

21 | 朴素贝叶斯分类（下）：如何对文档进行分类？

2019-01-30 陈旻



讲述：陈旻

时长 14:22 大小 13.17M



我们上一节讲了朴素贝叶斯的工作原理，今天我们来讲下这些原理是如何指导实际业务的。

朴素贝叶斯分类最适合的场景就是文本分类、情感分析和垃圾邮件识别。其中情感分析和垃圾邮件识别都是通过文本来进行判断。从这里你能看出来，这三个场景本质上都是文本分类，这也是朴素贝叶斯最擅长的地方。所以朴素贝叶斯也常用于自然语言处理 NLP 的工具。

今天我带你一起使用朴素贝叶斯做下文档分类的项目，最重要的工具就是 sklearn 这个机器学习神器。

sklearn 机器学习包

sklearn 的全名叫 Scikit-learn，它给我们提供了 3 个朴素贝叶斯分类算法，分别是高斯朴素贝叶斯（GaussianNB）、多项式朴素贝叶斯（MultinomialNB）和伯努利朴素贝叶斯（BernoulliNB）。

这三种算法适合应用在不同的场景下，我们应该根据特征变量的不同选择不同的算法：

高斯朴素贝叶斯：特征变量是连续变量，符合高斯分布，比如说人的身高，物体的长度。

多项式朴素贝叶斯：特征变量是离散变量，符合多项分布，在文档分类中特征变量体现在一个单词出现的次数，或者是单词的 TF-IDF 值等。

伯努利朴素贝叶斯：特征变量是布尔变量，符合 0/1 分布，在文档分类中特征是单词是否出现。

伯努利朴素贝叶斯是以文件为粒度，如果该单词在某文件中出现了即为 1，否则为 0。而多项式朴素贝叶斯是以单词为粒度，会计算在某个文件中的具体次数。而高斯朴素贝叶斯适合处理特征变量是连续变量，且符合正态分布（高斯分布）的情况。比如身高、体重这种自然界的现象就比较适合用高斯朴素贝叶斯来处理。而文本分类是使用多项式朴素贝叶斯或者伯努利朴素贝叶斯。

什么是 TF-IDF 值呢？

我在多项式朴素贝叶斯中提到了“词的 TF-IDF 值”，如何理解这个概念呢？

TF-IDF 是一个统计方法，用来评估某个词语对于一个文件集或文档库中的其中一份文件的重要程度。

TF-IDF 实际上是两个词组 Term Frequency 和 Inverse Document Frequency 的总称，两者缩写为 TF 和 IDF，分别代表了词频和逆向文档频率。

词频 TF计算了一个单词在文档中出现的次数，它认为一个单词的重要性和它在文档中出现的次数呈正比。

逆向文档频率 IDF，是指一个单词在文档中的区分度。它认为一个单词出现在的文档数越少，就越能通过这个单词把该文档和其他文档区分开。IDF 越大就代表该单词的区分度越

大。

所以 TF-IDF 实际上是词频 TF 和逆向文档频率 IDF 的乘积。这样我们倾向于找到 TF 和 IDF 取值都高的单词作为区分，即这个单词在一个文档中出现的次数多，同时又很少出现在其他文档中。这样的单词适合用于分类。

TF-IDF 如何计算

首先我们看下词频 TF 和逆向文档概率 IDF 的公式。

$$\text{词频 TF} = \frac{\text{单词出现的次数}}{\text{该文档的总单词数}}$$

$$\text{逆向文档频率 IDF} = \log \frac{\text{文档总数}}{\text{该单词出现的文档数} + 1}$$

为什么 IDF 的分母中，单词出现的文档数要加 1 呢？因为有些单词可能不会存在文档中，为了避免分母为 0，统一给单词出现的文档数都加 1。

TF-IDF=TF*IDF。

你可以看到，TF-IDF 值就是 TF 与 IDF 的乘积，这样可以更准确地对文档进行分类。比如“我”这样的高频单词，虽然 TF 词频高，但是 IDF 值很低，整体的 TF-IDF 也不高。

我在这里举个例子。假设一个文件夹里一共有 10 篇文档，其中一篇文档有 1000 个单词，“this”这个单词出现 20 次，“bayes”出现了 5 次。“this”在所有文档中均出现过，而“bayes”只在 2 篇文档中出现过。我们来计算一下这两个词语的 TF-IDF 值。

针对“this”，计算 TF-IDF 值：

$$\text{词频 TF} = \frac{20}{1000} = 0.02$$

$$\text{逆向文档频率 IDF} = \log \frac{10}{10 + 1} = -0.0414$$

所以 $\text{TF-IDF} = 0.02 * (-0.0414) = -8.28e-4$ 。

针对 “bayes” ，计算 TF-IDF 值：

$$\text{词频 TF} = \frac{5}{1000} = 0.005$$

$$\text{逆向文档频率 IDF} = \log \frac{10}{2 + 1} = 0.5229$$

$\text{TF-IDF} = 0.005 * 0.5229 = 2.61e-3$ 。

很明显 “bayes” 的 TF-IDF 值要大于 “this” 的 TF-IDF 值。这就说明用 “bayes” 这个单词做区分比单词 “this” 要好。


如何求 TF-IDF

在 sklearn 中我们直接使用 TfidfVectorizer 类，它可以帮我们计算单词 TF-IDF 向量的值。在这个类中，取 sklearn 计算的对数 log 时，底数是 e，不是 10。

下面我来讲下如何创建 TfidfVectorizer 类。

TfidfVectorizer 类的创建：

创建 TfidfVectorizer 的方法是：

 复制代码

```
1 TfidfVectorizer(stop_words=stop_words, token_pattern=token_pattern)
```

我们在创建的时候，有两个构造参数，可以自定义停用词 stop_words 和规律规则 token_pattern。需要注意的是传递的数据结构，停用词 stop_words 是一个列表 List 类

型，而过滤规则 token_pattern 是正则表达式。

什么是停用词？停用词就是在分类中没有用的词，这些词一般词频 TF 高，但是 IDF 很低，起不到分类的作用。为了节省空间和计算时间，我们把这些词作为停用词 stop words，告诉机器这些词不需要帮我计算。

参数表	作用
stop_words	自定义停用词表，为列表List类型
token_pattern	过滤规则，正则表达式，如r"(?u)\b\w+\b"

当我们创建好 TF-IDF 向量类型时，可以用 fit_transform 帮我们计算，返回给我们文本矩阵，该矩阵表示了每个单词在每个文档中的 TF-IDF 值。

方法表	作用
fit_transform(X)	拟合模型，并返回文本矩阵

在我们进行 fit_transform 拟合模型后，我们可以得到更多的 TF-IDF 向量属性，比如，我们可以得到词汇的对应关系（字典类型）和向量的 IDF 值，当然也可以获取设置的停用词 stop_words。

属性表	作用
vocabulary_	词汇表；字典类型
idf_	返回idf值
stop_words_	返回停用词表

举个例子，假设我们有 4 个文档：

文档 1：this is the bayes document；


文档 2：this is the second second document；

文档 3：and the third one；

文档 4：is this the document.


现在想要计算文档里都有哪些单词，这些单词在不同文档中的 TF-IDF 值是多少呢？

首先我们创建 TfidfVectorizer 类：

 复制代码


```
1 from sklearn.feature_extraction.text import TfidfVectorizer
2 tfidf_vec = TfidfVectorizer()
```

然后我们创建 4 个文档的列表 documents，并让创建好的 tfidf_vec 对 documents 进行拟合，得到 TF-IDF 矩阵：

 复制代码


```
1 documents = [
2     'this is the bayes document',
3     'this is the second second document',
4     'and the third one',
5     'is this the document'
6 ]
7 tfidf_matrix = tfidf_vec.fit_transform(documents)
```

输出文档中所有不重复的词：

 复制代码

```
1 print('不重复的词:', tfidf_vec.get_feature_names())
```


运行结果

 复制代码

```
1 不重复的词: ['and', 'bayes', 'document', 'is', 'one', 'second', 'the', 'third', 'this']
```




输出每个单词对应的 id 值：

 复制代码

```
1 print('每个单词的 ID:', tfidf_vec.vocabulary_)
```


运行结果

 复制代码

```
1 每个单词的 ID: {'this': 8, 'is': 3, 'the': 6, 'bayes': 1, 'document': 2, 'second': 5, 'a
```




输出每个单词在每个文档中的 TF-IDF 值，向量里的顺序是按照词语的 id 顺序来的：

 复制代码

```
1 print('每个单词的 tfidf 值:', tfidf_matrix.toarray())
```

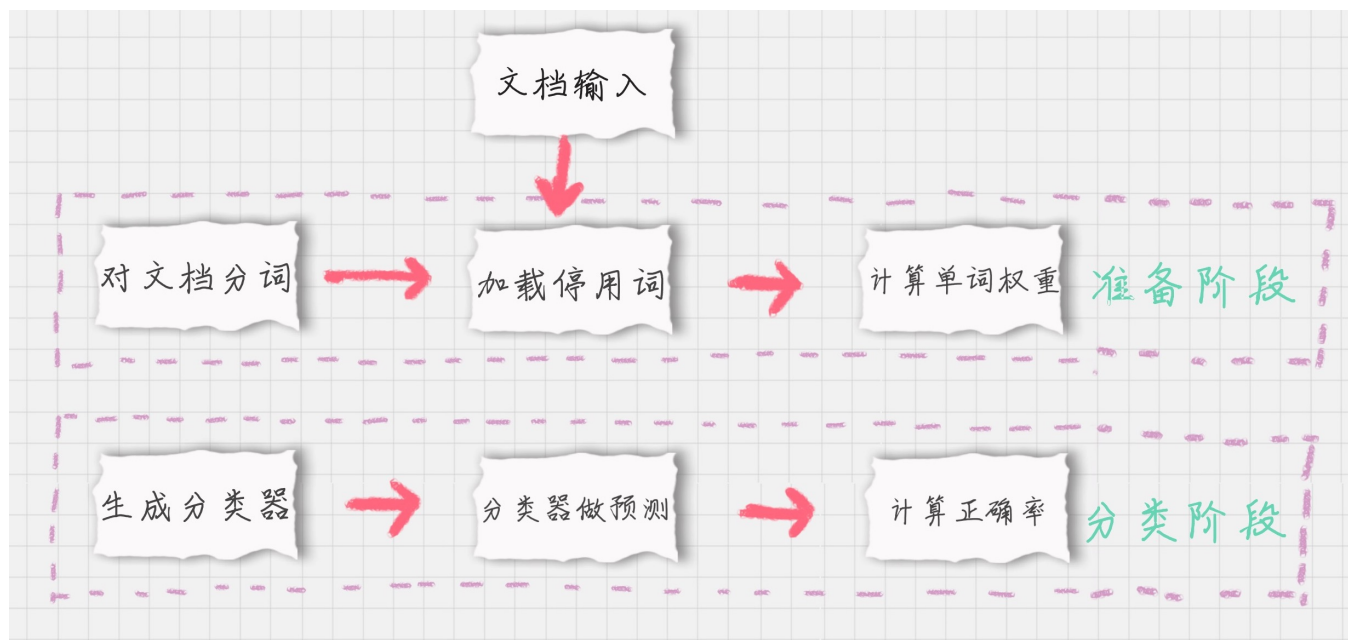
运行结果：

 复制代码

```
1 每个单词的 tfidf 值: [[0.          0.63314609 0.40412895 0.40412895 0.          0.
2   0.33040189 0.          0.40412895]
3  [0.          0.          0.27230147 0.27230147 0.          0.85322574
4   0.22262429 0.          0.27230147]
5  [0.55280532 0.          0.          0.          0.55280532 0.
6   0.28847675 0.55280532 0.          ]
7  [0.          0.          0.52210862 0.52210862 0.          0.
8   0.42685801 0.          0.52210862]]
```

如何对文档进行分类

如果我们要对文档进行分类，有两个重要的阶段：




1. **基于分词的数据准备**，包括分词、单词权重计算、去掉停用词；
2. **应用朴素贝叶斯分类进行分类**，首先通过训练集得到朴素贝叶斯分类器，然后将分类器应用于测试集，并与实际结果做对比，最终得到测试集的分类准确率。

下面，我分别对这些模块进行介绍。

模块 1：对文档进行分词


在准备阶段里，最重要的就是分词。那么如果给文档进行分词呢？英文文档和中文文档所使用的分词工具不同。

在英文文档中，最常用的是 NLTK 包。NLTK 包中包含了英文的停用词 stop words、分词和标注方法。

 复制代码

```
1 import nltk
2 word_list = nltk.word_tokenize(text) # 分词
3 nltk.pos_tag(word_list) # 标注单词的词性
```


在中文文档中，最常用的是 jieba 包。jieba 包中包含了中文的停用词 stop words 和分词方法。

 复制代码


```
1 import jieba
2 word_list = jieba.cut(text) # 中文分词
```

模块 2：加载停用词表

我们需要自己读取停用词表文件，从网上可以找到中文常用的停用词保存在 stop_words.txt，然后利用 Python 的文件读取函数读取文件，保存在 stop_words 数组中。


 复制代码

```
1 stop_words = [line.strip().decode('utf-8') for line in io.open('stop_words.txt').readlin
```

模块 3：计算单词的权重

这里我们用到 sklearn 里的 TfidfVectorizer 类，上面我们介绍过它使用的方法。

直接创建 TfidfVectorizer 类，然后使用 fit_transform 方法进行拟合，得到 TF-IDF 特征空间 features，你可以理解为选出来的分词就是特征。我们计算这些特征在文档上的特征向量，得到特征空间 features。

 复制代码

```
1 tf = TfidfVectorizer(stop_words=stop_words, max_df=0.5)
2 features = tf.fit_transform(train_contents)
```

这里 max_df 参数用来描述单词在文档中的最高出现率。假设 max_df=0.5，代表一个单词在 50% 的文档中都出现过了，那么它只携带了非常少的信息，因此就不作为分词统计。

一般很少设置 min_df，因为 min_df 通常都会很小。


模块 4：生成朴素贝叶斯分类器

我们将特征训练集的特征空间 `train_features`，以及训练集对应的分类 `train_labels` 传递给贝叶斯分类器 `clf`，它会自动生成一个符合特征空间和对应分类的分类器。

这里我们采用的是多项式贝叶斯分类器，其中 `alpha` 为平滑参数。为什么要使用平滑呢？因为如果一个单词在训练样本中没有出现，这个单词的概率就会被计算为 0。但训练集样本只是整体的抽样情况，我们不能因为一个事件没有观察到，就认为整个事件的概率为 0。为了解决这个问题，我们需要做平滑处理。

当 `alpha=1` 时，使用的是 Laplace 平滑。Laplace 平滑就是采用加 1 的方式，来统计没有出现过的单词的概率。这样当训练样本很大的时候，加 1 得到的概率变化可以忽略不计，也同时避免了零概率的问题。

当 $0 < \alpha < 1$ 时，使用的是 Lidstone 平滑。对于 Lidstone 平滑来说，`alpha` 越小，迭代次数越多，精度越高。我们可以设置 `alpha` 为 0.001。


 复制代码

```
1 # 多项式贝叶斯分类器
2 from sklearn.naive_bayes import MultinomialNB
3 clf = MultinomialNB(alpha=0.001).fit(train_features, train_labels)
```

模块 5：使用生成的分类器做预测

首先我们需要得到测试集的特征矩阵。


方法是用训练集的分词创建一个 `TfidfVectorizer` 类，使用同样的 `stop_words` 和 `max_df`，然后用这个 `TfidfVectorizer` 类对测试集的内容进行 `fit_transform` 拟合，得到测试集的特征矩阵 `test_features`。

 复制代码

```
1 test_tf = TfidfVectorizer(stop_words=stop_words, max_df=0.5, vocabulary=train_vocabulary)
2 test_features=test_tf.fit_transform(test_contents)
```

然后用训练好的分类器对新数据做预测。

方法是使用 `predict` 函数，传入测试集的特征矩阵 `test_features`，得到分类结果 `predicted_labels`。`predict` 函数做的工作就是求解所有后验概率并找出最大的那个。


 复制代码

```
1 predicted_labels=clf.predict(test_features)
```

模块 6：计算准确率

计算准确率实际上是对分类模型的评估。我们可以调用 `sklearn` 中的 `metrics` 包，在 `metrics` 中提供了 `accuracy_score` 函数，方便我们对实际结果和预测的结果做对比，给出模型的准确率。

使用方法如下：

 复制代码

```
1 from sklearn import metrics
2 print metrics.accuracy_score(test_labels, predicted_labels)
```

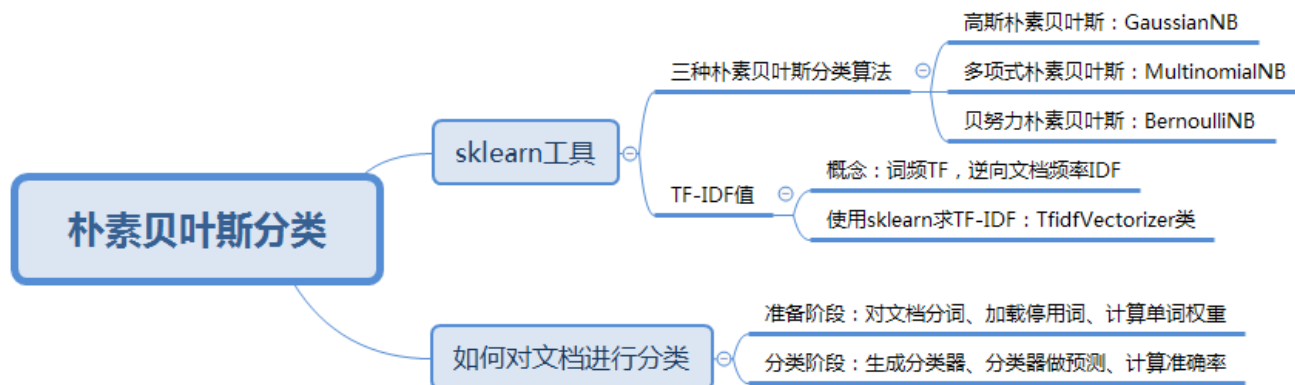
数据挖掘神器 `sklearn`

从数据挖掘的流程来看，一般包括了获取数据、数据清洗、模型训练、模型评估和模型部署这几个过程。

`sklearn` 中包含了大量的数据挖掘算法，比如三种朴素贝叶斯算法，我们只需要了解不同算法的适用条件，以及创建时所需的参数，就可以用模型帮我们进行训练。在模型评估中，`sklearn` 提供了 `metrics` 包，帮我们对预测结果与实际结果进行评估。

在文档分类的项目中，我们针对文档的特点，给出了基于分词的准备流程。一般来说 `NLTK` 包适用于英文文档，而 `jieba` 适用于中文文档。我们可以根据文档选择不同的包，对文档提取分词。这些分词就是贝叶斯分类中最重要的特征属性。基于这些分词，我们得到分词的权重，即特征矩阵。

通过特征矩阵与分类结果，我们就可以创建出朴素贝叶斯分类器，然后用分类器进行预测，最后预测结果与实际结果做对比即可以得到分类器在测试集上的准确率。



练习题

我已经讲了中文文档分类中的 6 个关键的模块，最后，我给你留一道对中文文档分类的练习题吧。

我将中文文档数据集上传到了 GitHub 上，[点击这里下载](#)。

数据说明：

1. 文档共有 4 种类型：女性、体育、文学、校园；

名称	修改日期	类型	大小
女性	2018/12/6 15:16	文件夹	
体育	2018/12/6 15:16	文件夹	
文学	2018/12/6 15:16	文件夹	
校园	2018/12/6 15:16	文件夹	

1. 训练集放到 train 文件夹里，测试集放到 test 文件夹里，停用词放到 stop 文件夹里。

名称	修改日期	类型	大小
stop	2018/12/6 15:16	文件夹	
test	2018/12/6 15:16	文件夹	
train	2018/12/6 15:16	文件夹	

请使用朴素贝叶斯分类对训练集进行训练，并对测试集进行验证，并给出测试集的准确率。

最后你不妨思考一下，假设我们要判断一个人的性别，是通过身高、体重、鞋码、外貌等属性进行判断的，如果我们用朴素贝叶斯做分类，适合使用哪种朴素贝叶斯分类器？停用

词的作用又是什么？

欢迎你在评论区进行留言，与我分享你的答案。也欢迎点击“请朋友读”，把这篇文章分享给你的朋友或者同事。

 极客时间

数据分析实战 45 讲

即学即用的数据分析入门课



陈旻
清华大学计算机博士

新版升级：点击「 请朋友读」，10位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得转载

上一篇 20 | 朴素贝叶斯分类（上）：如何让机器判断男女？

下一篇 22 | SVM（上）：如何用一根棍子将蓝红两色球分开？

精选留言 (24)

 写留言



szm

2019-01-30

 14

需要完整代码，不然看不明白！

展开 ∨



Python

 8

2019-01-30

老师，能不能在答疑的时候给这道题的完整代码看看

展开 ∨



姜戈

2019-01-30

👍 5

看过很多朴素贝叶斯原理和分类的讲解文章，很少能像前辈这样既有理论，又有实战的讲解，让大家既了解了理论知识，又有相应实际的操作经验可学，真的好棒，这个专栏，必须多多点赞，为老师加油！！

展开 ∨



北方

2019-02-14

👍 4

```
#!/usr/bin/env python
# -*- coding:utf8 -*-
# __author__ = '北方姆Q'
# __datetime__ = 2019/2/14 14:04
```

...

展开 ∨



池边的树

2019-02-12

👍 2

<https://github.com/yourSprite/AnalysisExcercise/tree/master/%E6%9C%B4%E7%B4>

展开 ∨



上官

2019-01-31

👍 2

```
print('不重复的词:', tfidf_vec.get_feature_names())
```

运行结果：不重复的词: ['and', 'bayes', 'document', 'is', 'one', 'second', 'the', 'third', 'this']

这明明就是打印所有词啊，有重复的啊

展开 ∨



Jack

2019-02-14

👍 1

```
#!/usr/bin/env python
```

```
# coding: utf-8
```

```
import os
```

```
import jieba...
```

展开 ∨



wzhan366

2019-02-06

👍 1

建议 大家先做英文版本，因为中文的unicode encode和decode不是很好弄，不利于中间步骤的可视化。如果对代码有疑惑，可以试试这个pipeline，sklearn 的。不过，这个没有用NLTK。

展开 ∨



王小王

2019-02-02

👍 1

能不能讲解下本堂课的练习题？

展开 ∨



Rickie

2019-01-30

👍 1

老师，token_pattern里的正则中 (?u)是什么意思呀？

展开 ∨



滨滨

2019-03-02

👍

适合用高斯朴素贝叶斯。因为身高、体重、鞋码是连续变量。

停用词的作用是去除无义词，减少重复计算。



飞Lisa

2019-03-02

👍

我觉得老师你讲的，太好了！以前看机器学习总是看不进去，看了你的讲解真的让我提起了兴趣，一下子看了十几篇。然后请问一下老师，要继续锻炼机器学习的实战能力的话你推荐什么书或者课程或者练习项目？

展开 ∨



王彬成

2019-02-21



一、最后你不妨思考一下，假设我们要判断一个人的性别，是通过身高、体重、鞋码、外貌等属性进行判断的，如果我们用朴素贝叶斯做分类，适合使用哪种朴素贝叶斯分类器？停用词的作用又是什么？

适合用高斯朴素贝叶斯。因为身高、体重、鞋码是连续变量。...

展开 ∨



一语中的

2019-02-18



前后看了3遍，我终于理解了，也可以自己敲出结果了！

展开 ∨



Python

2019-02-13



身高体重等等这些适合用高斯分类器，停用词的作用是减少重复计算

展开 ∨



乔巴

2019-02-13



#code:utf-8

```
import pandas as pd
```

```
import numpy as np
```

```
import io
```

```
import os...
```

展开 ∨



乔巴

2019-02-13



但准确度不高，不知怎么优化

```
# -*- coding: utf-8 -*-
```



```
#coding:utf-8
import os
import jieba ...
```

展开 ▾



乔巴

2019-02-13



TfidfVectorizer获取特征值运行异常，但CountVectorizer可以，

```
import pandas as pd
import numpy as np
import io
from sklearn.feature_extraction.text import TfidfVectorizer...
```

展开 ▾



无才不肖生

2019-02-07



首先，训练集和测试集中部分文件无法读取，我是直接跳过没加载，感觉少了点不大影响。

其次，我最后的预测结果都是女性，正确率只有百分之25

最后，我想问下，特征矩阵是4*20705这个对吗

展开 ▾



雨先生的晴...

2019-02-03



还是希望老师可以在github分享一下代码，练习题还是没有办法解

展开 ▾