朴素贝叶斯分类

什么是朴素贝叶斯分类

朴素贝叶斯(Naive Bayes)法是基于贝叶斯定理与特征条件假设进行的分类方法。对于给定的数据集，统计出相应数据的分布，根据贝叶斯定理，进行类别推断。

从概率开始

在机器学习领域的一个关键概念是不确定性的概念。它可以由测量的误差引起，也可以由数据集的有限大小引起。概率论提供了一个合理的框架，用来对不确定性进行量化和计算。概率论还构成了机器学习的一个中心基础，让我们能够根据所有能得到的信息做出最优的预测，即使信息可能是不完全的或者是含糊的。

一个例子

有红色和蓝色两个不同颜色的盒子，里面分别装有两种不同的水果：苹果和橙子。盒子中装有水果的情况如下表所示:

	红盒子	**蓝盒子**
苹果	2	3
橙子	6	1

传统概率论是基于对实验结果观察的统计，想象红盒子和蓝盒子在一个黑暗的房间中，一共有10个盒子，其中红盒子**4**个，蓝盒子**6**个

传统概率基于计数

我们可以根据分类和总数的比例算出以下概率:

$$P(\text{选中红盒子的概率}) = P(B = r) = \frac{4}{10}$$
$$P(\text{选中蓝盒子的概率}) = P(B = b) = \frac{6}{10}$$

其中我们称**B(取到盒子的颜色)**为随机变量，随机变量是表示随机试验各种结果的实值单值函数，这里**B**有两个取值**r(红色)和b(蓝色)**。相应取值的数量和总量的比值便是取到该颜色盒子的**概率**，这也是传统概率思想的朴素实践。如果已知盒子是红盒子，从红盒子中取到苹果和橙子的概率分别是多少，可以如下计算：

$$P(\text{红盒子中取出苹果的概率}) = P(F=a|B=r) = \frac{2}{8}$$
$$P(\text{红盒子中取出橙子的概率}) = P(F=o|B=r) = \frac{6}{8}$$

对概率加上限定条件，我们称为**条件概率**。如果问题变为当实验结果是红盒子且取出的是苹果，如何计算?

$$P(\text{红盒子中取出的是苹果}) = P(B=r, F=a) = P(F=a|B=r)P(B=r) = \frac{2}{8} * \frac{4}{10} = \frac{2}{20}$$

上述计算是条件概率公式$P(A|B) = \frac{P(AB)}{P(B)}$ 的一个变形，也是基于计数统计的很朴素的理解。特别的，当A发生的概率不依赖于B的情况下，我们称事件A，B相互独立即:$P(A|B) = P(A)$，条件概率公式可变形为:$P(AB) = P(A)P(B)$

上述公式是我们学习概率的最基本公式。

全概率公式

如果我们只关心取出苹果的概率，该如何计算？因为红盒子和蓝盒子中均有苹果，必须把所有含有苹果的情况都考虑进去，这个计算过程就是全概率公式。

$$\begin{aligned}
P(\text{取出苹果的概率}) =& P(\text{红盒子中取出苹果的概率}) + P(\text{蓝盒子中取出苹果的概率}) \\
=& P(B=r, F=a) + P(B=b, F=a) \\
=& P(F=a|B=r)P(B=r) + P(F=a|B=b)P(B=b) \\
=& \frac{2}{8} * \frac{4}{10} + \frac{3}{4} * \frac{6}{10} \\
=& \frac{11}{20}
\end{aligned}$$

全概率公式的一般形式如下所示:

$$P(A) = \sum_{i}^{n} P(A|B_i)P(B_i)$$
$$\text{其中} B_i \text{满足} \sum_{i}^{n} P(B_i) = 1 \text{ 且 } B_1 \cap B_2 \cap B_3 \cdots \cap B_n = \varnothing$$

贝叶斯推断

想象一个场景，你在玩一个猜箱子的游戏，游戏是这样的:你看不到箱子的颜色，从箱子里随便一抓，抓出来一个苹果，请问箱子的颜色是什么？实际需要计算的是，当取出苹果的情况下，箱子是蓝箱子或红箱子的概率，选择概率大的胜算更大。

$$P(B=r|F=a) = \frac{P(B=r, F=a)}{P(F=a)} = \frac{P(F=a|B=r)P(B=r)}{P(F=a)} = \frac{\frac{2}{20}}{\frac{11}{20}} = \frac{2}{11}$$

我们要求$P(B=r|F=a)$ 如果取到苹果那么箱子是红箱子的可能性，这个概率对应于我们日常生活中的一些推导，没办法直接求出，但是$P(F=a|B=r)$是可以求出的，所以我们借助于条件概率公式的变形$P(AB) = P(A|B)P(B) = P(B|A)P(A)$ 将其变形成上面的过程求解 。我们将$P(B=r|F=a)$称为**后验概率**,$P(F=a|B=r)$称为**似然函数**(可能性函数)，这是符合我们现实生活中直觉的(为什么?)。$P(B=r)$称为**先验经验**。这种使用似然函数及先验经验推导出后验概率的过程称

为**贝叶斯推断**

使用贝叶斯分类进行文档类别划分

文本分类在自然语言处理中有广泛的应用，如何将海量文本自动归档归类，是机器学习的一个主要应用方向。常见的垃圾邮件处理、评论情感分析、文章主题归类都属于这个范畴。贝叶斯分类器是解决这类问题的一种简单有效的方式。下面我们介绍文章主题归类，垃圾邮件处理和评论情感分析的原理类似

文章主题归类

假如我们有如下语料库:

类	编号	内容	类别
训练集	1	Chinese Beijing Chinese	zh
	2	Chinese Chinese Shanghai	zh
	3	Chinese Macao	zh
	4	Tokyo Japan Chinese	jp
测试集	5	Chinese Chinese Chinese Tokyo Japan	?

训练集是我们人为标记好的数据，是已知数据，测试集是未分类数据。如何根据训练集的已知数据预测测试集新文章的类别就是文章主题分类问题。

贝叶斯建模

语料库有很多文章构成，记为$d_1, d_2, d_3, \cdots, d_n$。每篇文章有若干词构成，记为$d_i = (w_1, w_2, w_3, \cdots, w_n)$。文章的类别，记为c，这里c有两个取值:zh(中国)或jp(日本)。我们的任务是计算第5篇文档(d_5)属于zh和jp的概率，以概率大的类别作为文档5的类别。利用贝叶斯公式得到下面的推导。

$$P(c = zh|d_5) = \frac{P(d_5|c = zh)P(c = zh)}{p(d_5)}$$

对于不同列别概率的比较分母是相同的，上式可记为：

$$P(c = zh|d_5) \propto P(d_5|c = zh)P(c = zh)$$

即：我们仅比较似然函数和证据即可

我们假设文章中出现的词都是条件独立的:

$$P(d_5|c = zh) = P((w_1, w_2, \cdots, w_n)|c = zh) = P(w_1|c = zh)P(w_2|c = zh)\dots P(w_n|c = zh) = \prod_i^n P(w_i|c)$$

我们只需计算出每一个词在相应类别中出现的概率，相乘即可得到该类别产生此文章的概率。

统计语料库计算模型概率参数

下面我们来计算上述贝叶斯模型中的各种参数。首先计算先验经验:

在门当前的语料库中一共有四篇文档，其中类别属于zh的文档有3篇，类别属于jp的文档有1篇，所以：

$$P(c=zh)=\frac{3}{4}$$

$$p(c=jp)=\frac{1}{4}$$

为了表示方便我们将$P(c=zh)$表示为$P(c)$,将$P(c=jp)$表示为$P(\bar{c})$

然后计算似然函数:

对于文档5里面出现了三种词：$Chinese$、$Tokyo$、$Japan$,我们计算他们在类别zh中的条件概率：

$$P(Tokyo|c)=\frac{\text{训练集}zh\text{类别}Tokyo\text{出现的次数}}{\text{训练集}zh\text{类别中词的总数}}=\frac{0}{8}$$

这里为我们碰到一个麻烦，如果某词在语料库的某个类别中没有出现，那么会出现概率为0的情况，不方便比较：

所以我们引入拉普拉斯平滑，将分子加上λ(λ称为平滑因子，此处取1)，分母加上不重复的词的个数,公式变为：

$$P(Tokyo|c)=\frac{\text{训练集}zh\text{类别}Tokyo\text{出现的次数}+\lambda}{\text{训练集}zh\text{类别中词的总数}+\text{词汇量}}=\frac{0+1}{8+6}$$

按照上述方法计算各个词出现的条件概率:

对于类别zh：

$$P(Chinese|c)=\frac{5+1}{8+6}=\frac{3}{7}$$

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

对于类别jp：

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

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

得到了先验经验和似然函数，我们计算文档5分别属于每个类别的概率:

计算文档5属于zh的概率：

$$P(c|d_5)\propto P(Chinese|c)*P(Cinese|c)*P(Chinese|c)*P(Tokyo|c)*P(Japan|c)*P(c)=0.0003$$

计算文档5属于jp的概率：

$$P(\bar{c}|d_5)\propto P(Chinese|\bar{c})^3*P(Tokyo|\bar{c})*P(Japan|\bar{c})*P(\bar{c})=0.0001$$

由于$P(c|d_5)>P(\bar{c}|d_5)$所以文档5被划分为类别zh。

如何处理中文

上一小节的例子是对英文做的统计，大家发现英文有一个特点每一个单词天然使用空格隔开。我们可以很容易统计每一个单词出现的次数。中文的情况怎么办？看下面的例子。

```
对"南京市长江大桥"这句话进行单词提取
结果是:
南京　市长　江大桥
还是:
南京市　长江大桥
```

将一个完整的汉语句子切分成单词的形式就是**中文分词**。中文分词有一套完整的理论体系，这里我们不赘述原理，仅仅阐述中文分词在python中的实现。

使用结巴分词

基本使用

首先安装结巴分词依赖包

```
pip install jieba
```

导入包，使用cut进行分词

```
import jieba
content = "床前明月光,我要学python"
#使用cut方法进行分词，返回结果是一个生成器
result = jieba.cut(content)
#使用循环打印生成结果
# for w in result:
#     print(w)
#要生成上一小节使用空格隔开英文形式的字符串还有更简洁的方式
print(" ".join(result))
#输出结果是:床前 明月光 ， 我 要学 python
```

分词模式

结巴分词cut函数的第二个参数是cut_all,设置为True是全模式，False是精确模式

1. 精确模式：根据内置词典，将句子最精确地切开，适合文本分析，生成较少的分词结果；
2. 全模式：尽可能切分出句子中所有可能出现的词

```
import jieba

#全模式
text = "今天天气真好"
result = jieba.cut(text, cut_all=True)
print("全模式: ", " ".join(result))
#全模式结果:  今天 今天天气 天天 天气 真好

#精确模式
result = jieba.cut(text, cut_all=False)
print("精确模式: ", " ".join(result))
#精确模式结果:  今天天气 真 好
```

结巴分词还有一个搜索引擎模式，能对长句进行更准确的切分，适合搜索引擎进行搜索

```
import jieba
text = "今天天气真好"
# 搜索引擎模式
result = jieba.cut_for_search(text)
print("搜索引擎模式: ", " ".join(result))
# 搜索引擎模式:  今天 天天 天气 今天天气 真 好
```

使用自定义词典

有时我们需要改变分词器的默认行为，如:后台开发，我们希望分词器把它作为一个词来处理，这时需要借助自定义词典了。按照如下格式创建词典:

```
一个词语占一行(分三部分)
格式: 词语 词频 词性(词性一般可以省略)
```

按照上述格式创建userdict.txt文件:

```
专业技能 100
后台开发 100
```

使用自定义词典

```
import jieba
content = "招聘具有后台开发专业技能的人员"
#加载自定义词典
jieba.load_userdict("userdict.txt")
result = jieba.cut(content)
print(" ".join(result))
#自定义词典结果是:招聘 具有 后台开发 专业技能 的 人员
#如果不使用自定词典结果是:招聘 具有 后台 开发 专业技能 的 人员
```

词向量化

大家注意一个问题，无论是引文还是中文，切词划分以后计算机是无法理解这些字符意思的，我们必须找到一种衡量标准，使计算机能够站在数的角度理解词的意思。将词转换为计算机能理解的向量形式称为**词向量化**

TF-ID

TF-IDF（term frequency–inverse document frequency）是一种用于信息检索与数据挖掘的常用加权技术。TF意思是词频(Term Frequency)，IDF意思是逆文本频率指数(Inverse Document Frequency)。

有很多不同的数学公式可以用来计算TF-IDF。词频 (TF) 是一词语出现的次数除以该文件的总词语数。假如一篇文件的总词语数是100个，而词语“计算机”出现了3次，那么“计算机”一词在该文件中的词频就是3/100=0.03。一个计算文件频率 (IDF) 的方法是文件集里包含的文件总数除以测定有多少份文件出现过“计算机”一词。所以，如果“计算机”一词在1,000份文件出现过，而文件总数是10,000,000份的话，其逆向文件频率就是 lg(10,000,000 / 1,000)=4。最后的TF-IDF的分数为0.03 * 4=0.12

使用公式表示如下:

$$tf_{w_i} = \frac{n_{w_i}}{\sum_{i=0}^{k} n_{w_i}}$$

其中n_{w_i}表示第i个词出现的次数，分母表示一篇文章有k个词，每个词出现次数的和

$$idf_i = lg\frac{|D|}{|\{j : w_i \in d_j\}|}$$

其中lg表示10为低的对数，$|D|$表示文章总数，分母表示含有词w_i的文章的个数

TF-ID向量化

将句子中每一个出现的词使用TF-IDF数值取代，组合成一维数组就是TF-IDF向量化。

```
import jieba
from sklearn.feature_extraction.text import TfidfVectorizer

d_1 = "招聘具有后台开发专业技能的人员"
d_2 = "招聘具有数据分析能力的工程师"
result_1 = jieba.cut(d_1)
space_word_1 = " ".join(result_1)

result_2 = jieba.cut(d_2)
space_word_2 = " ".join(result_2)

#初始化tf-idf向量化对象
tfidf_vector = TfidfVectorizer()
#将两篇文档组合成一个语料库，并进行向量化
data_vector = tfidf_vector.fit_transform([space_word_1,space_word_2])
#打印出每一个维度的特征值
print(tfidf_vector.get_feature_names())
#打印出转换后的结果
print(data_vector.toarray())
```

```
"""
结果为:
['专业技能', '人员', '具有', '后台', '工程师', '开发', '招聘', '数据分析', '能力']
[[0.44665616 0.44665616 0.31779954 0.44665616 0.         0.44665616
  0.31779954 0.         0.         ]
 [0.         0.         0.35520009 0.         0.49922133 0.
  0.35520009 0.49922133 0.49922133]]
"""
```

在实际问题中应用贝叶斯分类