

贝叶斯算法及其优化报考

2013年6月5日

刘士杰

1. 概论

贝叶斯过滤算法是纯粹用概率来估计一封邮件是垃圾邮件（简称spam）还是有用邮件（简称ham）。具体来说，不管是哪种语言，一封邮件都是由很多词段（简称token）组成的，比如一句话“我喜欢游泳”就可以分为三个token：“我”“喜欢”“游泳”。算法第一步是利用大量的测试数据（简称corpus），即许多封spam和ham，用里面的token数据估计出如果一封邮件出现某个token，那么他有多少概率是spam。这里要用到贝叶斯公式

贝叶斯定理：

对于binary variable:

所以对于一封邮件，用P(S)和P(H)表示他是spam或ham的概率（很自然P(S)+P(H)=1），用P(W)表示它包含某个token的概率（显然P(W)=1），根据贝叶斯定理，P(S|W)，邮件包含某个token而且是spam的conditional probability就可以算成

P(S)和P(W)一般来说我们都取0.5，这种方法假设一封邮件有一半概率是spam，一般概率是ham，所以叫做unbiased。上边的公式我们简化为

而P(W|S)是指一封邮件是spam而包含某个token的conditional probability，我们可以从corpus里面的数据得出。根据上面的公式，具体算法如下：首先计算

，，

最后，也就是一封邮件如果包含这个token，这封邮件是spam的概率。这个过程是机器学习的过程。

现在对于一封新的邮件，其包含的tokens为Wi(i=1 to n)，如果我们让P(S|Wi)=Pi，根据combining probability的公式，这封邮件为spam的总概率为

因为电脑精度问题，我们做一些代数变化而得出最后比较精确的公式

让，得出

如果一封邮件的P值超过一个门槛（比如0.9），那么他可以被视为spam，否则被视为ham。此时这封邮件的数据可以加入corpus，使数据量更大。

1. 问题与解决

首先以上算法是纯理论的算法，在不同的环境下需要不同的调整，影响因素包括不同的邮件来源，不同的语言风格等，所以需要测试数据来试验。不管是什么算法，理论上是一定会有ham被当做spam（false negative）和spam被当做ham（false positive）出现的情况，这是无法避免的，而且很多情况下减少一个的同时另一个会增加，所以我们需要平衡两者的出现。然后由于大多数可以搜索到的文献都是关于英文的，英文的算法不知道是否可以完全适合中文。其中一个大问题是中文切词，包括中文的语法，断句，歧义等问题。现在唯一的解决方案是用现有的切词API，比如盘古切词（适用于C#）。在学习过程中，我们会用到哈希表来存储每个token的概率，由于数组大小有限，我们不可能把每个token都存入哈希表。另外如果在新邮件中出现以前从未出现过的token，一般来说我们会设其Pi为0.4，但在优化中会提到应该不用考虑他们。还有个问题是标点符号是否要加入考虑范围。最后发spam的人可以通过多使用在ham里常出现的词等方法来扰乱算法，使得P值减小。所以我们不能利用所有的tokens。具体解决方法在优化里有所涉及。

1. 优化
2. 首先贝叶斯算法可以和其他算法一起使用，比如白名单黑名单方法。顾名思义，设置一个由邮箱和IP地址组成的白名单和一个黑名单，由这个名单里邮箱和IP发出的邮件不会被分析而会直接被当成ham和spam。这个名单可以由用户自己定义，也可以由算法决定。比如如果一个邮箱或者IP连续发了10封ham，就直接放入白名单，相对的如果一个邮箱或者IP连续发了10封spam，就直接放入黑名单。这个方法的问题是：1，白名单和黑名单发的邮件是否要加入corpus。2，spam发送者可以利用浮动邮箱（常见）或浮动IP。3，spam发送者可以利用这个漏洞，比如先发10封随便编写的ham，等加入白名单了然后开始发spam。
3. 对于一封新的邮件，我们需要选取其中的token，主要有四种方法：
4. 所有token都用上。这样会比较慢，空间也会用得比较多
5. 一定量的token，比如选取20个token。注意在我们的公式中，极小和极大的P值对最后结果影响较大，而0.5左右的token对结果基本没有影响，也就是说这些token在ham或spam中出现频率极大或极小。所以我们会选取20个和0.5相差最大的token放入公式里面。
6. 正态分布，一般来说我们只取P值小于0.1和P值大于0.9的token，因为他们对我们的结果影响比较大。
7. 一定百分比的token，比如我们选取30%的token。

根据论文<An Evaluation of Naïve Bayesian Anti-Spam Filtering Techniques>，比较好的方法是第二种，选取5-12个token。这样的话我们就不用考虑没有出现过的词，也不用计算那些常用的词比如“的”“我们”，因为他们在spam和ham中出现的次数都差不多，对结果影响并不大。另外最后P的门槛一般可以用0.9，但是具体效果如何得实验得出。

1. 暂时还不知道盘古切词是什么样的，不过根据中文的特征，我们可以只考虑4个字及其以下的token。另外，由于我们采用了b中的方法，P值在0.25到0.75间的token我们不用放入哈希表中，因为他们和0.5比较近，不可能被我们的算法选中，所以不用考虑以节省空间。
2. 复杂度

对于空间复杂度来说，每个邮件会有一个哈希表，而哈希表的大小一般稍大于总token的个数。我们估计一封邮件最多1000字的话，最多也就是1000个token，也只要开1000的数组。所以即使是100000封邮件也是不是很大。空间上应该没有问题。而时间上来说，大部分算法包括搜索和计算哈希表都是O(1)，最慢的也就是分词。虽然不知道盘古分词的时间复杂度（预测在O(n)到O(n^2)之间），但由于一封邮件的字数不太可能超过1000，即使是O(n^3)也不要紧。所以时间上也没有问题。

1. 准确度

由于暂时没有测试数据，准确度还有待测试。但无论如何，false positive和false negative是无法同时避免的。

1. 具体实施
2. 测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 5 | 7 | 9 | 11 | 13 |
| 0.99 | 0.9405/93/26 | 0.9195/91/30 | 0.9395/91/30 | 0.939/89/33 | 0.9405/88/31 |
| 0.9 | 0.942/84/32 | 0.9395/88/33 | 0.94/87/33 | 0.95/67/33 | 0.9505/65/34 |
| 0.5 | 0.945/77/33 | 0.943/81/33 | 0.951/63/35 | 0.952/61/35 | 0.952/61/35 |

