机器学习概论 exp1 Naive Bayes Spam Classfier 实验报告

石景宜 2016011395

> 较上次提交仅修改了 2.3 SPECIAL FEATURES部分

1. 实验设计

1.1 实验原理

分类器原理

根据贝叶斯公式可得，一封邮件中有词$w_1, w_2 \dots w_n$，则该邮件的属性y = spam/ham的概率如下：

$$P(y|w_1, w_2, \dots, w_n) = \frac{P(y)\prod_{i=1}^{n} P(w_i|y)}{\prod_{i=1}^{n} P(w_i))}$$

则$outputy_{predict} = argmax_y P(y)\prod_{i=1}^{n} P(w_i|y)$

结果衡量指标

$$Accuracy = \frac{NumberOfCorrectlyClassifiedRecords}{NumberOfTestRecords}$$

1.2 具体实现

使用python实现，分为两个模块，`bayes.py`和`util.py`

代码结构

`util.py`

包含`initLabel()`,`readfile(total_id)`两个函数，用来提供文件处理的基础支持

- `readfile(total_id)`
  - 用于通过文件编号直接读取文件，文件编号在[0,64620)区间内
  - 返回值为文件中包含的中文词语和发件地址关键词、敏感符号等信息，返回一个去重的列表
- `initLabel()`
  - 读取`label`文件信息，返回一个包含所有文件label的列表

`bayes.py`

包含`MailList`类，其中包含初始化、训练、测试的主要实现。

- 初始化过程
  - `randSetInit(self)`
    - 生成一个[0,64620)区间自然数的随机排列，用于打乱测试、训练次序

        - 划分训练集、测试集（首先按照8:2的比例划分打乱后的数据集，然后按照给定比例在8的部分取训练集，将2的部分整体作为测试集
- 训练过程
    - `train(self)`
        - 读取训练集内所有文件进行训练
        - 训练模型保存在dic字典内，对于关键字`k`，`dic[k][0]`是关键字`k`在spam中出现的总次数，`dic[k][1]`是`k`在ham中出现的总次数
    - `test(self)`
        - 读取测试集内所有文件进行按照原理中公式分别计算邮件为spam/ham的概率，概率大者作为预测结果

$P(w_i|y)$为零的处理

在计算式$outputy_{predict} = argmax_y P(y)\prod_{i=1}^{n} P(w_i|y)$中，$P(w_i|y) = 0$即测试邮件上出现了未曾出现过的关键词会导致整个结果为0，这是我们所不希望看到的。

为了处理这种情形，需要做拉普拉斯平滑处理，将原计算式：

$$P(w_i|y) = \frac{w_i在y类邮件中出现的次数}{y类邮件出现的总次数}$$

改为：

$$P(w_i|y) = \frac{\alpha}{所有关键字个数 + y类邮件总数 * \alpha}$$

为了简单，取α = 1

精度不够的处理

若在程序中直接按照乘法计算$outputy_{predict} = argmax_y P(y)\prod_{i=1}^{n} P(w_i|y)$，会因为精度不够而造成结果为0，于是对该式取一次`log`，则该式变为：

$$outputy_{predict} = log(P(y))\sum_{i=1}^{n} log(P(w_i|y))$$

关键字的选取

由于分词后的文件包含了一些无关紧要的英文字母、空格、符号等信息，在一开始处理时仅将中文作为关键词处理。

观察邮件发现，在中文汉字之外，邮件来源地址，邮件中是否包含网址、邮件地址、电话号码、一些特殊字符也可以成为分类的重要依据。

- 来源地址
    - 通过正则表达式在邮件头信息中找到邮件来源邮箱地址，并按照`@`、`.`切分，作为关键字加入邮件的关键字列表
- 正文中的号码、网址等信息处理
    - 使用正则表达式匹配三位以上数字，若正文中匹配到了结果，则在列表中加入`phone`关键字
    - 判断正文中是否含英文`http`、`com`、`www`、`mail`、`qq`等关键字，若有，则在列表中加入关键字`http`
- 其他关键符号的选取

- 读取所有符号进行处理，输出出现次数大于10且在spam和ham中出现次数悬殊的符号，观察结果后，我选取了这些符号作为关键字：
  - `'\','□','㎡','┻','$','¥','┣','╋','VIP','±','【','】','>','<','[',']','゜','へ','↙'`

结果证明，以上关键字对结果有一定程度的证明影响(大约提高了正确率的0.5%)

偏置项的选取

在最后预测结果判断 $P(y=spam) >= P(y=ham)+bias$中，偏置项对实验结果有大影响，我对偏置项最佳取值进行了实验,实验结果见 *3.实验结果* ，结果发现当bias =-6.25时，正确率最高。

分析：当邮件为垃圾邮件和为正常邮件概率相近时，将邮件预测结果置为垃圾邮件能够提高正确率，因为在数据集中，垃圾邮件所占比例大于正常邮件。但是在实际使用中，这样处理会增加正常邮件误判的概率，容易给用户造成损失。

2. ISSUES

2.1 ISSUE 1：THE SIZE OF TRAINING SET

训练集大小	100%	95%	50%	5%
平均正确率	**97.77%**	97.67%	97.55%	96.88%
最大正确率	97.83%	**97.85%**	97.72%	97.08%
最小正确率	**97.73%**	97.61%	97.32%	96.61%

从表中可明显看出，随着训练集大小的增加，所有正确率在不断提高，但是提高的趋势变小。在所有训练集上，训练

分析：训练集增大意味着关键词增加，对于邮件问题而言，关键词是区分垃圾邮件的主要信息，关键词信息越多越好，训练结果越稳定，但随着关键词数量增加，关键词带来的信息增益逐渐饱和，带来的正确率提高不明显，本实验中训练集为100%时已经趋近饱和，这时候我们需要通过其他信息来进一步区分垃圾邮件。

2.2 ISSUE 2: ZERO-PROBABILITIES

这种情况出现的原因是测试邮件中出现的关键词在训练集中的垃圾邮件或者正常邮件没有出现过，会导致式子：$outputy_{predict} = argmax_y P(y)\prod_{i=1}^{n} P(w_i|y) = 0$

这是极为不合理的结果，我采用了拉普拉斯平滑进行处理，具体处理措施见 *1.2具体实现* 中的 $P(w_i|y)$*为零的处理* 一节。

处理了零概率问题正确率可以上升到97%，不处理零概率问题正确率只能达到94%左右。

2.3 ISSUE 3: SPECIFIC FEATURES

在关键词带来的信息增益达到饱和的情况下，特定属性对提高正确率有很大作用，关于特定属性的选取和处理，我在 *1.2具体实现* 中的 *关键字的选取* 一节中已有部分阐述，现做详细补充，所有处理的特性如下：

- X-Priority
  - 该特性分为三个属性作为关键词加入字典，分别是：
    - `'priorty3','priorty2','priorty1'`
- X-Mailer
  - 匹配该属性后第一个词作为属性加入字典，此属性取值较多
- Time
  - 发送时间按小时分为以下属性，同时加入字典
    - `'night','morning','deep_night','afternoon','noon'`
- phone
  - 该特性是指邮件正文中是否出现号码，若出现号码，将`phone`关键词加入字典
- http
  - 该特性是指邮件正文中是否出现网址、邮件地址等，若出现，将`http`关键词加入字典

加入以上特性的处理后,平均正确率为97.83%，最高准确率为97.90%，最低准确率为97.73%，有所提高但提高并不明显，为了增加这些特定属性对结果的影响，我把他们的出现次数乘以一个权重`weight`，在前几个问题最优化情况下，对`weight`测试如下：

weight	**1**	**2**	**5**	**10**	**20**
平均正确率%	97.83	97.80	97.73	**97.87**	97.71
最高正确率%	97.90	97.86	97.84	**97.91**	97.01
最低正确率%	97.73	97.68	97.66	**97.79**	97.43

最终结果提升较小，没达到理想状态，个人认为是处理方式有一些问题。

对这个问题进行了分析，设置special key的权重为1后，输出`delta = P(sk|y=spam) - P(sk|y=ham)`，得到结果如下，其中`以后、公司、快乐、销售`四个词为对照，`以后`为出现在ham中较有标志性的对照，`公司`、`销售`为出现在spam中较有标志性的对照：

```
1234m    delta: 0.0004082942051386742
CSM2     delta: 0.0004374580769342938
ChinaHR          delta: 0.002245618128262708
EhooPost         delta: 0.012482137128525183
EhooSend         delta: 0.0024497652308320453
FoxMail          delta: 0.21140483874958194
Foxmail          delta: 0.17427293543624592
Internet         delta: -0.00804098224476444
JMail    delta: 0.0009915716410510659
JiXing   delta: 0.0010790632564379246
Lotus    delta: -0.0064341931406905264
MIME     delta: -0.023982404918611377
Microsoft        delta: 0.10505748431677903
Open     delta: -0.0049405411616016545
SinaMail         delta: -0.003008551515029129
VolleyMail       delta: 0.04812038846277232
XMail    delta: -0.003102200264261504
XiaoYan          delta: 0.0012248826154160227
```

```
afternoon        delta: -0.009426303851794215
deep_night       delta: 0.018583385979061617
http     delta: 0.656606626480347
iPlanet          delta: -0.05767794565404722
jpfree   delta: 0.0004666219487299134
morning          delta: 0.014832898997942462
night    delta: -0.017209701773852504
noon     delta: -0.00678027935135736
phone    delta: 0.609967195410773
priorty1         delta: 0.006459014348288165
priorty2         delta: 0.23747734013486585
priorty3         delta: 0.16679451752651436
qmail    delta: 0.0004374580769342938
以后     delta: -0.10589335502730893
公司     delta: 0.5523104638053657
销售     delta: 0.13480000606431608
```

由对照可假设，$|delta| \geq 0.1$，该关键词对结果起到影响。由此，可看出仅有`X-Priorty`这个Feature中的几个关键词能够对结果产生影响，我之前设置的邮件中是否出现号码的关键词`phone`和是否出现网址的关键词`http`的delta值远远大于其他feature。由此，可以看出，`X-Priority`等`Special Features`在两种邮件中出现概率delta值相差实际不大，对结果起到的作用较小，而按关键词处理后其作用更是微乎其微，需要适当设置权重或者更改处理方式。

3. 实验结果

bias选取测试

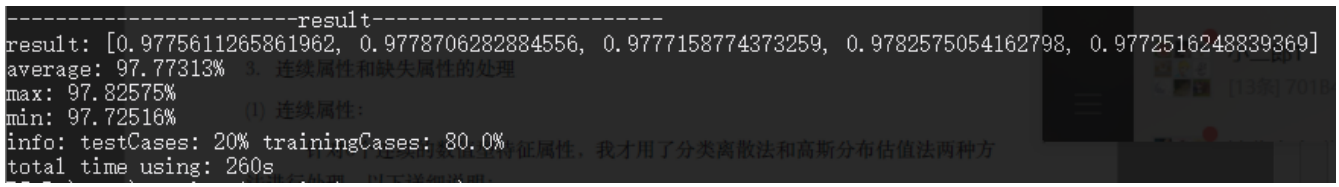
> 以下数据均在测试集100%选取，五折交叉验证情况下的得到：

bias	10	5	0	-5	-6.25	-7.5	-10
平均正确率	93%	95%	97.52%	97.72%	**97.84%**	97.70%	97.65%

训练集大小测试

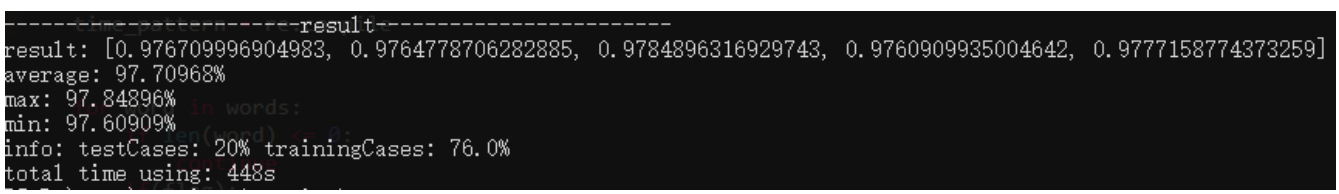
> 以下数据均在bias=-6.25情况下,使用五折交叉验证得到

训练集大小为100%



平均准确率 97.77%，最高准确率97.83%，最低准确率97.73%

训练集大小为95%



平均准确率为97.67%，最高准确率为97.85%，最低准确率为97.61%

训练集大小为50%

```
------------------------result------------------------
result: [0.9756267409470752, 0.9765552460538532, 0.9749303621169917, 0.9732281027545652, 0.9771742494583721]
average: 97.55029%
max: 97.71742%
min: 97.32281%
info: testCases: 20% trainingCases: 40.0%
total time using: 151s
```

平均正确率为97.55%，最高准确率为97.72%，最低准确率为97.32%。

训练集大小为5%

```
------------------------result------------------------
result: [0.9682760755184153, 0.9686629526462396, 0.9657226864747757, 0.9707520891364902, 0.9705199628597957]
average: 96.87867%
max: 97.07520%
min: 96.57226%
info: testCases: 20% trainingCases: 4.000000%
total time using: 68s
```

平均正确率为96.88%，最高准确率为97.08%，最低准确率为96.57%。

分析

在训练集为100%时达到最佳效果。

SPECIAL FEATURE测试

> 该测试在训练集100%，bias=-6.25情况下，使用五折交叉验证进行：

直接将SPECIAL FEATURE作为关键词加入字典：

```
------------------------result------------------------
result: [0.9775611265861962, 0.9789538842463633, 0.978644382544104, 0.9772516248839369, 0.9790312596719282]
average: 97.82884%
max: 97.90312%
min: 97.72516%
info: testCases: 20% trainingCases: 80.0%
total time using: 282s
```

平均正确率为97.83%，最高准确率为97.90%，最低准确率为97.73%，比没加之前有所提高。

提高SPECIAL FEATURE权重

weight = 10

```
------------------------result------------------------
result: [0.979108635097493, 0.9778706282884556, 0.9790312596719282, 0.9789538842463633, 0.9784896316929743]
average: 97.86908%
max: 97.91086%
min: 97.78706%
info: testCases: 20% trainingCases: 80.0%
total time using: 315s
```

weight = 5

```
------------------------result------------------------
result: [0.9768647477561127, 0.9773290003095016, 0.9770968740328072, 0.9765552460538532, 0.978412256267409
average: 97.72516%
max: 97.84122%
min: 97.65552%
info: testCases: 20% trainingCases: 80.0%
total time using: 316s
weight: 5
bias: -6.25
```

weight = 2

```
result: [0.9775611265861962, 0.9784896316929743, 0.9785670071185392, 0.9785670071185392, 0.9767873723305478]
average: 97.79944%
max: 97.85670%
min: 97.67873%
info: testCases: 20% trainingCases: 80.0%
total time using: 308s
weight: 2
bias: -6.25
```

bias	10	5	0	-5	-6.25
平均正确率	93%	95%	97.52%	97.72%	**97.84%**

weight=20

```
----------result----------
result: [0.9759362426493345, 0.9776385020117611, 0.980114515629836, 0.9776385020117611, 0.9743113587124729]
average: 97.71278%
max: 98.01145%
min: 97.43113%
info: testCases: 20% trainingCases: 80.0%
total time using: 329s
weight: 20
bias: -6.25
```

4. 其他思考

在实际分类过程中，垃圾邮件的分类正确率（约99.1%）总是大于正常邮件的分类正确率（约95.5%），结果如下：

```
96% corect count: 12149  false count: 259  spam corect rate: 0.9907486305538649  ham corect rate: 0.956355831147
97% corect count: 12273  false count: 264  spam corect rate: 0.990719537182114  ham corect rate: 0.955896226415
98% corect count: 12400  false count: 266  spam corect rate: 0.9908202193609918  ham corect rate: 0.955820476858
99% corect count: 12527  false count: 268  spam corect rate: 0.9909144542772861  ham corect rate: 0.955787037037
```

猜测原因如下：

- 数据集影响：
  - 数据集中，垃圾邮件数为42854，正常邮件数为21766，垃圾邮件接近正常邮件两倍，导致模型中正常邮件信息不足。
- 这是垃圾邮件分类问题固有难点：
  - 垃圾邮件中标志性词语较多，如 `购买`、`销售` 等，容易分辨，而正常邮件标志性词语较少，没有能够直接辨别的词语，且有可能会出现垃圾邮件中相对集中的词语。
  - 实际使用中，清华邮箱实际分类结果好像也经常将正常邮件分为垃圾邮件，而垃圾邮件很少漏过。

5. 实验总结

本次实验使用朴素贝叶斯对垃圾邮件进行分类，算法实现难度不大，但是一些参数的调整花了很长时间，让我回想起了人工智能导论中一些实验的调参过程，越发觉得在人工智能领域参数十分重要。本次实验中还有一些诸如*4.其他思考*中的问题需要探究，但时间有限，没有通过实验来确定原因。

总的来说，这次实验比较有趣，也不是很困难。谢谢老师和助教的帮助！

6. 程序说明

运行时请将trec06c-utf8文件夹与放置在源代码文件夹src内

使用python3运行`bayes.py`，运行时先输入一个小数，表示使用训练集总大小的多少倍进行训练。

`bayes.py`中有一个全局变量`bias`用于调整偏置项,默认为-6.25，有一个全局变量`special_key_weight`用于调整`SPECIFIC FEATURES`的权重，默认为10。