贝叶斯分类器在文本分类中的应用及改进

——人工智能课程期末论文

施炎辰
09300240031
Fudan University
09300240031@fudan.edu.cn

Abstract

文本分类（Document Classification或Document Categorization），是数据挖掘（Data Mining）中的重要分支。本文将通过有监督的学习（Supervised Learning），基于贝叶斯分类器（Bayesian classifier），通过对其进行研究和改进，从而产生布尔型，词频型，加权词频型三种贝叶斯分类器，对一组新闻文档的训练文本数据进行训练，对另外一组测试数据进行分类并测试。

1 Introduction

随着网络的发展，人们获得信息的速度以及数量大幅度上升，文本自动分类成为了一个重要的问题。在垃圾邮件鉴别，新闻类别甄别，科学文献分类等很多方面都有重要的作用。对于文本分类目前常用的方法有决策树，贝叶斯（Bayes）方法，神经网络方法（ANN），K2最近邻法（KNN），遗传算法（GA），支持向量机（SVM）等，其中最常用的前两种方法 [1]。朴素贝叶斯分类器（naive Bayesian classifier，NBC）作为贝叶斯网分类器的一种，是目前公认的一种简单而且有效的概率分类方法。其性能可与决策树、神经网络等算法相媲美，在某些领域汇中性能优越 [2]。

贝叶斯分类器中最著名的是朴素贝叶斯分类器（naive Bayesian classifier)，本文除了实现朴素的贝叶斯分类器以外还将对其进行一定的改进，并进行比较。一个文本分类的系统主要由文本的向量模型表示、文本的特征选择以及分类器训练三部分组成。其中文本的向量的维度将会很大，训练样本也将很大。这里的训练样本来源于CMU的一份用于朴素贝叶斯文本分类训练用的数据。[1]这个地址由模式识别课的池明旻老师提供，但是本文绝不是模式识别课的作业。

[1] http://www-2.cs.cmu.edu/afs/cs/project/theo-11/www/naive-bayes.html

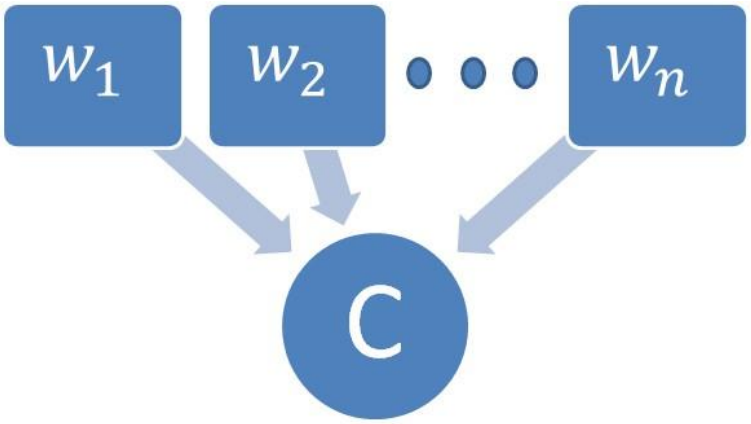

图 1: Schematic diagram for Beyasian classifier

2 Bayesian classifier

朴素的贝叶斯分类器主要是基于贝叶斯公式实现的，公式的定义为公式 1

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}; \tag{1}$$

在文本分类中，一个主要的假设是所有单词之间，假设C为类别，共有k类，记为$C = \{C_1, C_2, \ldots, C_k\}$，$w$为单词，则原理图为图 1。
假设每个类C_i的先验概率为$P(C_i), i = 1, 2, \ldots, k$，在这里我们可以设为每个类$C_i$占样本集总数$N$的比重。假设对于一个测试样本$\boldsymbol{x}$，则在类$C_i$的得到它的条件概率概率为$P(\boldsymbol{x}|C_i)$。根据贝叶斯公式 1，则$\boldsymbol{x}$属于$C_i$的后验概率为公式 2

$$P(C_i|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|C_i)P(C_i)}{P(\boldsymbol{x})}; \tag{2}$$

由于$P(\boldsymbol{x})$对于所有的类均为一个常量，我们可以将公式 2改为公式 3

$$P(C_i|\boldsymbol{x}) \propto P(\boldsymbol{x}|C_i)P(C_i); \tag{3}$$

为了防止$P(C_i)$的值为零，影响计算结构，我们利用拉普拉斯估计（Laplace Estimator），定义$P(C_i)$为等式 4

$$P(C_i) = \frac{1 + |D_c|}{k + |D_c|}; \tag{4}$$

其中$|D_C|$训练集的文本数量，k为类的数量。
如果每个C_i的训练集的文本数量都相等，那么公式 3可以简化为公式 5、、

$$P(C_i|\boldsymbol{x}) \propto P(\boldsymbol{x}|C_i); \tag{5}$$

所以对于一个输入$\boldsymbol{x}$，我们通过寻找$argmax\{P(\boldsymbol{x}|C_i)P(C_i)\}, i = 1, 2 \ldots, k$，决定$\boldsymbol{x}$属于$C_i$。
输入$\boldsymbol{x}$，表示这个文档的一些特征词。特征词我们可以认为是除了一些没有意义的连接词、动词、名词，如”is”,”for”,”I”之外的其他词语。那么有$\boldsymbol{x} = \{w_1, w_2, \ldots, w_m\}$，其中$m$为$\boldsymbol{x}$ 的特征词的数量，即$m = |\boldsymbol{x}|$。又因为我们假设了每个特征词之间相互独立，所以我们可以得

到式子 6。

$$P(\boldsymbol{x}|C_i) = P((w_1, w_2, \ldots, w_m)|C_i) = \prod_{j=1}^{m} P(w_j|C_i); \tag{6}$$

其中$P(w_j|C_i)$表示某个单词w_j在类C_i中发生的概率。因此公式 2转变为公式 7

$$P(C_i|\boldsymbol{x}) \propto P(C_i) \prod_{j=1}^{m} P(w_j|C_i); \tag{7}$$

根据$P(w_j|C_i)$的设置不同，我们就能够获得不同的，我们可以分为一下三种不同的分类器：

1. 布尔型：
   仅仅考虑某个单词w_j在一个文档中是否出现。出现则记为1，否则为0。那么我们则可以定义$P(w_j|C_i)$为公式 8 [3]。

$$P(w_j|C_i) = \frac{1 + N(doc(w_j)|C_i)}{2 + |D_c|}; \tag{8}$$

   这里为了防止数字为零，同样利用了拉普拉斯估计。其中$N(doc(w_j)|C_i)$表示一个特征词w_j在C_i中出现的总文档数。

2. 词频型：
   将一个单词w_j在文档中出现的频率考虑在内，从而的到公式 9。

$$P(w_j|C_i) = \frac{1 + F(w_j|C_i)}{|V| + \sum_{k=1}^{|V|} F(w_k|C_i)}; \tag{9}$$

   其中，$|V|$表示所有特征词的综合。$F(w_j|C_i)$表示C_i类中特征词w_j出现的频率总和。

3. 加权词频型：
   对上述的词频型进行修改，得到公式 10。

$$P(w_j|C_i) = \left(\frac{1 + F(w_j|C_i)}{|V| + \sum_{k=1}^{|V|} F(w_k|C_i)}\right)^{Weight(w_j)}; \tag{10}$$

   这里$Weight(w_j)$表示w_j的一个权重。如果权重越大，w_j在分类器中的作用也越大。相反权重越小，w_j的作用也就越小。当$Weight(w_j) = 0$时，有$P(w_j|C_i) = 1$，即w_j将在分类中完全不起作用。

3 Implementation

对三种分类器将分开讨论。

3.1 Boolean type

根据上面提到的公式，从而可以训练一个布尔型的朴素贝叶斯分类器，从而可以对一个未知分类的文档进行分类。具体的算法为算法 1。 由此我们能够得到该位置文档的分类。
初始化的时间复杂度为$O(N \times v)$，其中N为训练集的大小，v为每个训练文档平均的单词数量。若对n个文件进行分类，那么分类所需的时间复杂度为$O(n \times K \times M \times u)$，其中$K$的定义为类别的数量，$M$的定义为训练集的单词数量。$u$代表查询**useless**数组所需的时间，如果

Algorithm 1 Boolean type

Inputs: x, a vector contains the words of a test document;
K, number of classes;
Local variables: useless, a vector contains the words useless for classify, like "I","a","the";
Initialization: record, the type is vector(map(string,int)) defined in stl;
record[i][w] refers to the number of documents containing word w in class C_i;
Each w in **record** is distinct;
M, number of all words;
D, number of all training documents
dic, a vector refer to the dictionary of all words;
P, a vector contain the prior probability for each class;
for $i = 0$ to K **do**
  Initialize the vector **rate**, make all element in it equal to 1.0
  for $j = 0$ to M **do**
    if $\mathbf{dic}[j]$ is included in **useless then**
      continue;
    end if
    $rate[i] *= (1 + \mathbf{record}[i][\mathbf{dic}[j]]) * \mathbf{P}[i]/(D + 2)$;
  end for
end for
return the maximum of **rate**;

是线性查找的话，$u = |\mathbf{useless}|$。所以总共的时间复杂度为$O(N \times v) + O(n \times K \times M \times u)$。当$n$跟$N$属于基本一个数量级时，因为$v \leq M$，所以有$O(N \times v) + O(n \times K \times M \times u) = O(n \times K \times M \times u)$。

3.2 Frequency type

根据上面提到的公式，从而可以训练一个词频型的朴素贝叶斯分类器，从而可以对一个未知分类的文档进行分类。具体的算法为算法 2。 由此我们能够得到该位置文档的分类。
在保留跟布尔型分类器时间复杂度分析相同的变量名，有初始化的时间复杂度为$O(N \times v)$。为了提高效率，我们可以实现计算好每个组C_i对应的sum值，这个需要$O(K \times M)$ 若对n个文件进行分类，那么分类所需的时间复杂度为$O(n \times N \times M \times u)$ 所以总共的时间复杂度为$O(N \times v) + O(K \times M) + O(n \times K \times M \times u)$。当$n$跟$N$属于基本一个数量级时，因为$v \leq M$，所以有$O(N \times v) + O(K \times M) + O(n \times K \times M \times u) = O(n \times K \times M \times u)$。

3.3 Weighted frequency type

根据上面提到的公式，从而可以训练一个加权词频型的朴素贝叶斯分类器，从而可以对一个未知分类的文档进行分类。其中权重的选择我采用了利用方差。如果一个单词w在每个类中出现的频率的方差越大，我则将给予越小的权重。具体的算法为算法 3。 其中Weight()函

Algorithm 2 Frequency type

Inputs and Local variances: the same as boolean type classifier
Initialization: record[i][w] refers to the time word w has appeared in all documents of class C_i; The rest initialization are same as boolean type classifer
for $i = 0$ to K **do**
  Initialize the vector **rate**, make all element in it equal to 1.0
  for $j = 0$ to M **do**
    if $\mathbf{dic}[j]$ is included in **useless then**
      continue;
    end if
    define sum, the total frequency of all words in class C_i;
    $rate[i] * = (1 + \mathbf{record}[i][\mathbf{dic}[j]]) * \mathbf{P}[i]/(sum + M)$;
  end for
end for
return the maximum of **rate**;

Algorithm 3 Weighted frequency type

Inputs and Local variables: the same as frequency type
Initialization: except all the variables in frequency type, a map(string,int) type variable **deviation** is required to record each word's deviation;
for $i = 0$ to K **do**
  Initialize the vector **rate**, make all element in it equal to 1.0
  for $j = 0$ to M **do**
    if $\mathbf{dic}[j]$ is included in **useless then**
      continue;
    end if
    define sum, the total frequency of all words in class C_i;
    $rate[i] * = Weight(eight((1 + \mathbf{record}[i][\mathbf{dic}[j]])/(sum + M), \mathbf{dict}[j]) * \mathbf{P}[i]$;
  end for
end for
return the maximum of **rate**;

数为一个加权的函数，权值的大小由方差所决定。定义为等式 11

$$Weight(x, s) = \begin{cases} x^6 & s \in \mathbf{keyword} \\ \sqrt{x} & \mathbf{deviation}[s] > 1000 \\ x & 1000 \geq \mathbf{deviation}[s] > 100 \\ x^2 & 100 \geq \mathbf{deviation}[s] > 10 \\ x^3 & 10 \geq \mathbf{deviation}[s] > 1 \\ x^4 & 1 \geq \mathbf{deviation}[s] \end{cases} \tag{11}$$

向量**keyword**是一个可以设定的一些特别关键词。如果包含这些词的话可以人为加大其权限。由此我们能够得到该位置文档的分类。
在时间复杂度上，除了为了加权函数Weight()计算方差之外，其他均相同。在保留跟前两个分类器相同的变量下，有方差的时间复杂度为$O(K \times M)$，所以总的时间复杂度为：$O(N \times v) + O(K \times M) + O(K \times M) + O(n \times K \times M \times u) = O(n \times K \times M \times u)$。

4 Evaluation

通过上面的算法，我们可以获得一个对应的贝叶斯文本分类器。训练数据与测试数据均来源于CMU，已经在Introduction中说明。训练数据集包括了20个不同分类的新闻报道的文本，每个分类均有100个文本文档。测试数据同样为20个分类，每个分类的文本文档同样为100个，测试数据与训练数据的文本文档均不相同。
本人对于对于三种分类器均进行了实现，使用C语言编程，其结果均表示在表格 1。其中第一列表示每个类的名词，第二列到第四列是分类的结果跟真实的类完全相同时的正确率。从中我们可以看出，布尔型跟词频型在表现力上基本差不多，无论是每个类的准确度，总的准确度以及方差均相差不远。而对于加权词频型，我们可以看到，在某些组，如"rec.autos"上有明显的提升，但是在某些类如"sci.med"上准确度反而不及较为朴素的布尔型跟词频型。但总的来说加权词频型拥有更好的表现力，相比起来有8.5%左右的提升。更关键的是，方差比起朴素的两种分类器，几乎减少了一半。这意味着不会有那个类非常差的表现。布尔型跟词频型表现最差的类别分别只有35%与34%的准确度，而加权词频型最差的类别表现也有49%，近乎五成，这对于一个随机猜测仅有5%准确率的分类而言还是可以接受。这说明了通过一个简单的加权（利用方差的加权），就能够一定程度的提高准确度，并且有效的提升稳定性。通过设计更好的加权函数，肯定能够获得更好的准确度和稳定性。
如果将分类到同一大类也认作是正确的话，那么对应的正确率以及方差则如表格 1第五列到第七列中所指出。这里的分到同一大类也就是两个类的成分有交集。比如*comp.graphics*跟*comp.sys.ibm.pc.hardware*我们就可以认为是一个大类，比如*alt.atheism*这个大类中仅有自己这一个类别。我们可以称其为模糊分类（Fuzzy Classification）。那么我们可以看到比起表格 1中第二列到第五列而言除了提高了准确率以外，其他特征，如加权词频算法更为优秀的准确率和更加小的方差的特征依旧可以体现。

5 Conclusion

从本文中我们讨论了贝叶斯分类器的原理及其实现。对布尔型，词频型，加权词频型都进行了测试。这里可以看到词频型拥有略好于布尔型的分类效果，而加权词频型拥有比起两者更加优秀的分类效果。而加权词频型的加权函数可以另外设计，本文中采用了相对快速的计算词频方差的方差来设计。在未来的工作中，可以设计更加优秀，快速，有效的加权算法从而获得一个更加优秀的分类器。

参考文献

[1] J. Qin, F.-R. Chen, and W.-J. Wang, "文本分类中的特征抽取," 计算机应用, vol. 23(2), 2003.

[2] R. Agrawal, T. Imielinski, and A. Swami, "Database mining：a performance perspective," *IEEE Transactions on Knowledge and Data Engineering*, vol. 5, 1993.

[3] J.-M. Li, L.-H. Sun, and Q.-R. Zhang, "一种文本处理中的朴素贝叶斯分类器," 哈尔滨工程大学学报, 2003.

表 1: Accurate rate and variance of each classifier

Class of document	Accu-rate for exact classification			Accu-rate for fuzzy classification		
	Boolean	Frequency	Weighted	Boolean	Frequency	Weighted
alt.atheism	0.51	0.62	0.83	0.51	0.62	0.83
comp.graphics	0.78	0.84	0.56	0.81	0.88	0.86
comp.os.ms-windows.misc	0.40	0.34	0.73	0.80	0.93	0.98
comp.sys.ibm.pc.hardware	0.49	0.49	0.71	0.71	0.95	0.95
comp.sys.mac.hardware	0.66	0.66	0.85	0.72	0.91	0.91
comp.windows.x	0.69	0.50	0.49	0.85	0.88	0.90
misc.forsale	0.35	0.34	0.55	0.47	0.41	0.57
rec.autos	0.42	0.54	0.86	0.45	0.59	0.91
rec.motorcycles	0.64	0.69	0.86	0.65	0.71	0.90
rec.sport.baseball	0.85	0.91	0.96	0.85	0.92	0.99
rec.sport.hockey	0.92	0.89	0.83	0.93	0.93	0.95
sci.crypt	0.91	0.89	0.93	0.92	0.90	0.93
sci.electronics	0.66	0.76	0.68	0.80	0.80	0.71
sci.med	0.92	0.85	0.71	0.92	0.89	0.84
sci.space	0.89	0.89	0.89	0.92	0.92	0.92
soc.religion.christian	1.00	1.00	0.96	1.00	1.00	0.97
talk.politics.guns	0.87	0.85	0.96	0.98	0.98	0.98
talk.politics.mideast	0.96	0.92	0.94	0.98	0.98	0.98
talk.politics.misc	0.79	0.81	0.64	0.92	0.93	0.94
talk.religion.misc	0.57	0.53	0.56	0.90	0.85	0.79
total	0.714	0.716	0.775	0.8050	0.8390	0.8905
variance	0.041657	0.040625	0.023963	0.028668	0.022452	0.010826