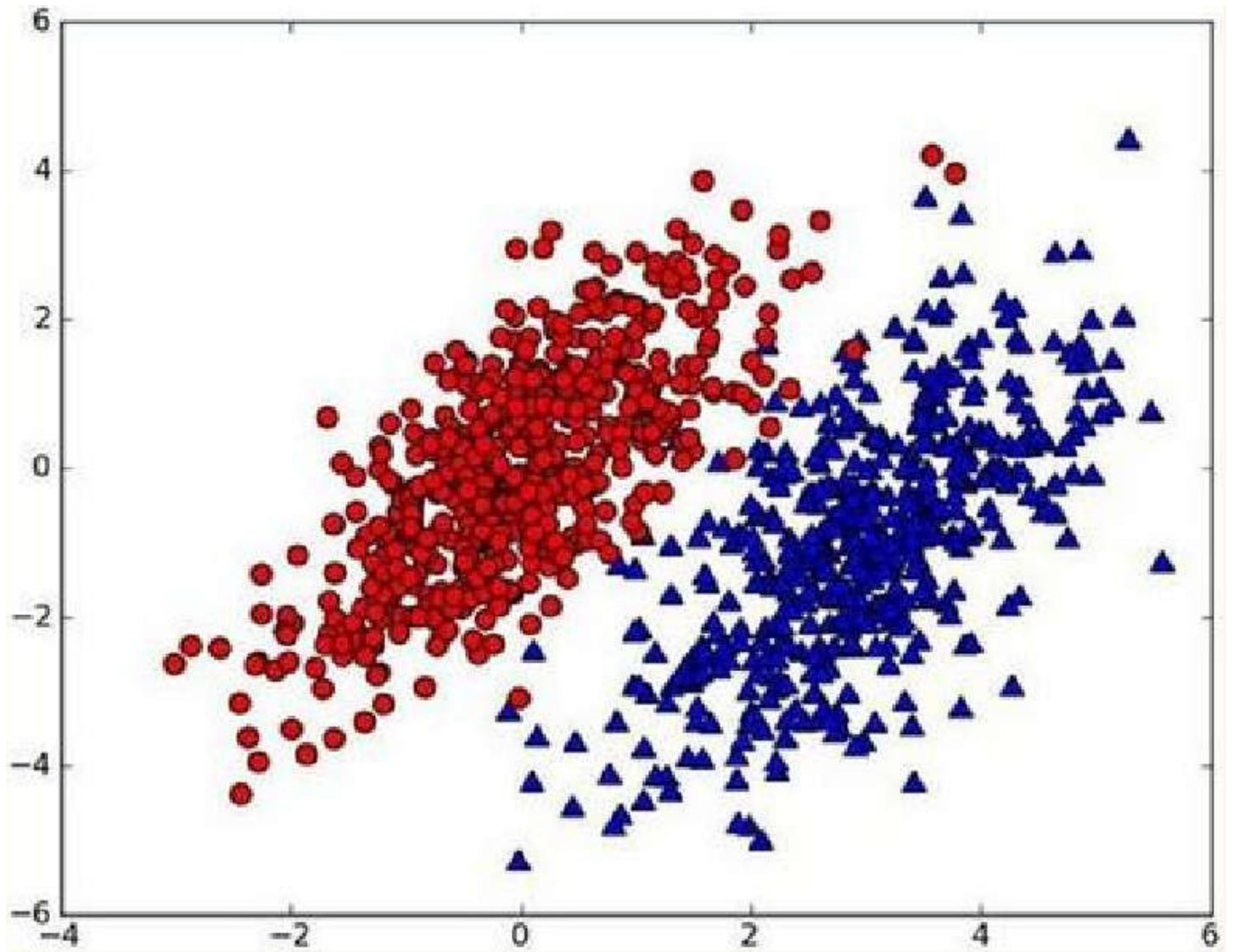

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

二、朴素贝叶斯理论

朴素贝叶斯是贝叶斯决策理论的一部分，所以在讲述朴素贝叶斯之前有必要快速了解一下贝叶斯决策理论。

1、贝叶斯决策理论

假设现在我们有一个数据集，它由两类数据组成，数据分布如下图所示：



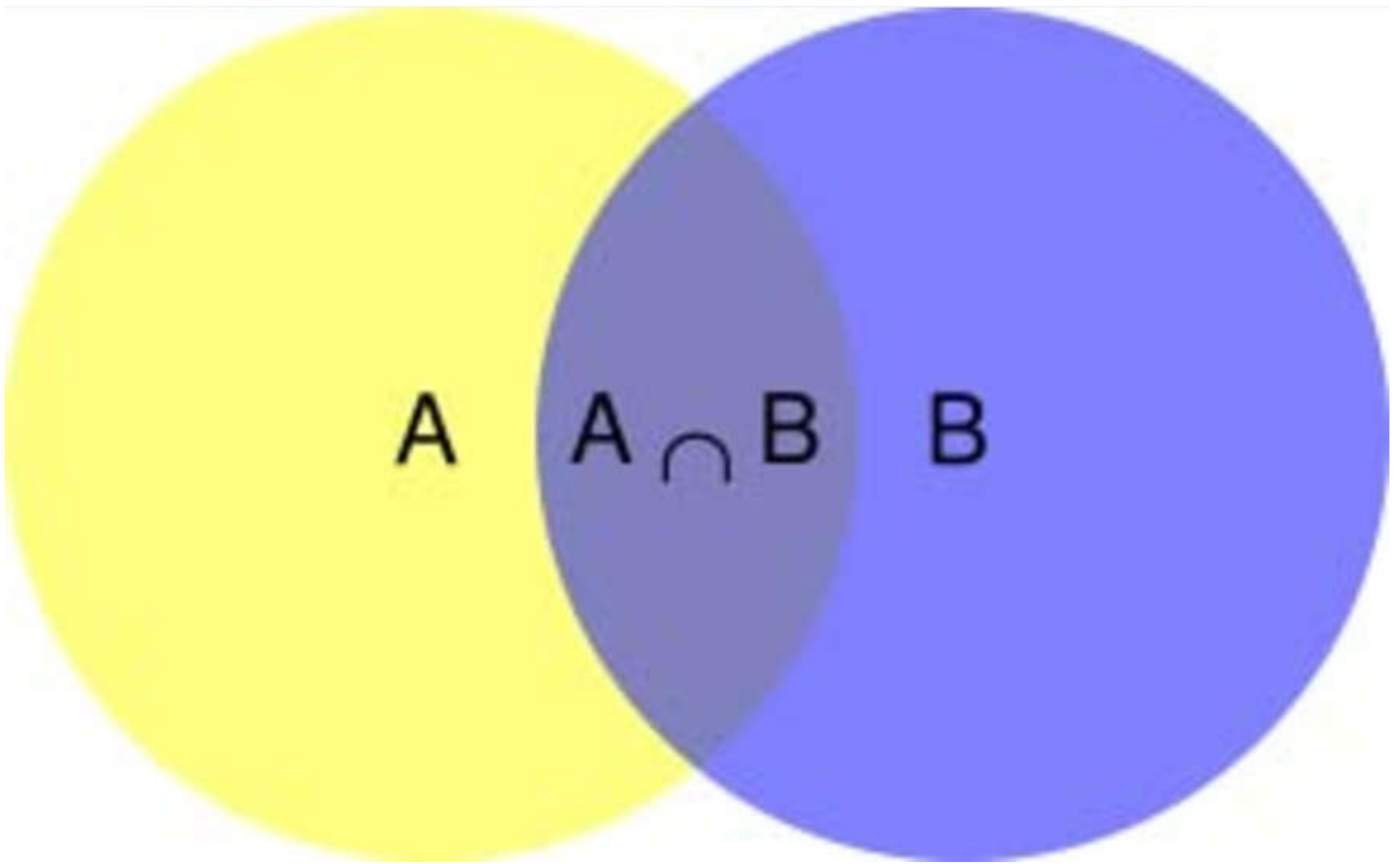
我们现在用 $p_1(x,y)$ 表示数据点 (x,y) 属于类别1(图中红色圆点表示的类别)的概率, 用 $p_2(x,y)$ 表示数据点 (x,y) 属于类别2(图中蓝色三角形表示的类别)的概率, 那么对于一个新数据点 (x,y) , 可以用下面的规则来判断它的类别:

- 如果 $p_1(x,y) > p_2(x,y)$, 那么类别为1
- 如果 $p_1(x,y) < p_2(x,y)$, 那么类别为2

也就是说, 我们会选择高概率对应的类别。这就是贝叶斯决策理论的核心思想, 即选择具有最高概率的决策。已经了解了贝叶斯决策理论的核心思想, 那么接下来, 就是学习如何计算 p_1 和 p_2 概率。

2、条件概率

在学习计算 p_1 和 p_2 概率之前, 我们需要了解什么是条件概率(Conditional probability), 就是指在事件B发生的情况下, 事件A发生的概率, 用 $P(A|B)$ 来表示。



根据文氏图，可以很清楚地看到在事件B发生的情况下，事件A发生的概率就是 $P(A \cap B)$ 除以 $P(B)$ 。

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

因此，

$$P(A \cap B) = P(A|B)P(B)$$

同理可得，

$$P(A \cap B) = P(B|A)P(A)$$

所以，

$$P(A|B)P(B) = P(B|A)P(A)$$

即

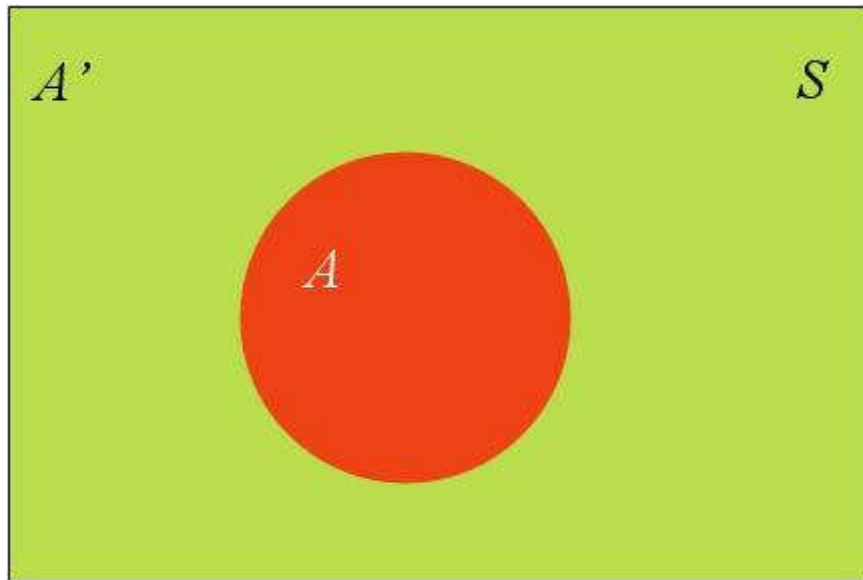
$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

这就是条件概率的计算公式。

3、全概率公式

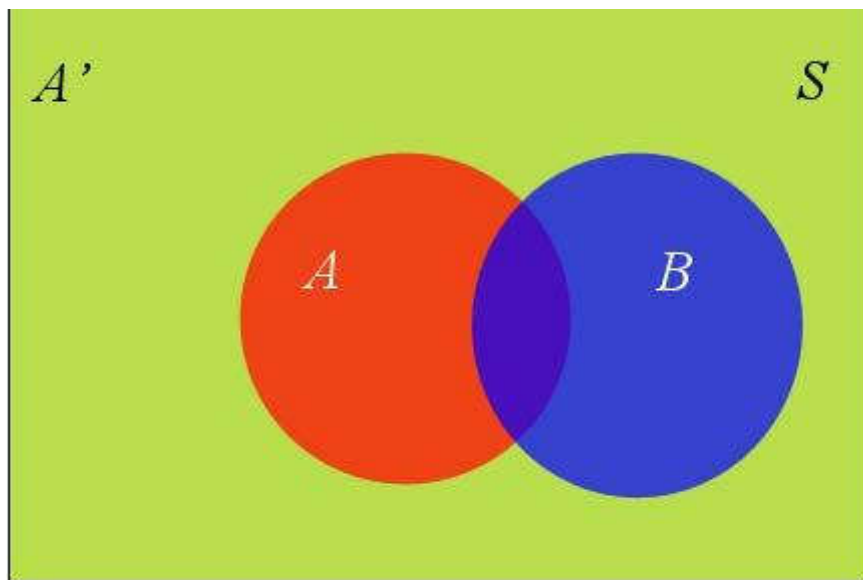
除了条件概率以外，在计算 p_1 和 p_2 的时候，还要用到全概率公式，因此，这里继续推导全概率公式。

假定样本空间 S ，是两个事件 A 与 A' 的和。



上图中，红色部分是事件 A ，绿色部分是事件 A' ，它们共同构成了样本空间 S 。

在这种情况下，事件 B 可以划分成两个部分。



即

$$P(B) = P(B \cap A) + P(B \cap A')$$

在上一节的推导当中，我们已知

$$P(B \cap A) = P(B|A)P(A)$$

所以，

$$P(B) = P(B|A)P(A) + P(B|A')P(A')$$

这就是全概率公式。它的含义是，如果A和A'构成样本空间的一个划分，那么事件B的概率，就等于A和A'的概率分别乘以B对这两个事件的条件概率之和。

将这个公式代入上一节的条件概率公式，就得到了条件概率的另一种写法：

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

4、贝叶斯推断

对条件概率公式进行变形，可以得到如下形式：

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

我们把P(A)称为"先验概率" (Prior probability)，即在B事件发生之前，我们对A事件概率的一个判断。

P(A|B)称为"后验概率" (Posterior probability)，即在B事件发生之后，我们对A事件概率的重新评估。

P(B|A)/P(B)称为"可能性函数" (Likelyhood)，这是一个调整因子，使得预估概率更接近真实概率。

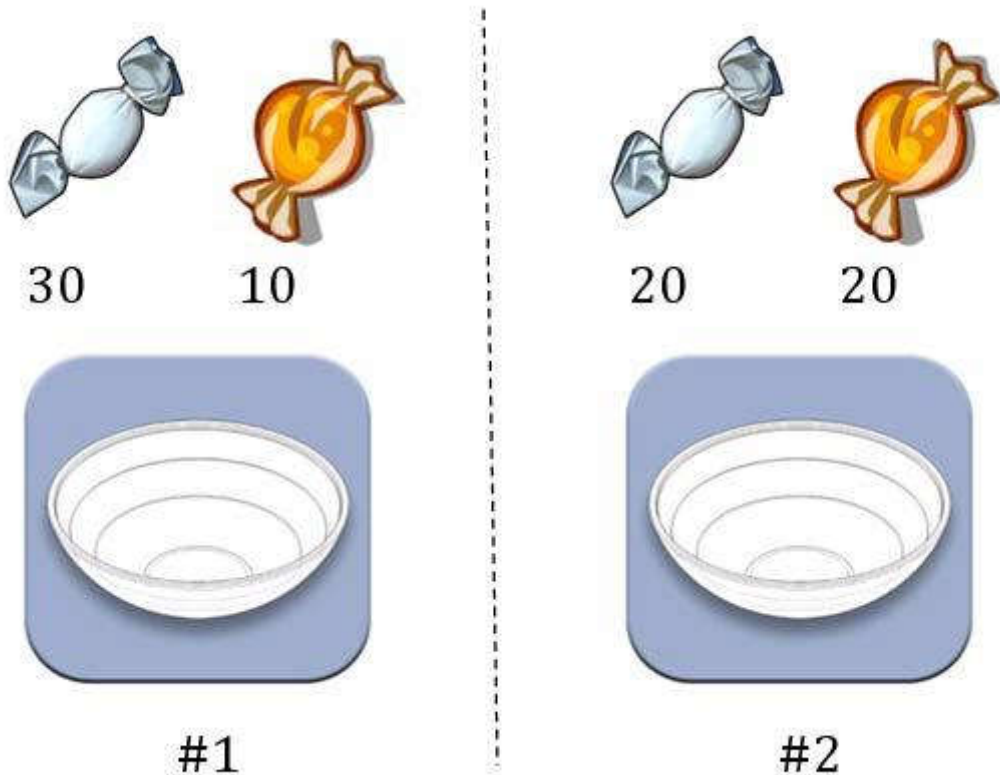
所以，条件概率可以理解成下面的式子：

			Shell
1	后验概率	= 先验概率 x 调整因子	

这就是贝叶斯推断的含义。我们先预估一个"先验概率"，然后加入实验结果，看这个实验到底是增强还是削弱了"先验概率"，由此得到更接近事实的"后验概率"。

在这里，如果"可能性函数" $P(B|A)/P(B) > 1$ ，意味着"先验概率"被增强，事件A的发生的可能性变大；如果"可能性函数" = 1，意味着B事件无助于判断事件A的可能性；如果"可能性函数" < 1 ，意味着"先验概率"被削弱，事件A的可能性变小。

为了加深对贝叶斯推断的理解，我们举一个例子。



两个一模一样的碗，一号碗有30颗水果糖和10颗巧克力糖，二号碗有水果糖和巧克力糖各20颗。现在随机选择一个碗，从中摸出一颗糖，发现是水果糖。请问这颗水果糖来自一号碗的概率有多大？

我们假定， H_1 表示一号碗， H_2 表示二号碗。由于这两个碗是一样的，所以 $P(H_1)=P(H_2)$ ，也就是说，在取出水果糖之前，这两个碗被选中的概率相同。因此， $P(H_1)=0.5$ ，我们把这个概率就叫做"先验概率"，即没有做实验之前，来自一号碗的概率是0.5。

再假定， E 表示水果糖，所以问题就变成了在已知 E 的情况下，来自一号碗的概率有多大，即求 $P(H_1|E)$ 。我们把这个概率叫做"后验概率"，即在 E 事件发生之后，对 $P(H_1)$ 的修正。

根据条件概率公式，得到

$$P(H_1|E) = P(H_1) \frac{P(E|H_1)}{P(E)}$$

已知， $P(H_1)$ 等于0.5， $P(E|H_1)$ 为一号碗中取出水果糖的概率，等于 $30 \div (30+10)=0.75$ ，那么求出 $P(E)$ 就可以得到答案。根据全概率公式，

$$P(E) = P(E|H_1)P(H_1) + P(E|H_2)P(H_2)$$

所以，

$$P(E) = 0.75 \times 0.5 + 0.5 \times 0.5 = 0.625$$

将数字代入原方程，得到

$$P(H1|E) = 0.5 \times \frac{0.75}{0.625} = 0.6$$

这表明，来自一号碗的概率是0.6。也就是说，取出水果糖之后，H1事件的可能性得到了增强。

同时再思考一个问题，在使用该算法的时候，如果不需要知道具体的类别概率，即上面 $P(H1|E)=0.6$ ，只需要知道所属类别，即来自一号碗，我们有必要计算 $P(E)$ 这个全概率吗？要知道我们只需要比较 $P(H1|E)$ 和 $P(H2|E)$ 的大小，找到那个最大的概率就可以。既然如此，两者的分母都是相同的，那我们只需要比较分子即可。即比较 $P(E|H1)P(H1)$ 和 $P(E|H2)P(H2)$ 的大小，所以为了减少计算量，全概率公式在实际编程中可以不使用。

5、朴素贝叶斯推断

理解了贝叶斯推断，那么让我们继续看看朴素贝叶斯。贝叶斯和朴素贝叶斯的概念是不同的，区别就在于“朴素”二字，朴素贝叶斯对条件个概率分布做了条件独立性的假设。比如下面的公式，假设有 n 个特征：

$$P(a|X) = p(X|a)p(a) = p(x_1, x_2, x_3, \dots, x_n|a)p(a)$$

由于每个特征都是独立的，我们可以进一步拆分公式：

$$\begin{aligned} p(a|X) &= p(X|a)p(a) \\ &= \{p(x_1|a) * p(x_2|a) * p(x_3|a) * \dots * p(x_n|a)\}p(a) \end{aligned}$$

这样我们就可以进行计算了。如果有些迷糊，让我们从一个例子开始讲起，你会看到贝叶斯分类器很好懂，一点都不难。

某个医院早上来了六个门诊的病人，他们的情况如下表所示：

症状	职业	疾病
打喷嚏	护士	感冒
打喷嚏	农夫	过敏
头痛	建筑工人	脑震荡
头痛	建筑工人	感冒
打喷嚏	教师	感冒
头痛	教师	脑震荡

现在又来了第七个病人，是一个打喷嚏的建筑工人。请问他患上感冒的概率有多大？

根据贝叶斯定理：

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

可得：

$$\begin{aligned} & P(\text{感冒} | \text{打喷嚏} \times \text{建筑工人}) \\ &= \frac{P(\text{打喷嚏} \times \text{建筑工人} | \text{感冒}) \times P(\text{感冒})}{P(\text{打喷嚏} \times \text{建筑工人})} \end{aligned}$$

根据朴素贝叶斯条件独立性的假设可知，“打喷嚏”和“建筑工人”这两个特征是独立的，因此，上面的等式就变成了

$$\begin{aligned} & P(\text{感冒} | \text{打喷嚏} \times \text{建筑工人}) \\ &= \frac{P(\text{打喷嚏} | \text{感冒}) \times P(\text{建筑工人} | \text{感冒}) \times P(\text{感冒})}{P(\text{打喷嚏}) \times P(\text{建筑工人})} \end{aligned}$$

这里可以计算：

$$P(\text{感冒} | \text{打喷嚏} \times \text{建筑工人}) = \frac{0.66 \times 0.33 \times 0.5}{0.5 \times 0.33} = 0.66$$

因此，这个打喷嚏的建筑工人，有66%的概率是得了感冒。同理，可以计算这个病人患上过敏或脑震荡的概率。比较这几个概率，就可以知道他最可能得什么病。

这就是贝叶斯分类器的基本方法：在统计资料的基础上，依据某些特征，计算各个类别的概率，从而实现分类。

同样，在编程的时候，如果不需要求出所属类别的具体概率， $P(\text{打喷嚏}) = 0.5$ 和 $P(\text{建筑工人}) = 0.33$ 的概率是可以不用求的。

三、动手实战

说了这么多，没点实践编程怎么行？

以在线社区留言为例。为了不影响社区的发展，我们要屏蔽侮辱性的言论，所以要构建一个快速过滤器，如果某条留言使用了负面或者侮辱性的语言，那么就将该留言标志为内容不当。过滤这类内容是一个很常见的需求。对此问题建立两个类型：侮辱类和非侮辱类，使用1和0分别表示。

我们把文本看成单词向量或者词条向量，也就是说将句子转换为向量。考虑出现所有文档中的单词，再决定将哪些单词纳入词汇表或者说所要的词汇集合，然后必须要将每一篇文档转换为词汇表上的向量。简单起见，我们先假设已经将本文切分完毕，存放到列表中，并对词汇向量进行分类标注。编写代码如下：

	Python
<pre>1 # -*- coding: UTF-8 -*- 2 3 """ 4 函数说明: 创建实验样本 5 6 Parameters: 7 无 8 Returns: 9 postingList - 实验样本切分的词条 10 classVec - 类别标签向量 11 Author: 12 Jack Cui 13 Blog: 14 http://blog.csdn.net/c406495762 15 Modify: 16 2017-08-11 17 """ 18 def loadDataSet(): 19 postingList=[['my', 'dog', 'has', 'flea', 'problems', 'help', 'please'], 20 ['maybe', 'not', 'take', 'him', 'to', 'dog', 'park', 'stupid'], 21 ['my', 'dalmation', 'is', 'so', 'cute', 'I', 'love', 'him'], 22 ['stop', 'posting', 'stupid', 'worthless', 'garbage'], 23 ['mr', 'licks', 'ate', 'my', 'steak', 'how', 'to', 'stop', 'him'], 24 ['quit', 'buying', 'worthless', 'dog', 'food', 'stupid']] 25 classVec = [0,1,0,1,0,1] 26 return postingList,classVec 27 28 if __name__ == '__main__': 29 postingList, classVec = loadDataSet() 30 for each in postingList: 31 print(each) 32 print(classVec)</pre>	#切分

从运行结果可以看出，我们已经将postingList是存放词条列表中，classVec是存放每个词条的所属类别，1代表侮辱类，0代表非侮辱类。

```

249
250 ▼ if __name__ == '__main__':
251     postingList, classVec = loadDataSet()
252     for each in postingList:
253         print(each)
254     print(classVec)
255
['my', 'dog', 'has', 'flea', 'problems', 'help', 'please']
['maybe', 'not', 'take', 'him', 'to', 'dog', 'park', 'stupid']
['my', 'dalmation', 'is', 'so', 'cute', 'I', 'love', 'him']
['stop', 'posting', 'stupid', 'worthless', 'garbage']
['mr', 'licks', 'ate', 'my', 'steak', 'how', 'to', 'stop', 'him']
['quit', 'buying', 'worthless', 'dog', 'food', 'stupid']
[0, 1, 0, 1, 0, 1]
[Finished in 0.5s]

```

继续编写代码，前面我们已经说过我们要先创建一个词汇表，并将切分好的词条转换为词条向量。

	Python
<pre> 1 # -*- coding: UTF-8 -*- 2 3 """ 4 函数说明:创建实验样本 5 6 Parameters: 7 无 8 Returns: 9 postingList - 实验样本切分的词条 10 classVec - 类别标签向量 11 Author: 12 Jack Cui 13 Blog: 14 http://blog.csdn.net/c406495762 15 Modify: 16 2017-08-11 17 """ 18 def loadDataSet(): 19 postingList=[['my', 'dog', 'has', 'flea', 'problems', 'help', 'please'], 20 ['maybe', 'not', 'take', 'him', 'to', 'dog', 'park', 'stupid'], 21 ['my', 'dalmation', 'is', 'so', 'cute', 'I', 'love', 'him'], 22 ['stop', 'posting', 'stupid', 'worthless', 'garbage'], 23 ['mr', 'licks', 'ate', 'my', 'steak', 'how', 'to', 'stop', 'him'], 24 ['quit', 'buying', 'worthless', 'dog', 'food', 'stupid']] 25 classVec = [0,1,0,1,0,1] 26 return postingList,classVec 27 28 """ 29 函数说明:根据vocabList词汇表，将inputSet向量化，向量的每个元素为1或0 30 31 Parameters: 32 vocabList - createVocabList返回的列表 33 inputSet - 切分的词条列表 34 Returns: 35 returnVec - 文档向量,词集模型 36 Author: 37 Jack Cui 38 Blog: 39 http://blog.csdn.net/c406495762 40 Modify: </pre>	<pre> #切 </pre>

52 函数说明:将切分的实验样本词条整理成不重复的词条列表,也就是词汇表

[illegible]

```
Python
1  # -*- coding: UTF-8 -*-
2  import numpy as np
3
4  """
5  函数说明:创建实验样本
6
7  Parameters:
8      无
9  Returns:
10     postingList - 实验样本切分的词条
```



```

11     classVec - 类别标签向量
12 Author:
13     Jack Cui
14 Blog:
15     http://blog.csdn.net/c406495762
16 Modify:
17     2017-08-11
18 """
19 def loadDataSet():
20     postingList=[['my', 'dog', 'has', 'flea', 'problems', 'help', 'please'],          #1
21                  ['maybe', 'not', 'take', 'him', 'to', 'dog', 'park', 'stupid'],
22                  ['my', 'dalmation', 'is', 'so', 'cute', 'I', 'love', 'him'],
23                  ['stop', 'posting', 'stupid', 'worthless', 'garbage'],
24                  ['mr', 'licks', 'ate', 'my', 'steak', 'how', 'to', 'stop', 'him'],
25                  ['quit', 'buying', 'worthless', 'dog', 'food', 'stupid']]
26     classVec = [0,1,0,1,0,1]
27     return postingList,classVec
28
29 """
30 函数说明:根据vocabList词汇表,将inputSet向量化,向量的每个元素为1或0
31
32 Parameters:
33     vocabList - createVocabList返回的列表
34     inputSet - 切分的词条列表
35 Returns:
36     returnVec - 文档向量,词集模型
37 Author:
38     Jack Cui
39 Blog:
40     http://blog.csdn.net/c406495762
41 Modify:
42     2017-08-11
43 """
44 def setOfWords2Vec(vocabList, inputSet):
45     returnVec = [0] * len(vocabList)
46     for word in inputSet:
47         if word in vocabList:
48             returnVec[vocabList.index(word)] = 1
49         else: print("the word: %s is not in my Vocabulary!" % word)
50     return returnVec
51
52 """
53 函数说明:将切分的实验样本词条整理成不重复的词条列表,也就是词汇表
54
55 Parameters:
56     dataSet - 整理的样本数据集
57 Returns:
58     vocabSet - 返回不重复的词条列表,也就是词汇表
59 Author:
60     Jack Cui
61 Blog:
62     http://blog.csdn.net/c406495762
63 Modify:
64     2017-08-11
65 """
66

```

运行结果如下, p0V存放的是每个单词属于类别0, 也就是非侮辱类词汇的概率。比如p0V的倒数第6个概率, 就是stupid这个单词属于非侮辱类的概率为0。同理, p1V的倒数第6个概率, 就是stupid这个单词属于侮辱类的概率为0.15789474, 也就是约等于15.79%的概率。我们知道stupid的中文意思是蠢货, 难听点的叫法就是傻逼。显而易见, 这个单词属于侮辱类。pAb是所有侮辱类的样本占有所有样本的概率, 从classVec中可以看出, 一共有3个侮辱类, 3个非侮辱类。所以侮辱类的概率是0.5。因此p0V存放的就是 $P(\text{him} | \text{非侮辱类}) = 0.0833$, $P(\text{is} | \text{非侮辱类}) = 0.0417$, 一直到 $P(\text{dog} | \text{非侮辱类})$


```

34     dataSet - 整理的样本数据集
35 Returns:
36     vocabSet - 返回不重复的词条列表，也就是词汇表
37 Author:
38     Jack Cui
39 Blog:
40     http://blog.csdn.net/c406495762
41 Modify:
42     2017-08-11
43 """
44 def createVocabList(dataSet):
45     vocabSet = set([]) #创建一个空的不重复列表
46     for document in dataSet:
47         vocabSet = vocabSet | set(document) #取并集
48     return list(vocabSet)
49 """
50 函数说明:根据vocabList词汇表，将inputSet向量化，向量的每个元素为1或0
51 Parameters:
52     vocabList - createVocabList返回的列表
53     inputSet - 切分的词条列表
54 Returns:
55     returnVec - 文档向量,词集模型
56 Author:
57     Jack Cui
58 Blog:
59     http://blog.csdn.net/c406495762
60 Modify:
61     2017-08-11
62 """
63 def setOfWords2Vec(vocabList, inputSet):
64     returnVec = [0] * len(vocabList) #创建一个其中所含元素都为0的向量

```

我们测试了两个词条，在使用分类器前，也需要对词条向量化，然后使用classifyNB()函数，用朴素贝叶斯公式，计算词条向量属于侮辱类和非侮辱类的概率。运行结果如下：

```

171 if __name__ == '__main__':
172     testingNB()

p0: 0.0
p1: 0.0
['love', 'my', 'dalmation'] 属于非侮辱类
p0: 0.0
p1: 0.0
['stupid', 'garbage'] 属于非侮辱类
[Finished in 0.3s]

```

你会发现，这样写的算法无法进行分类，p0和p1的计算结果都是0，显然结果错误。这是为什么呢？下一篇文章继续讲解~

四、总结

朴素贝叶斯推断的一些优点：

- 生成式模型，通过计算概率来进行分类，可以用来处理多分类问题。
- 对小规模的数据表现很好，适合多分类任务，适合增量式训练，算法也比较简单。

朴素贝叶斯推断的一些缺点：

- 对输入数据的表达形式很敏感。
- 由于朴素贝叶斯的“朴素”特点，所以会带来一些准确率上的损失。
- 需要计算先验概率，分类决策存在错误率。

其它：

- 本文中的编程实例，存在一定的问题，需要进行改进，下篇文章会讲解改进方法；
- 同时，本文中的编程实例，没有进行前期的文本切分，下一篇文章会讲解英文单词和中文单词的切分方法；
- 下篇文章将使用sklearn进行中文实例练习；
- 朴素贝叶斯的准确率，其实是比较依赖于训练语料的，机器学习算法就和纯洁的小孩一样，取决于其成长（训练）条件，“吃的是草挤的是奶”，但“不是所有的牛奶，都叫特仑苏”。
- 参考文献：http://www.ruanyifeng.com/blog/2013/12/naive_bayes_classifier.html
- 如有问题，请留言。如有错误，还望指正，谢谢！

PS：如果觉得本篇本章对您有所帮助，欢迎关注、评论、赞！

本文出现的所有代码和数据集，均可在我的github上下载，欢迎Follow、Star：

<https://github.com/Jack-Cherish/Machine-Learning>



微信公众号

分享技术，乐享生活：微信公众号搜索「JackCui-AI」关注一个在互联网摸爬滚打的潜行者。

黑夜无论怎样悠长，白昼总会到来。--- 莎士比亚