朴素贝叶斯分类器

赵晴岳 2019010974

> **建议先看`README.md`**

一. 实验目的

- 实现一个朴素贝叶斯分类器并在真实数据上测试；
- 完整地实现，**评估**、**分析**使用机器学习算法处理数据的全流程；
- 针对**邮件**数据的特殊性改进分类方法的实现细节。

二. 实验内容

编写朴素贝叶斯分类器，用`trec06p`数据集进行训练和测试，并至少探讨以下问题

- 训练集规模对评价指标的影响
- 概率计算中平滑化程度对测试效果的影响
- 特征工程的使用（除词袋之外的其他角度）

三. 实验原理和设计

基本方法一定是基于**词袋**模型的朴素贝叶斯方法，另外应实验要求需使用手工指定的特征（即**特征工程**）。

那么我们需要终点设计以下三个环节

1. 如何训练：词袋模型的**参数估计**
2. 如何预测：贝叶斯**决策理论**
3. 特征工程：如何将手工指定的特征**融合**到词袋模型中

参数估计

如果一封邮件的标签为y，并且其正文的文本的长度为n，依次为$x_1,\dots,x_n$，则根据贝叶斯公式以及**条件独立性假设**，得

$$P(y|x_1,\dots,x_n)\propto P(y)\prod_{i=1}^{n}P(x_i|y)$$

其中$P(y|x_1,\dots,x_n)$为该邮件属于类别y的概率，在本实验中，y只有两种取值：$True$（正样本，即垃圾邮件*spam*）或$False$（负样本，即正常邮件*ham*）。$P(x_i|y)$为类别y中x_i出现的概率。上式中尤为需要注意的是：n为**文本的长度**

> 也就是说，长为n的文本中，可能出现重复单词；比如x_1和x_n恰好都是`"free"`。

本实验中$P(y)$就是正负样本在训练集中的比例，记$\#\{y=c\}$为训练集中类c的数量，$\#\{y\}$为训练集的样本量，则

$$c=[False,True]$$
$$P(y=c)=\frac{\#\{y=c\}}{\#\{y\}}$$

$P(x_i|y)$则可以通过频率来估计，记$\#\{y=c,x_i=k\}$为类别c中x_i出现的**次数**

$$P(x_i = k|y = c) = \frac{\#\{y = c, x_i = k\}}{\#\{y = c\}}$$

参数估计会遇到以下现实问题

- 单词特征向量是人为选出（此过程见第四节的`四.2全局数据清洗&特征提取`）的，比如特征向量为`x=['!', '$', 'free']`，看训练集中所有负样本（ham，下同）恰好没有出现过`'$'`，则对应概率会变为0，即：*出现过美元符号的邮件不可能是负样本*，这显然**不**合理，需要引入平滑机制，本实验源码中的`ALPHA`即解决此问题的平滑因子，其功能详见第四节`四.3变量选择`。
- 本次实验要求特征工程，那么手工（hand-crafted）特征的使用是否需要融合进$P(y|x_1, \ldots, x_n)$的计算中？本项目确实**融合**了。
- $P(y|x_1, \ldots, x_n)$计算公式中出现连乘，可能导致浮点数系统下溢，并且乘法速度不快；鉴于自然对数单调增（保序），本项目将公式中的乘法转为**取对数再相加**——————那就意味着平滑处理在所难免（`np.log(0)`会报警告）了。

决策理论

依贝叶斯决策理论，对于某条单词数为n的样本$\{x_1, \ldots, x_n\}$的输出$\hat{y}$为其最大后验概率（MAP）估计

$$\hat{y}_{MAP} = argmax_{y \in \{False, True\}} P(y|x_1, \ldots, x_n)$$

特征工程

本项目特征工程有三个特点

1. 特征的参数计算非常容易：通过多种**二值化**手段简化参数估计过程
2. 特征可融合进贝叶斯分类器：换言之，在预测阶段，这些特征**并不是**基于规则的，可以通过其先验概率对朴素贝叶斯的最大后验概率做出贡献
3. 特征对垃圾邮箱有较强特异性

本项目共选定了**4**个特征，其中

- 前三个进行了**基于知识的二值化**。
- 最后一个是数值特征，**在预测阶段**基于数据进行二值化。
  换言之，四个特征最终贡献到后验概率时，分别对应到四个伯努利（Bernoulli）分布的的参数的交叉熵之相反数。
  形式化如下：

  如果二值化特征f在类别c中为`True`的概率的极大似然估计为p_f，那么类别c的**对数**后验概率中就**加**上

  $$f \log(p_f) + (1 - f) \log(1 - p_f)$$

四. 实验流程

1. 探索性数据分析

对于充满**噪声**且具有**结构**的数据，端到端并不是永远的神，EDA对于了解数据结构和特征选取十分重要。

数据集基本情况

整个数据集共有37822封邮件，其中正负样本并不平衡：spam有24912封，ham只有12910封，垃圾邮件几乎达到了普通邮件的两倍，也许这个比例与日常的经验不符，但也正好使朴素贝叶斯分类器倾向于将一封特征不明显（i.e.极大似然估计下属于垃圾或正常邮件的概率几乎相同）邮件判为垃圾，进而提高查全率（recall）。邮件文本含有不少噪声，比如日文、俄文等无法通过`encoding='utf-8'`解码的样本，事实上这也可以算一种特征（详见本小节第2部分“特征提取”）；邮件样本大多为几KB到几十KB不

等，但也有很大的邮件，比如`../data/109/116`的邮件几乎达到了**8MB**之大，有趣的是，正是这封最大的邮件，指明了本项目特征提取和变量选择的方向。

在全局数据探索的过程中还发现，伴随这大邮件的常常是很多形似哈希的乱码——这些乱码可以被`'utf-8'`解码，也确实是正常的ASCII码，仅仅是人类**不可读**而已。

> 如果直接对原始文本清洗分词，统计词频，那么这些人类不可读的字符串势必成为难以逾越的干扰——至少在可解释性方面是干扰——经过细致调研，我终于搞清了这一切的由来：邮件**不是纯文本**，邮件的**收发格式**和**格式中各字段的一致性**有严格的**协议**加以约束！

邮件的数据结构

直观起见，我们不一上来就介绍邮件的协议和格式结构，而是举例。就以全数据集中*最大*的邮件为例：此文本中包含了大量人类不可读的合法ASCII字符，比如`'/9j/4Rv+RXhpZgAASUkqAA'...`，而我们只需解析此邮件就可恍然大悟……

应用python标准库`email`

```
import email
from email.iterators import _structure
f = open('./trec06p/data/109/116')
msg = email.message_from_file(f)
_structure(msg)
```

得到分析树：

```
multipart/mixed
    multipart/alternative
        text/plain
        text/html
    image/jpeg
    image/jpeg
    image/jpeg
    image/jpeg
    image/jpeg
    image/jpeg
```

当代的多部份（multipart）邮件都是基于最基本的RFC 822扩展而来，例子中的邮件结构就是典型的"multipart message"，它的格式服从RFC 6838——具有典型的**树**结构，其中的`multipart/mixed`和`multipart/alternative`一定是内部结点，而此例中叶子结点包括一份纯文本，一份具有HTML格式的文本以及6份分立的图片附件，它们按照协议格式经ASCII编码后，就形成了上文提到的所谓"形似哈希"的内容。

按照MIME types的规定，一封多部份邮件的邮件体（**body**）主要有两种基本**内部**节点（`multipart`和`message`）和十种基本**叶子**结点（如上例中的`text`、`image`）。MIME types的存在赋予了邮件"多媒体"的功能，但也增大了邮件解析和特征提取的难度，具体来说：

- 朴素贝叶斯分类器的判别词汇原则上只统计**body**中基本类型为`text/^.*$`的部分的词汇，邮件头（**headers**）中的信息和多媒体部分的信息需要**单独**处理！
- 对于`text/html`类型的部分，**为防止**`<`、`<p>`**等超文本标识干扰朴素贝叶斯分类器**，等应解析HTML，朴素贝叶斯分类器只统计其中的纯文本，而`<span>`、`<html>`之类的超文本标识作为一种特征**单独**处理！
- 上文举例中并未展示headers，但邮件头因其格式的规定严格，很多字段之间存在**一致性约束**，故成为特征提取的重点（见文末Fun facts）；为了通用性，本项目会更多地关注协议格式中mandatory的字段之间的关系。

2. 全局数据清洗&特征提取

> 原则上，如果数据清洗的过程中用到了该数据集本身的数字特征（数据驱动），那么这种数据清洗应**只在训练集**上进行，或者清洗本身要加入交叉验证；但如果数据清洗完全是基于先验知识的，就可以在整个数据集上做。本项目的全局数据清洗完全基于知识。

数据清洗&特征提取的成果是将**原始数据**转换成样本矩阵（`pandas.DataFrame`）。本项目的样本矩阵设计了以下字段

- `label`：`bool`型字段，`True`表示正样本（垃圾邮件），`False`表示负样本。
- `utf-8`：`bool`型字段，如原始数据能被`utf-8`正常解码，则为`True`
- `html`：`bool`型字段，邮件的邮件体`text`基本部分中是否含有HTML，有则为`True`
- `consistent`：`bool`型字段，衡量邮件头的一致性，若有潜在自相矛盾则为`False`
- `mess_index`：`int`型字段，表示邮件头中出现特殊或奇异字符（邮件头中不应常出现的字符）的数量。（此字段很稀疏，很多邮件此项为0）
- `body`：`str`型字段，该邮件的body中全部`text`基本部分的所有英文单词的拼接（单词间以空格分隔）

如果通过`utf-8`打开一封邮件并解析时遇到乱码，则忽略乱码，并对应的`utf-8`字段记为`False`，如果`text`基本部分中含HTML，则解析HTML，只保留纯文本内容，对应的1`html`字段记为`True`。根据先验知识，正文中的`'$'`、`'!'`具有一定判别力，故保留；除此之外，含空白符在内的所有表颠覆哈全部视为分隔符对正文进行切分，并剔除所有长度不超过3的单词；由于词干化（stemming）过于暴力，故考虑调用`nltk.WordNetLemmatizer`逐词做词形还原（词形还原**不改变大小写！**），最后根据停止词词库`nltk.corpus.stopwords`剔除停止词，并用空格连接各单词，存入矩阵。

根据协议，邮件头中的`From`和`Message-ID`字段都是强制要求的，如果邮件头中缺少这些字段，则`consistent`字段直接记`False`，另外，原则上`From`字段的域名应该和`Message-ID`的域名一致，如若不然，则`consistent`字段记为`False`。虽然`Subject`字段在协议中只是optional，但一封有用的邮件不可能没有主题（标题），故有邮件头没有`Subject`，对应的`consistent`字段也记`False`；如果邮件头的日期不合法，对应的`consistent`也记为`False`。

> `consistent`字段聚合了许多特征，看似丢失信息，但这符合朴素贝叶斯的假设：**如果将`consistent`字段中的各特征拆分开，那么这些特征一定不是条件独立的，并且相依性会很强**，这极可能降低朴素贝叶斯的表现；故本项目选择将这些一致性信息融合为一个特征！

一封正常客户端发出的邮件中，邮件头**除主题字段**外，不应该出现`'?'`、`'!'`、`'$'`，对于非垃圾邮件，主题中也不应该高频出现上述三个符号，故`mess_index`就统计邮件头中三个符号的出现频数。

另外值得注意的是：词形还原不改变单词大小写，但根据Better Bayesian Classifier的提示，单词大小写也许对垃圾邮件过滤有影响，故**是否将单词全部转为小写也是本项目中的一个`bool`型超参数。**

3. 变量选择（朴素贝叶斯）

很多单词不是停止词（stopwords），但它们对垃圾（spam）和非垃圾邮件(ham)的判别力很弱，这些词不应该出现在朴素贝叶斯的变量列表中。变量选择是**数据驱动**的，所以要在交叉验证阶段才做，且只对当前的训练集做。

> 换言之，对于一次完整的五折交叉验证，五次的变量列表不一定是一样的

变量选择两点注意事项：

1. 首先在变量选择时出现频率高的单词不一定合适，因为它在正负样本中的比例可能很接近，以至于没有判别力
2. 某些判别力极强（无限大）的词一定不合适：假设在训练集中某个单词w出现了一次，并且恰好那一次对应一封垃圾邮件，则根据训练集的情况，单词w的判别力无限大，毕竟训练集中出现了单词w就一定是正样本（没出现的不一定不是）。

针对以上两问题，下面给出一种方法一箭双雕。

首先回顾用于文本分类的一种朴素贝叶斯假设——这种朴素贝叶斯实际上假设了样本的**生成方式**：有一个假想的"生成器"，欲生成一个长度为n的文本，首先决定生成正样本还是负样本，这对应了正负样本的先验分布$P(y)$，而正负样本分别有一个分类分布（Categorical distribution），对于文本中的每一个位置，独立的从对应的categorical distribution中抽出一个单词放上去。在上述过程中，如果以某一个单词w为主体，那该categorical distribution就对应了单词w的一个二项分布（binomial distribution），而单词w所对应的二项分布在正负样本中可能是不同的，朴素贝叶斯分类器正是利用了这种（假设中的）不同；所以直观上正负样本中的二项分布差异越大，单词w的判别力就越强，那么方法就呼之欲出了：

$$\text{依相对熵（Kullback-Leibler divergence）排序}$$

可是考虑到有的单词w在某一类中出现了0次，导致其在两类间的相对熵为正无穷，我们必须使用课件中的光滑化。依平滑化后二项分布的相对熵降序排列，去前N个作为朴素贝叶斯的变量即可。

> 计算相对熵是需要注意实际意义：相对熵是**非对称的**，我们找的是"明显不像是正常邮件中出现"的单词，所以计算$D(p||q)$时，p为负样本的二项分布参数，即以负样本为参照物，看那些单词在正样本的出现的频率与负样本中明显不同。

4. 模型编写

提交的代码可以直接从原始数据开始执行，具体的**使用**（复现）**过程**和项目的架构见`README.md`

编写时唯一需要注意的细节就是——手工指定的特征全部对应到二项分布的参数，而二项分布的对数似然函数是**交叉熵函数的相反数**，故在计算概率时直接加上四个交叉熵函数的相反数即可。

五. 评估与分析

0. 基本方法

超参数共有**5**个，那就……grid search。

> **超参数**:
>
> ```
> # 是否将文本中所有单词转为小写
> LOWERCASE = {
>  'default': False,
>  'grid': np.array([
>      False, True
>  ])
> }
> # 词袋模型的预期变量个数
> N = {
>  'default': 15,
>  # The default value of N=15 is from [Better Bayesian Filtering]
> (http://www.paulgraham.com/better.html)
>  'grid': np.logspace(start=0, stop=3, num=4, dtype=int)
> }
> # 训练集的使用比例
> PERCENTAGE = {
>  'default': 1,
>  'grid': np.array([
>      0.05, 0.25, 0.5, 1
>  ])
> }
> # 概率平滑化因子
> ```

```
ALPHA = {
 'default': 1.0,
 'grid': np.logspace(-2, 4, num=7)
}
# 是否使用手工指定的特征
CRAFT = {
 'default': True,
 'grid': np.array([
     False, True
 ])
}
```

> 变量个数默认为15的设定来自Better Bayesian Classifier

1. 默认参数下对变量的感性认知

首先在**默认**参数下进行一次交叉验证，由于是五折，故产生了五个变量列表如下，其中的单词都是按照相对熵降序排列的。

```
['!', '$', 'DMDX', 'arizona', 'Vulnerable', 'psych', 'ucsb', 'Received',
'CRUST', 'list', 'node', 'From', 'Subject', 'pdfzone', 'RPCSS']
['!', '$', 'DMDX', 'Vulnerable', 'arizona', 'psych', 'Received', 'ucsb',
'CRUST', 'list', 'node', 'From', 'RPCSS', 'file', 'Subject']
['!', '$', 'DMDX', 'arizona', 'psych', 'Vulnerable', 'Received', 'CRUST',
'node', 'list', 'From', 'ucsb', 'Subject', 'file', 'pdfzone']
['!', '$', 'Vulnerable', 'ucsb', 'node', 'RPCSS', 'list', 'pdfzone', 'Received',
'file', 'Patched', 'pnfs', 'DMDX', 'starship', 'From']
['!', '$', 'DMDX', 'arizona', 'Vulnerable', 'psych', 'ucsb', 'CRUST', 'node',
'Received', 'From', 'RPCSS', 'file', 'Subject', 'list']
```

先验知识的正确性得到了确认，五个列表中排在前面的都是`'!'`和`'$'`排在前二，且叹号的相对熵比美元符号高。

另外，上面的五个列表中，`'DMDX'`、`'arizona'`和`'RPCSS'`都出现了多达**4**次，`'ucsb'`更是出现了**5**次；经过人为信息检索，得知DMDX是亚利桑那大学于2003年发布的一种心理学编程软件，可见这个**数据集本身的偏见不小**。

2. 超参数搜索结果

如前所述，整个网格搜索跑了$2 \times 4 \times 4 \times 7 \times 2 = 448$次，加上第一次默认参数的过程；共计进行了$449$次交叉验证，结果见`./trec06p/inter/records.csv`。

默认参数的表现很一般，只取得了83.29%的`accuracy`和87.49%的`f1_score`

最优结果与预期并不完全相符，预期中，根据Better Bayesian Filtering的结论，N=15时效果好，并且把所有单词都转为小写不利于提高准确率；根据理论知识，训练集的规模越大，估计量的方差应该会减小，得出的概率估计数值也应该越准；但是最优结果打破了以上三条论断。

如下超参在`accuracy`和`f1_score`两项指标上**均取得了第一**：

LOWERCASE	N	PERCENTAGE	ALPHA	CRAFT	accuracy	precision	recall	f1_score
True	1000	0.05	100	True	91.16%	94.43%	92.01%	93.20%

最优结果中的`N`比较符合理论，毕竟使用的特征越多可利用的信息也就越多，`CRAFT`也比较符合预期，手工特征可以提高准确率，若只把上述最优参数中的`CRAFT`改为`False`，则`accuracy`降为88.66%，`f1_score`降为91.21%。

对两个`bool`型超参的探究

是否将所有单词转为小写，以及是否使用手工特征是本项目的两个`bool`型超参。

> 下两表中的"最优值"指此次网格搜索全部449次中**在给定条件下**的最优值，"平均值"同理。（下同）

`LOWERCASE`	`accuracy`**最优值**	`accuracy`**平均值**
`False`	90.58%	75.54%
`True`	91.16%	75.75%

`CRAFT`	`accuracy`**最优值**	`accuracy`**平均值**
`False`	88.66%	68.60%
`True`	91.16%	82.66%

无论是最优值还是平均值，都显示：有手工特征时效果更好，将所有单词都转为小写效果更好。

`LOWERCASE`的实验结果与直觉不完全一致，唯一合理的解释大概是：比如`FREE`、`Free`、`free`等单词，它们出现的概率**不是条件独立的**，所以聚合在一起更符合朴素贝叶斯的基本假设。

查准率和查全率的探究

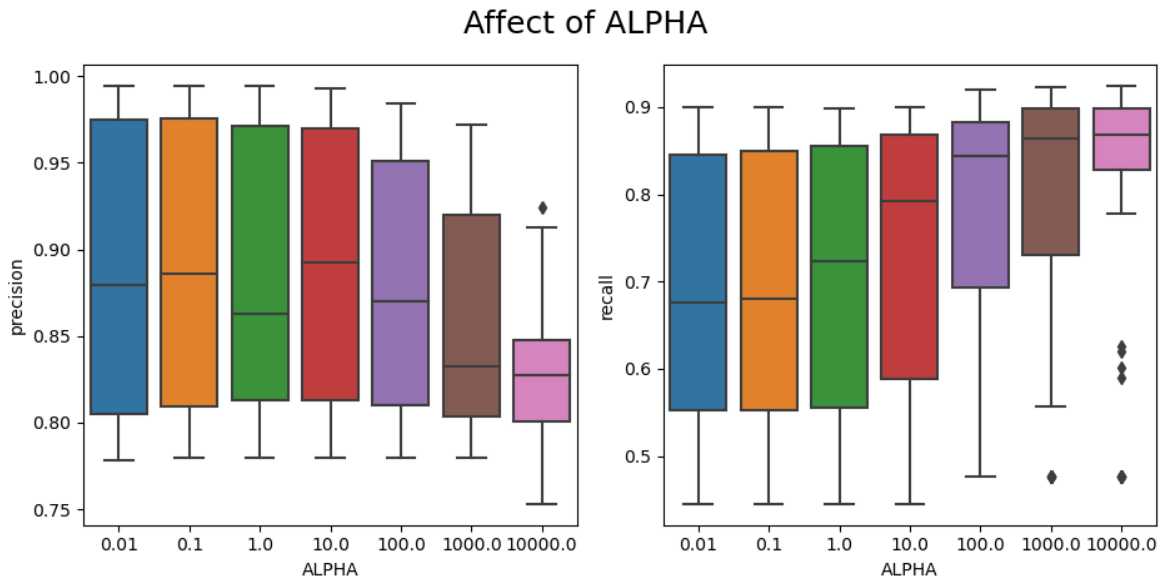
我们知道，`precision`和`recall`之间存在权衡（trade-off），而作为两者的调和平均，`f1_score`更能反映模型查准率和查全率的综合评估；但具体到邮件分类：

垃圾邮件过滤需要极低的FDR（假发现率）

$FDR + precision = 1$，所以模型最好具有很高的`precision`，如果我们按照`precision`排序，得到最优`precision`所对应的超参：

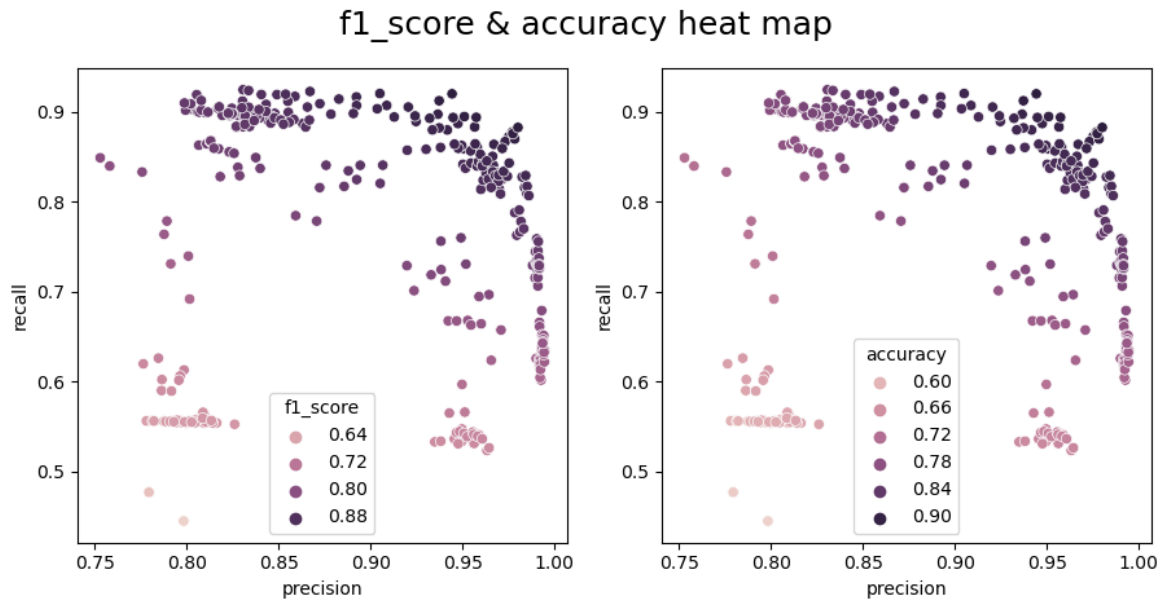
`LOWERCASE`	`N`	`PERCENTAGE`	`ALPHA`	`CRAFT`	`accuracy`	`precision`	`recall`	`f1_score`
`True`	1000	0.5	0.01	`False`	75.71%	99.45%	63.49%	77.49%

正如预期，几乎完美的`precision`带来的是较低的`recall`，同时`accuracy`也一般；而且不使用手工特征有助于提高`precision`，这说明有些正常邮件的邮件头也不是很整洁。另外我们注意到，`precision`**最优时**`ALPHA`**取值很小**，这启发我们对`ALPHA`的作用做进一步的探究如下图：



在本项目网格范围内可以看到，`ALPHA`的与查全率在**一定范围内**正相关，与查准率**在一定范围内**负相关；换言之，如果平滑过度（**over-smoothing**）那么假发现率就会升上来！而这对垃圾邮件过滤是致命的。

同时我还验证了一下查准率和查全率的权衡关系，如下图：



我们看到，对于相近的或相同的`f1_score`，查准率和查全率之间确实存在权衡（这是理所当然的），另外我们还注意到，上图中的`f1_score`和`accuracy`的"步调"非常一致，于是我们计算了`f1_score`和`accuracy`之间的**相关系数**ρ

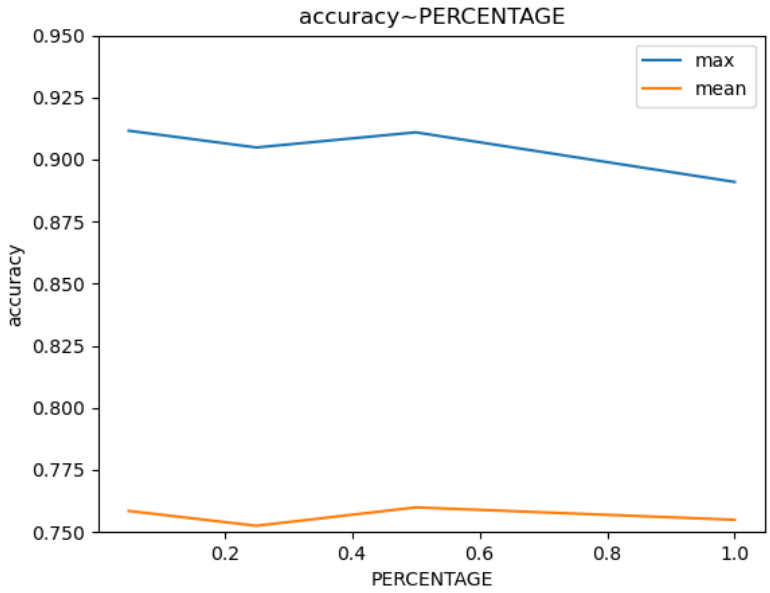
$$\rho(f1\_score, accuracy) \approx 0.9890$$

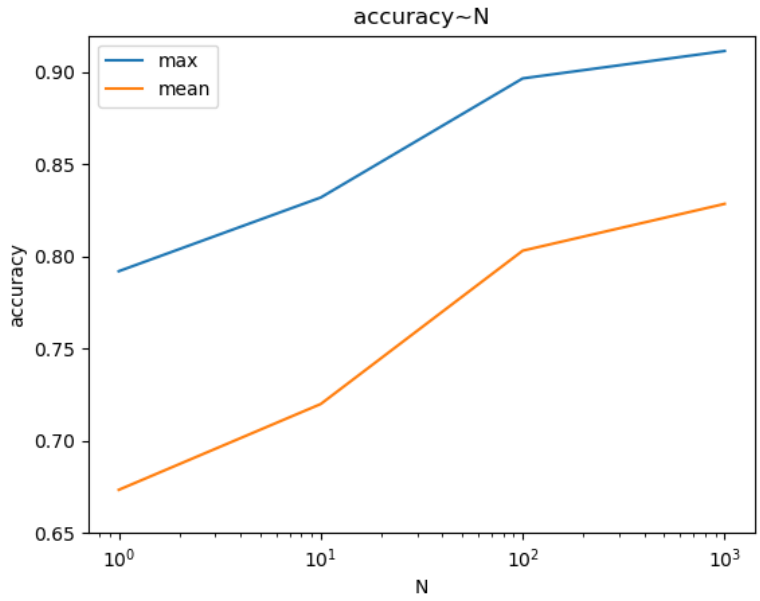
然而这既不是偶然也不是必然——本次作业中`f1_score`和`accuracy`的相关系数如此之高，是因为**样本并非特别不平衡**，虽然正样本几乎是负样本的两倍，但这还在可接受范围之内。如果$\#spam/\#ham \gg 10$，那么`f1_score`和`accuracy`就不会那么一致了。

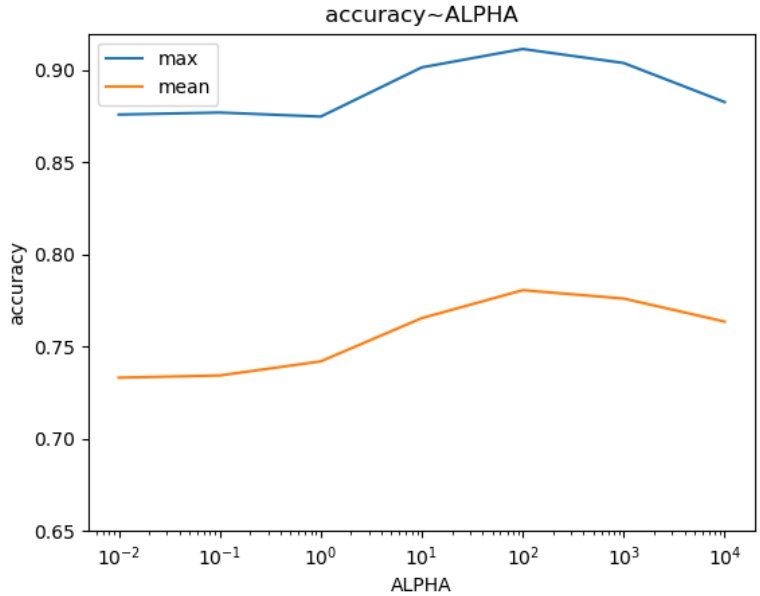
对其余三个超参数的探究

如下图所示，

> "最优值"指此次网格搜索全部449次中**在给定条件下**的最优值，"平均值"同理。







`PERCENTAGE` 的情况有些出乎意料，模型对训练集使用比例 `PERCENTAGE` 并不敏感，不知道是不是因为有些判别力强的词无论在什么范围内都判别力强？

`N`的情况很正常，在一定范围内，增加朴素贝叶斯的变量个数可以增强判别力，并且也直接否定了Better Bayesian Classifier中直接指定N=15的做法。

`ALPHA`的情况也很正常，平滑不足和过度平滑都会导致准确率下降。

六. 总结及项目结构说明

总结

- 理论优雅的模型在实现中会遇到**数值计算**方面的若干问题，比如本实验中
    1. 计算后验概率时若连乘则易发生浮点数**下溢**。
    2. 若特征选择较多则概率的分布十分稀疏，计算速度也可能被拖慢。
    3. 本次实验中，经过预处理的数据集并不大（full batch也只有40+MB），可以**全部读入内存**进行训练，但若训练集样本量再扩大许多倍，恐怕就不能这样做了。
    4. 本实验中的第四个手工特征`mess_index`是一个极为**稀疏**的特征，在共计**37822**条样本中，有**19162**条的`mess_index`为`0`！而为了便于处理，本项目将此特征也作为数据表中的一个普通特征对待，并未使用`scipy`的稀疏矩阵存储；也许对于更大的数据集，对稀疏特征的高效存储和处理势在必行。
- 这次作业本质是处理数据。虽然核心是概率计算（也即朴素贝叶斯的参数估计）；但是对邮件数据结构的理解程度和数据预处理的合理性都极大地影响着实验结果，对邮件的**协议**的了解也有利于提高预测的准确率。
- 前人的文献，即使是经典文献也不能全信，比如课上提到的Better Bayesian Classifier中，选用最显著的15个单词作为变量，但是本次实验实测发现15个变量在此数据集上效果远非最佳；这篇文献还提出了不应词干化以及不应将所有字母转为小写，但本次实验实测就发现全部转为小写后一些指标更高。

关于项目

> 关于复现，见`README.md`

七. Fun Facts

应本次实验的要求，此项目确实用到了邮件体（body）的信息，而有趣的是，一篇发表在ICCNS 2019的文章Recognizing Email Spam from Meta Data Only提出了一种仅基于*metadata*的分类器，并在CSDMC2010数据集上取得了SOTA的成绩。正如这篇文章所指出的那样，有经验的人只看邮件头就可以很有把握地识别垃圾邮件，metadata蕴含了很强的*判别*信息，而本次实验也一定程度上印证了这一点。