

基于朴素贝叶斯的文本分类算法

作者: 灵魂机器

联系方式: soulmachine@gmail.com

作者博客: www.yanjiuyanjiu.com

摘要: 常用的文本分类方法有支持向量机、K-近邻算法和朴素贝叶斯。其中朴素贝叶斯具有容易实现, 运行速度快等特点, 被广泛使用。本文详细介绍了朴素贝叶斯的基本原理, 讨论了两种常见模型: 多项式模型 (MM) 和伯努利模型 (BM), 实现了可运行的代码, 并进行了一些数据测试。

关键字: 朴素贝叶斯; 文本分类

Text Classification Algorithm Based on Naive Bayes

Author: soulmachine

Email: soulmachine@gmail.com

Blog: www.yanjiuyanjiu.com

Abstract: Usually there are three methods for text classification: SVM, KNN and Naïve Bayes. Naïve Bayes is easy to implement and fast, so it is widely used. This article introduced the theory of Naïve Bayes and discussed two popular models: multinomial model(MM) and Bernoulli model(BM) in details, implemented runnable code and performed some data tests.

Keywords: naïve bayes; text classification

第 1 章 贝叶斯原理

1.1 贝叶斯公式

设 A、B 是两个事件, 且 $P(A) > 0$, 称

$$P(Y|X) = \frac{P(XY)}{P(X)}$$

为在事件 A 发生的条件下事件 B 发生的 条件概率。

乘法公式 $P(XYZ)=P(Z|XY)P(Y|X)P(X)$

全概率公式 $P(X)=P(X|Y_1)+ P(X|Y_2)+\dots+ P(X|Y_n)$

贝叶斯公式 $P(Y_i|X)=\frac{P(XY_i)}{P(X)}=\frac{P(X|Y_i)P(Y_i)}{P(X)}=\frac{P(X|Y_i)P(Y_i)}{\sum_{j=1}^n P(X|Y_j)}$

在此处，贝叶斯公式，我们要用到的是 $P(Y_i|X)=\frac{P(X|Y_i)P(Y_i)}{P(X)}$

以上公式，请读者参考[《概率论与数理统计（第五版）》](#)的 1.4 节“条件概率”（这里将原书中的 A 换成了 X，B 换成了 Y），获得更深的理解。

1.2 贝叶斯定理在分类中的应用

在分类（classification）问题中，常常需要把一个事物分到某个类别。一个事物具有很多属性，把它的众多属性看做一个向量，即 $x=(x_1,x_2,x_3,\dots,x_n)$ ，用 x 这个向量来代表这个事物。类别也是有很多种，用集合 $Y=\{y_1,y_2,\dots,y_m\}$ 表示。如果 x 属于 y_1 类别，就可以给 x 打上 y_1 标签，意思是说 x 属于 y_1 类别。这就是所谓的**分类(Classification)**。

x 的集合记为 X ，称为属性集。一般 X 和 Y 的关系是不确定的，你只能在某种程度上说 x 有多大可能性属于类 y_1 ，比如说 x 有 80% 的可能性属于类 y_1 ，这时可以把 X 和 Y 看做是随机变量， $P(Y|X)$ 称为 Y 的**后验概率**（posterior probability），与之相对的， $P(Y)$ 称为 Y 的**先验概率**（prior probability）^[2]。

在训练阶段，我们要根据从训练数据中收集的信息，对 X 和 Y 的每一种组合学习**后验概率 $P(Y|X)$** 。分类时，来了一个实例 x ，在刚才训练得到的一堆后验概率中找出所有的 $P(Y|x)$ ，其中最大的那个 y ，即为 x 所属分类。根据贝叶斯公式，后验概率为

$$P(Y|X)=\frac{P(X|Y)P(Y)}{P(X)}$$

在比较不同 Y 值的后验概率时，分母 $P(X)$ 总是常数，因此可以忽略。先验概率 $P(Y)$ 可以通过计算训练集中属于每一个类的训练样本所占的比例容易地估计。

我们来举个简单的例子，让读者对上述思路有个形象的认识^[3]。

考虑一个医疗诊断问题，有两种可能的假设：（1）病人有癌症。（2）病人无癌症。样本数据来自某化验测试，它也有两种可能的结果：阳性和阴性。假设我们已经有先验知识：在所有人口中只有 0.008 的人患病。此外，化验测试对有病的患者有 98% 的可能返回阳性结果，对无病患者有 97% 的可能返回阴性结果。

上面的数据可以用以下概率式子表示：

$P(\text{cancer})=0.008, P(\text{无 cancer})=0.992$

$P(\text{阳性}|\text{cancer})=0.98, P(\text{阴性}|\text{cancer})=0.02$

$P(\text{阳性}|\text{无 cancer})=0.03, P(\text{阴性}|\text{无 cancer})=0.97$

假设现在有一个新病人，化验测试返回阳性，是否将病人断定为有癌症呢？

在这里， $Y=\{\text{cancer}, \text{无 cancer}\}$ ，共两个类别，这个新病人是一个样本，他有一个属性阳性，可以令 $x=(\text{阳性})$ 。

我们可以来计算各个类别的后验概率：

$$P(\text{cancer} | \text{阳性}) = P(\text{阳性} | \text{cancer})p(\text{cancer}) = 0.98 * 0.008 = 0.0078$$

$$P(\text{无 cancer} | \text{阳性}) = P(\text{阳性} | \text{无 cancer}) * p(\text{无 cancer}) = 0.03 * 0.992 = 0.0298$$

因此，应该判断为无癌症。

在这个例子中，类条件概率， $P(\text{cancer} | \text{阳性})$ 和 $P(\text{无 cancer} | \text{阳性})$ 直接告诉了我们。

一般地，对**类条件概率** $P(X|Y)$ 的估计，有朴素贝叶斯分类器和贝叶斯信念网络两种方法，这里介绍朴素贝叶斯分类器。

1.3 朴素贝叶斯分类器

1、条件独立性

给定类标号 y ，朴素贝叶斯分类器在估计类条件概率时假设属性之间条件独立。条件独立假设可以形式化的表达如下：

$$P(X | Y = y) = \prod_{i=1}^n P(x_i | Y = y)$$

其中每个训练样本可用一个属性向量 $X=(x_1, x_2, x_3, \dots, x_n)$ 表示，各个属性之间条件独立。

比如，对于一篇文章，

Good good study, Day day up.

可以用一个**文本特征向量**来表示， $x=(\text{Good}, \text{good}, \text{study}, \text{Day}, \text{day}, \text{up})$ 。一般各个词语之间肯定不是相互独立的，有一定的上下文联系。但在朴素贝叶斯文本分类时，我们假设个单词之间没有联系，可以用一个文本特征向量来表示这篇文章，这就是“朴素”的来历。

2、朴素贝叶斯如何工作

有了**条件独立假设**，就不必计算 X 和 Y 的每一种组合的**类条件概率**，只需对给定的 Y ，计算每个 x_i 的条件概率。后一种方法更实用，因为它不需要很大的训练集就能获得较好的概率估计。

3、估计分类属性的条件概率

$P(x_i|Y=y)$ 怎么计算呢？它一般根据类别 y 下包含属性 x_i 的实例的比例来估计。以文本分类为例， x_i 表示一个单词， $P(x_i|Y=y)$ =包含该类别下包含单词的 x_i 的文章总数/ 该类别下的文章总数。

4、贝叶斯分类器举例

假设给定了如下训练样本数据，我们学习的目标是根据给定的天气状况判断你对 PlayTennis 这个请求的回答是 Yes 还是 No。

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

可以看到这里样本数据集提供了 14 个训练样本，我们将使用此表的数据，并结合朴素贝叶斯分类器来分类下面的新实例：

$x = (\text{Outlook} = \text{Sunny}, \text{Temperature} = \text{Cool}, \text{Humidity} = \text{High}, \text{Wind} = \text{Strong})$

在这个例子中，属性向量 $X=(\text{Outlook}, \text{Temperature}, \text{Humidity}, \text{Wind})$ ，类集合 $Y=\{\text{Yes}, \text{No}\}$ 。我们需要利用训练数据计算后验概率 $P(\text{Yes}|x)$ 和 $P(\text{No}|x)$ ，如果 $P(\text{Yes}|x)>P(\text{No}|x)$ ，那么新实例分类为 Yes，否则为 No。

为了计算后验概率，我们需要计算先验概率 $P(\text{Yes})$ 和 $P(\text{No})$ 和类条件概率 $P(x_i|Y)$ 。

因为有 9 个样本属于 Yes，5 个样本属于 No，所以 $P(\text{Yes})=9/14$, $P(\text{No})=5/14$ 。

类条件概率计算如下：

$P(\text{Outlook} = \text{Sunny}|\text{Yes})=2/9$ $P(\text{Outlook} = \text{Sunny}|\text{No})=3/5$

$P(\text{Temperature} = \text{Cool}|\text{Yes})=3/9$ $P(\text{Temperature} = \text{Cool}|\text{No})=1/5$

$P(\text{Humidity} = \text{High}|\text{Yes})=3/9$ $P(\text{Humidity} = \text{High}|\text{No})=4/5$

$P(\text{Wind} = \text{Strong}|\text{Yes})=3/9$ $P(\text{Wind} = \text{Strong}|\text{No})=3/5$

后验概率计算如下：

$P(\text{Yes} | x) = P(\text{Outlook} = \text{Sunny}|\text{Yes}) \times P(\text{Temperature} = \text{Cool}|\text{Yes}) \times P(\text{Humidity} = \text{High}|\text{Yes}) \times P(\text{Wind} = \text{Strong}|\text{Yes}) \times P(\text{Yes}) = 2/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14 = 2/243 = 9/1701 \approx 0.00529$

$P(\text{No} | x) = P(\text{Outlook} = \text{Sunny}|\text{No}) \times P(\text{Temperature} = \text{Cool}|\text{No}) \times P(\text{Humidity} = \text{High}|\text{No}) \times P(\text{Wind} = \text{Strong}|\text{No}) \times P(\text{No}) = 3/5 \times 1/5 \times 4/5 \times 3/5 \times 5/14 = 18/875 \approx 0.02057$

通过计算得出 $P(\text{No} | x) > P(\text{Yes} | x)$ ，所以该样本分类为 No[3]。

5、条件概率的 m 估计

假设有来了一个新样本 $x_1 = (\text{Outlook} = \text{Cloudy}, \text{Temperature} = \text{Cool}, \text{Humidity} = \text{High}, \text{Wind} = \text{Strong})$ ，要求对其分类。我们来开始计算，

$P(\text{Outlook} = \text{Cloudy}|\text{Yes})=0/9=0$ $P(\text{Outlook} = \text{Cloudy}|\text{No})=0/5=0$

计算到这里，大家就会意识到，这里出现了一个新的属性值，在训练样本中所没有的。如果有一个属性的类条件概率为 0，则整个类的后验概率就等于 0，我们可以直接得到后验概率

$P(\text{Yes} | x_1) = P(\text{No} | x_1) = 0$ ，这时二者相等，无法分类。

当训练样本不能覆盖那么多的属性值时，都会出现上述的窘境。简单的使用样本比例来估计类条件概率的方法太脆弱了，尤其是当训练样本少而属性数目又很大时。

解决方法是使用 m 估计(m -estimate)方法来估计条件概率：

$$P(x_i | y_j) = \frac{n_c + mp}{n + m}$$

n 是类 y_j 中的样本总数， n_c 是类 y_j 中取值 x_i 的样本数， m 是称为等价样本大小的参数，而 p 是用户指定的参数。如果没有训练集（即 $n=0$ ），则 $P(x_i | y_j) = p$ ，因此 p 可以看作是在类 y_j 的样本中观察属性值 x_i 的先验概率。等价样本大小决定先验概率和观测概率 n_c/n 之间的平衡 [2]。

第 2 章 朴素贝叶斯文本分类算法

现在开始进入本文的主旨部分：如何将贝叶斯分类器应用到文本分类上来。

2.1 文本分类问题

在文本分类中，假设我们有一个文档 $d \in X$ ， X 是文档向量空间(document space)，和一个固定的类集合 $C = \{c_1, c_2, \dots, c_j\}$ ，类别又称为标签。显然，文档向量空间是一个高维度空间。我们把一堆打了标签的文档集合 $\langle d, c \rangle$ 作为训练样本， $\langle d, c \rangle \in X \times C$ 。例如：

$\langle d, c \rangle = \{\text{Beijing joins the World Trade Organization, China}\}$

对于这个只有一句话的文档，我们把它归类到 China，即打上 china 标签。

我们期望用某种训练算法，训练出一个函数 γ ，能够将文档映射到某一个类别：

$$\gamma: X \rightarrow C$$

这种类型的学习方法叫做有监督学习，因为事先有一个监督者（我们事先给出了一堆打好标签的文档）像个老师一样监督着整个学习过程。

朴素贝叶斯分类器是一种有监督学习，常见有两种模型，多项式模型(multinomial model)和伯努利模型(Bernoulli model)。

2.2 多项式模型

1、基本原理

在多项式模型中，设某文档 $d = (t_1, t_2, \dots, t_k)$ ， t_k 是该文档中出现过的单词，允许重复，则

先验概率 $P(c) = \text{类 } c \text{ 下单词总数} / \text{整个训练样本的单词总数}$

类条件概率 $P(t_k | c) = (\text{类 } c \text{ 下单词 } t_k \text{ 在各个文档中出现过的次数之和} + 1) / (\text{类 } c \text{ 下单词总数} + |V|)$

V 是训练样本的单词表（即抽取单词，单词出现多次，只算一个）， $|V|$ 则表示训练样本包含

多少种单词。在这里， $m=|V|$, $p=1/|V|$ 。

$P(t_k|c)$ 可以看作是单词 t_k 在证明 d 属于类 c 上提供了多大的证据，而 $P(c)$ 则可以认为是类别 c 在整体上占多大比例(有多大可能性)。

2、伪代码

//C, 类别集合, D, 用于训练的文本文件集合

TrainMultiNomialNB(C,D) {

// 单词出现多次, 只算一个

V←ExtractVocabulary(D)

// 单词可重复计算

N←CountTokens(D)

for each $c \in C$

// 计算类别 c 下的单词总数

// N 和 N_c 的计算方法和 [Introduction to Information Retrieval](#) 上的不同, 个人认为

//该书是错误的, 先验概率和类条件概率的计算方法应当保持一致

$N_c \leftarrow \text{CountTokensInClass}(D,c)$

prior[c]← N_c/N

// 将类别 c 下的文档连接成一个大字符串

$\text{text}_c \leftarrow \text{ConcatenateTextOfAllDocsInClass}(D,c)$

for each $t \in V$

// 计算类 c 下单词 t 的出现次数

$T_{ct} \leftarrow \text{CountTokensOfTerm}(\text{text}_c,t)$

for each $t \in V$

//计算 $P(t|c)$

$$\text{condprob}[t][c] \leftarrow \frac{T_{ct} + 1}{\sum_{t' \in V} T_{ct'} + |V|}$$

return V,prior,condprob

}

ApplyMultiNomialNB(C,V,prior,condprob,d) {

// 将文档 d 中的单词抽取出来, 允许重复, 如果单词是全新的, 在全局单词表 V 中都

// 没出现过, 则忽略掉

W←ExtractTokensFromDoc(V,d)

for each $c \in C$

score[c]←prior[c]

for each $t \in W$

if $t \in Vd$

score[c] *= condprob[t][c]

return max(score[c])

}

3、举例

给定一组分类好了的文本训练数据，如下：

docId	doc	类别 In c=China?
1	Chinese Beijing Chinese	yes
2	Chinese Chinese Shanghai	yes
3	Chinese Macao	yes
4	Tokyo Japan Chinese	no

给定一个新样本 `Chinese Chinese Chinese Tokyo Japan`，对其进行分类。

该文本用属性向量表示为 $d=(\text{Chinese}, \text{Chinese}, \text{Chinese}, \text{Tokyo}, \text{Japan})$ ，类别集合为 $Y=\{\text{yes}, \text{no}\}$ 。

类 yes 下总共有 8 个单词，类 no 下总共有 3 个单词，训练样本单词总数为 11，因此 $P(\text{yes})=8/11$ ， $P(\text{no})=3/11$ 。类条件概率计算如下：

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

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

$$P(\text{Chinese} | \text{no})=(1+1)/(3+6)=2/9$$

$$P(\text{Japan} | \text{no})=P(\text{Tokyo} | \text{no})=(1+1)/(3+6)=2/9$$

分母中的 8，是指 yes 类别下 text_c 的长度，也即训练样本的单词总数，6 是指训练样本有 Chinese, Beijing, Shanghai, Macao, Tokyo, Japan 共 6 个单词，3 是指 no 类下共有 3 个单词。

有了以上类条件概率，开始计算后验概率，

$$P(\text{yes} | d)=(3/7)^3 \times 1/14 \times 1/14 \times 8/11=108/184877 \approx 0.00058417$$

$$P(\text{no} | d)=(2/9)^3 \times 2/9 \times 2/9 \times 3/11=32/216513 \approx 0.00014780$$

因此，这个文档属于类别 china。

2.3 伯努利模型

1、基本原理

$P(c)$ = 类 c 下文件总数/整个训练样本的文件总数

$P(t_k|c)$ =(类 c 下包含单词 t_k 的文件数+1)/(类 c 下单词总数+2)

在这里， $m=2$ ， $p=1/2$ 。

后验概率的计算，也有点变化，见下面的伪代码。

2、伪代码

//C，类别集合，D，用于训练的文本文件集合

```
TrainBernoulliNB(C, D) {
```

```
    // 单词出现多次，只算一个
```

```
    V ← ExtractVocabulary(D)
```

```
    // 计算文件总数
```

```
    N ← CountDocs(D)
```

```

for each c ∈ C
    // 计算类别 c 下的文件总数
    Nc ← CountDocsInClass(D,c)
    prior[c] ← Nc/N
    for each t ∈ V
        // 计算类 c 下包含单词 t 的文件数
        Nct ← CountDocsInClassContainingTerm(D,c,t)
        //计算 P(t|c)
        condprob[t][c] ← (Nct+1)/(Nct+2)
    return V,prior,condprob
}

ApplyBernoulliNB(C,V,prior,condprob,d) {
    // 将文档 d 中单词表抽取出来，如果单词是全新的，在全局单词表 V 中都没出现过，
    // 则舍弃
    V_d ← ExtractTermsFromDoc(V,d)
    for each c ∈ C
        score[c] ← prior[c]
        for each t ∈ V
            if t ∈ V_d
                score[c] *= condprob[t][c]
            else
                score[c] *= (1-condprob[t][c])
    return max(score[c])
}

```

3、举例

还是使用前面例子中的数据，不过模型换成了使用伯努利模型。

类 yes 下总共有 3 个文件，类 no 下有 1 个文件，训练样本文件总数为 11，因此 $P(\text{yes})=3/4$,

$P(\text{Chinese} | \text{yes})=(3+1)/(3+2)=4/5$

$P(\text{Japan} | \text{yes})=P(\text{Tokyo} | \text{yes})=(0+1)/(3+2)=1/5$

$P(\text{Beijing} | \text{yes})= P(\text{Macao}|\text{yes})= P(\text{Shanghai} | \text{yes})=(1+1)/(3+2)=2/5$

$P(\text{Chinese}|\text{no})=(1+1)/(1+2)=2/3$

$P(\text{Japan}|\text{no})=P(\text{Tokyo}|\text{no})=(1+1)/(1+2)=2/3$

$P(\text{Beijing}|\text{no})= P(\text{Macao}|\text{no})= P(\text{Shanghai} | \text{no})=(0+1)/(1+2)=1/3$

有了以上类条件概率，开始计算后验概率，

$P(\text{yes} | d)=P(\text{yes}) \times P(\text{Chinese}|\text{yes}) \times P(\text{Japan}|\text{yes}) \times P(\text{Tokyo}|\text{yes}) \times (1-P(\text{Beijing}|\text{yes})) \times (1-P(\text{Shanghai}|\text{yes})) \times (1-P(\text{Macao}|\text{yes}))$

$=3/4 \times 4/5 \times 1/5 \times 1/5 \times (1-2/5) \times (1-2/5) \times (1-2/5)=81/15625 \approx 0.005$

$P(\text{no} | d)= 1/4 \times 2/3 \times 2/3 \times 2/3 \times (1-1/3) \times (1-1/3) \times (1-1/3)=16/729 \approx 0.022$

因此，这个文档不属于类别 china。

2.4 两个模型的区别

二者的计算粒度不一样，多项式模型以单词为粒度，伯努利模型以文件为粒度，因此二者的先验概率和类条件概率的计算方法都不同。

计算后验概率时，对于一个文档 d ，多项式模型中，只有在 d 中出现过的单词，才会参与后验概率计算，伯努利模型中，没有在全局单词表中出现的单词，也会参与计算，不过是作为“反方”参与的。

第 3 章 代码详解

本文附带了一个 eclipse 工程，有完整的源代码，以及一个微型文本训练库。

ChineseSplitter 用于中文分词，StopWordsHandler 用于判断一个单词是否是停止词，ClassifyResult 用于保存结果，IntermediateData 用于预处理文本语料库，TrainedModel 用于保存训练后得到的数据，NaiveBayesClassifier 是基础类，包含了贝叶斯分类器的主要代码，MultiNomialNB 是多项式模型，类似的，BernoulliNB 是伯努利模型，二者都继承自 NaiveBayesClassifier，都只重写了父类的计算先验概率，类条件概率和后验概率这 3 个函数。

3.1 中文分词

中文分词不是本文的重点，这里我们直接使用第三方工具，本源码使用的是[极易中文分词组件](#)，你还可以使用 [MMSEG](#)，中科院的 [ICTCLAS](#) 等等。

```
/**
 * 对给定的文本进行中文分词.
 *
 * @param text
 *         给定的文本
 * @param splitToken
 *         用于分割的标记,如"|"
 * @return 分词完毕的文本
 */
public String split(final String text, final String splitToken) {
    String result = null;

    try {
        result = analyzer.segment(text, splitToken);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

```

    }
    return result;
}

```

3.2 停止词处理

停止词(Stop Word)是指那些无意义的字或词，如“的”、“在”等。去掉文档中的停止词也是必须的一项工作,这里简单的定义了一些常见的停止词，并根据这些常用停止词在分词时进行判断。

```

/** 常用停用词. */
private static String[] stopWordsList = {
    // 来自 c:\Windows\System32\NOISE.CHS
    "的", "一", "不", "在", "人", "有", "是", "为", "以", "于", "上",
    "他", "而",
    "后", "之", "来", "及", "了", "因", "下", "可", "到", "由", "这",
    "与", "也",
    "此", "但", "并", "个", "其", "已", "无", "小", "我", "们", "起",
    "最", "再",
    "今", "去", "好", "只", "又", "或", "很", "亦", "某", "把", "那",
    "你", "乃",
    "它",
    // 来自网络
    "要", "将", "应", "位", "新", "两", "中", "更", "我们", "自己", "没有",
    "\"", "\"",
    "(", ")", " ";
};

/**
 * 判断一个词是否是停止词.
 *
 * @param word
 *         要判断的词
 * @return 是停止词, 返回true, 否则返回false
 */
public static boolean isStopWord(final String word) {
    for (int i = 0; i < stopWordsList.length; ++i) {
        if (word.equalsIgnoreCase(stopWordsList[i])) {
            return true;
        }
    }
    return false;
}

```

3.3 预处理数据

我们这里使用[搜狗的文本分类语料库](#)作为训练样本，把 SogouC.reduced.20061102.tar.gz 解压到 D 盘，目录结构为

```
D:\Reduced
    |-- C000008
    |-- C000010
    |-- C000013
    |-- C000014
    |-- C000016
    |-- C000020
    |-- C000022
    |-- C000023
    |-- C000024
```

IntermediateData.java 主要用于处理文本数据，将所需要的信息计算好，存放到数据库文件中。

中间数据文件主要保存了如下信息，

```
/** 单词x在类别c下出现的总数. */
    public HashMap[] filesOfXC;
    /** 给定分类下的文件数目. */
    public int[] filesOfC;
    /** 根目录下的文件总数. */
    public int files;

    /** 单词x在类别c下出现的总数 */
    public HashMap[] tokensOfXC;
    /** 类别c下所有单词的总数. */
    public int[] tokensOfC;
    /** 整个语料库中单词的总数. */
    public int tokens;
    /** 整个训练语料所出现的单词. */
    public HashSet<String> vocabulary;
```

我们使用命令

```
IntermediateData d:\Reduced\ gbk d:\reduced.db
```

将文本训练库的信息计算好，保存到中间文件中。以后的阶段，我们都不再需要文本语料库了，只需要 reduced.db。

3.3 训练

基本的框架代码都在 `NaiveBayesClassifier` 中, `MultiNomialNB` 和 `BernoulliNB` 都只是重新实现 (override) 了

```
/** 计算先验概率 $P(c)$ . */
protected void calculatePc() {
}

/** 计算类条件概率 $P(x|c)$ . */
protected void calculatePxc() {
}

/**
 * 计算文本属性向量 $x$ 在类 $c_j$ 下的后验概率 $P(c_j|x)$ .
 *
 * @param x
 *         文本属性向量
 * @param cj
 *         给定的类别
 * @return 后验概率
 */
protected double calcProd(final String[] x, final int cj) {
    return 0;
}
```

这三个函数。

训练函数如下:

```
public final void train(String intermediateData, String modelFile) {
    // 加载中间数据文件
    loadData(intermediateData);

    model = new TrainedModel(db.classifications.length);

    model.classifications = db.classifications;
    model.vocabulary = db.vocabulary;
    // 开始训练
    calculatePc();
    calculatePxc();
    db = null;

    try {
        // 用序列化, 将训练得到的结果存放到模型文件中
        ObjectOutputStream out = new ObjectOutputStream(
```

```

        new FileOutputStream(modelFile));
    out.writeObject(model);
    out.close();
} catch (IOException e) {
    e.printStackTrace();
}
}
}

```

我们使用命令：

```
MultiNomialNB -t d:\reduced.db d:\reduced.mdl
```

开始训练，得到的模型文件保存在 reduced.mdl 中。

3.4 分类

有了模型文件，就可以用它来进行分类了。

可以使用命令

```
MultiNomialNB d:\reduced.mdl d:\temp.txt gbk
```

对文本文件 temp.txt 进行分类。

还可以将当初训练出这个模型文件的文本库，进行分类，看看正确率有多少，即“吃自己的狗食”，命令行如下

```
MultiNomialNB -r d:\reduced\ gbk d:\reduced.mdl
```

分类函数如下：

```

/**
 * 对给定的文本进行分类.
 *
 * @param text
 *         给定的文本
 * @return 分类结果
 */
public final String classify(final String text) {
    String[] terms = null;
    // 中文分词处理(分词后结果可能还包含有停用词)
    terms = textSplitter.split(text, " ").split(" ");
    // 去掉停用词，以免影响分类
    terms = ChineseSplitter.dropStopWords(terms);

    double probability = 0.0;
    // 分类结果
    List<ClassifyResult> crs = new ArrayList<ClassifyResult>();
    for (int i = 0; i < model.classifications.length; i++) {
        // 计算给定的文本属性向量terms在给定的分类Ci中的分类条件概率

```

```

    probability = calcProd(terms, i);
    // 保存分类结果
    ClassifyResult cr = new ClassifyResult();
    cr.classification = model.classifications[i]; // 分类
    cr.probability = probability; // 关键字在分类的条件概率
    System.out.println("In process...");
    System.out.println(model.classifications[i] + ": " + probability);
    crs.add(cr);
}

// 找出最大的元素
ClassifyResult maxElem = (ClassifyResult) java.util.Collections.max(
    crs, new Comparator() {
        public int compare(final Object o1, final Object o2) {
            final ClassifyResult m1 = (ClassifyResult) o1;
            final ClassifyResult m2 = (ClassifyResult) o2;
            final double ret = m1.probability - m2.probability;
            if (ret < 0) {
                return -1;
            } else {
                return 1;
            }
        }
    });

return maxElem.classification;
}

```

测试正确率的函数 `getCorrectRate()`，核心代码就是对每个文本文件调用 `classify()`，将得到的类别和原始的类别比较，经过统计后就可以得到百分比。

更多细节请读者阅读源代码。

参考文献

- [1] Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze, [Introduction to Information Retrieval](#), Cambridge University Press, 2008, chapter 13, *Text classification and Naive Bayes*.
- [2] Pang-Ning Tan, Michael Steinbach, Vipin Kumar, [《数据挖掘导论》](#)，北京：人民邮电出版社，2007，第 140~145 页。
- [2] 石志伟，吴功宜，“[基于朴素贝叶斯分类器的文本分类算法](#)”，第一届全国信息检索与内容安全学术会议，2004
- [3] 洞庭散人，“[基于朴素贝叶斯分类器的文本分类算法（上）](#)”，“[基于朴素贝叶斯分类器的](#)

[文本分类算法（下）](#)”，2008

[4]DL88250, “[朴素贝叶斯中文文本分类器的研究与实现（1）](#)”，“[朴素贝叶斯中文文本分类器的研究与实现（2）](#)”，2008

由于笔者水平有限，本文出现错误在所难免，欢迎读者批评指正，请到原文发表评论或给我发 email，谢谢 :)

原文地址：<http://www.yanjiuyanjiu.com/2010/05/28/naive-bayes-text-classification/>