机器学习概论 Exp1 Report

计52 路橙 2015010137

本次实验我实现了朴素贝叶斯对垃圾邮件的分类，最终可以在我随机划分的数据集上达到99.48%的准确率。本文档对实验原理进行分析，并对实验结果中的一些进展和困难进行了讨论。

源代码设计

dataset.py

1. `read_label()`：读取`label/index`文件，得到每个文件的`label`
2. `divide_set()`：划分训练集与测试集。具体划分规则如下：
   - 对于`data_cut`中的每一个**文件夹**，其中的文件以**90%**的概率被选择到训练集中，否则被选择到测试集中。选取的方法是蒙特卡洛方法，即循环到某个文件时，调用`random.random()`，若返回结果小于`0.9`，则将该文件路径加到训练集中，否则加到测试集中。
   - 最终结果在`dataset/trainset`和`dataset/testset`中。

train.py

1. 对每个训练集中的文件，以`sample_rate`的概率选中它作为训练数据（`issue 1`相关）。
2. 对于每个已分词文件，用如下规则找到每个需要被考虑的x_i：
   - `regex = u'[\u4E00-\u9FA5]+'`，表示只匹配中文；
   - `regex = u'http'`，表示匹配`'http'`，用于`issue 3`；
   - `regex = u'From.*@.*'`，表示匹配头部`From`中的发件人。然后通过如下代码找到发件人的域名（即@后面的域名）：

     ```
     mails = from_text[0].split('@')[-1].split('>')[0].split(' ')[0]
     ```
3. 最终，通过计算已知y的条件下x_i的频数，得到所需的所有频数，用于`test`阶段的计算。之所以不直接计算$p(x_i|y)$，是因为考虑到`issue 2`的平滑处理。
4. 训练出来的原始数据保存到`train_result`文件夹下。

test.py

1. `test()`函数用于计算概率。其中，关于`issue 2`的平滑处理及关于`issue 3`的feature处理在之后详细讨论。
2. 为了避免浮点误差的累积（因为各个数据的概率都很小），因此需要对计算过程取`log`进行计算。
3. 预测的y为：

$$\text{Output } \hat{y} = \text{argmax}_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

Issues 讨论

issue 0. 浮点误差

在计算连乘时，发现大部分数字的概率很小（甚至出现**1e-6**的概率），因此若使用浮点进行连乘，若一封邮件中字词较多，会很容易导致浮点运算的溢出导致预测失败。因此，我们需要对运算过程中取`log`，并对概率的连乘变成连加。

在没有转化为`log`时，我的预测准确率只有**56%**左右。而实现了转化`log`后，在没有考虑`issue 2`之前，我的预测准确率也达到了**81%**.

issue 1. 训练集大小的影响

对于已经划分好的一个训练集，在训练中以`sample_rate`的概率取数据进行训练（详见**源代码设计/dataset.py**中的描述）。

不实现laplace平滑时：

当不实现`issue 2`中的平滑处理，即把概率为0的词直接跳过时，得到如下结果：

```
sample 5%:
[0.4039664378337147, 0.3528604118993135, 0.34782608695652173, 0.3816933638443936,
0.350419527078566]
average: 0.36735316552250186
min: 0.34782608695652173
max: 0.4039664378337147

sample 50%:
[0.7691838291380626, 0.7725400457665904, 0.7496567505720824, 0.7565217391304347,
0.7525553012967201]
average: 0.760091533180778
min: 0.7496567505720824
max: 0.7725400457665904

sample 100%:
[0.8096109839816934, 0.8096109839816934, 0.8096109839816934, 0.8096109839816934,
0.8096109839816934]
average = mim = max = 0.8096109839816934
```

实现laplace平滑时：

在测试时，我实现了`issue 2`中的laplace平滑，并设置`alpha = 1`（关于公式的描述见`issue 2`）。经过测试，得到如下结果：

```
sample 5%:
[0.9510297482837529, 0.9490465293668955, 0.9522501906941266, 0.9527078565980168,
0.9487414187643021]
average: 0.9507551487414189
min: 0.9487414187643021
max: 0.9527078565980168

sample 50%:
[0.9604881769641495, 0.9598779557589626, 0.9609458428680396, 0.9606407322654462,
0.9603356216628528]
average: 0.9604576659038901
min: 0.9598779557589626
max: 0.9609458428680396

sample 100%:
[0.9626239511823036, 0.9626239511823036, 0.9626239511823036, 0.9626239511823036,
0.9626239511823036]
average = min = max = 0.9626239511823036
```

分析如下：

- 当不实现laplace平滑时，可以明显地看到训练集的大小对结果的影响。当训练集越大时，朴素贝叶斯的准确率越高。且在训练集很小时，朴素贝叶斯的准确率甚至不如随机预测。这说明训练集的大小对朴素贝叶斯的准确率起到了很大的决定性作用。解释如下：
    - 一方面，当不实现平滑处理时，对未出现的词汇全部忽略，这会造成所有“正常”但在垃圾邮件中未出现的词被垃圾邮件忽视，他们一方面会在计算时降低正常邮件的计算概率，另一方面又不对垃圾邮件起作用，因而对正常邮件起到了反作用。
    - 另一方面，所有“垃圾”词汇没有在正常邮件中出现，也会造成对垃圾邮件识别的概率下降。
    - 从本质上来讲，这是因为朴素贝叶斯全部依赖于各个数据的“概率”的估计，通过频率估计概率，这要求频数达到一定的大小才可以保证，因此需要足够大的数据量才可以保证准确率。
- 当实现laplace平滑时，训练集的大小对结果虽然有一定的影响，也是训练集越大，准确率越高，但这个影响不再显著。这一点可以如下理解：
    - laplace平滑为每个不出现的数据分配了一个小概率，这可以让上面的前两点造成的影响全部消失，即“小概率但未出现”事件被模拟了一个小概率，从而导致它对总体的影响不再显著。

issue 2. 零概率

零概率事件在如下可能的情况下发生：

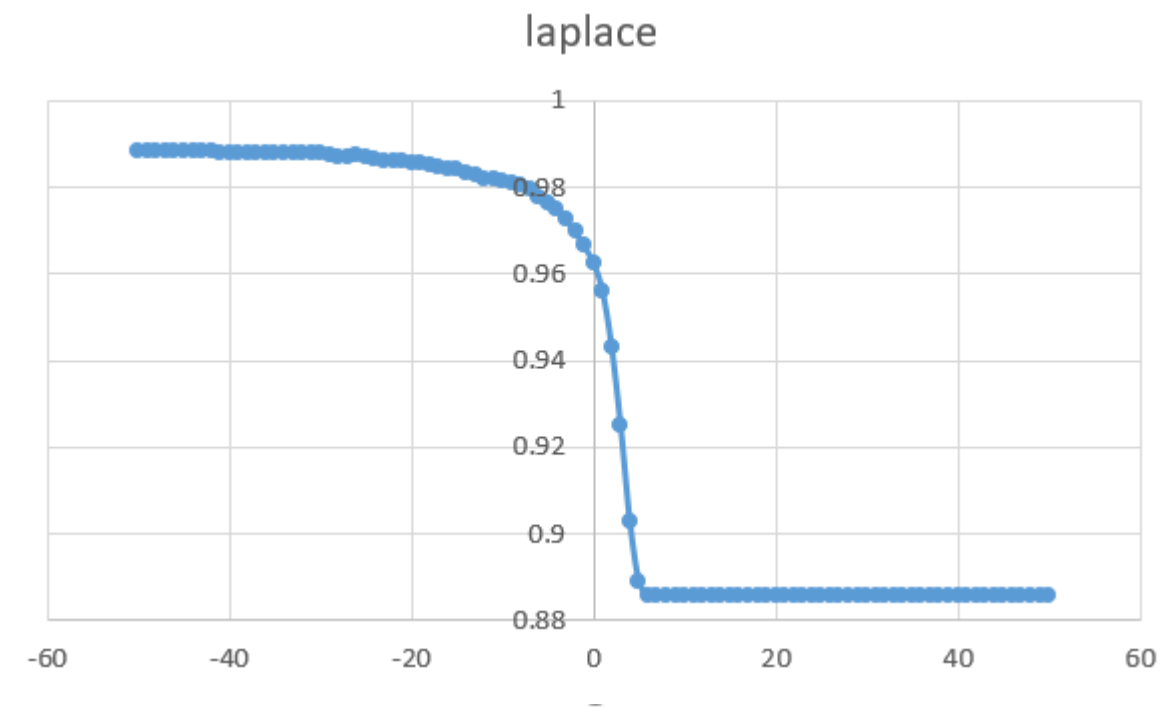
- 由于数据量有限，总有某些特定的词汇只在某个label中出现，因此在另一个label中该词出现的概率必定是0.
- 当测试集中有词汇不属于训练集时，在两个label中的概率都是零。

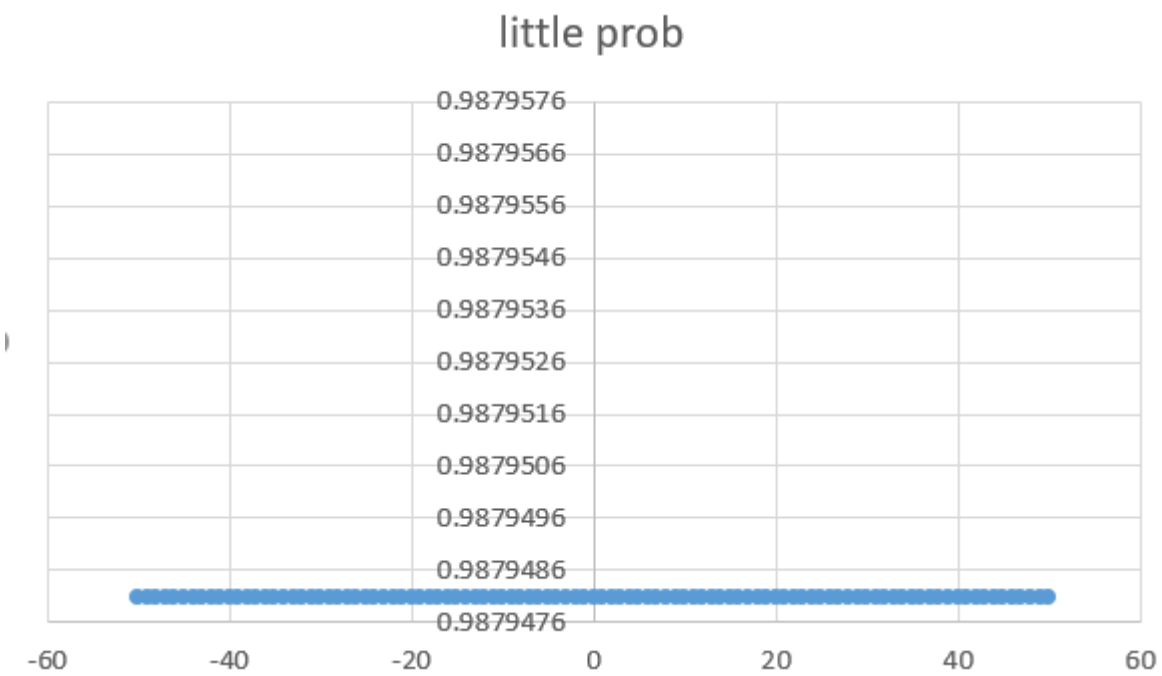
对于零概率事件，它直接会导致连乘的概率为零从而导致预测失败。因此，一种可能的做法是实现平滑处理。具体方法如下：

对于零概率事件，我分别实现了两种平滑处理：

1. Laplace平滑，即用$\frac{\alpha}{\#\{y=c\}+M\alpha}$估计频率，其中$\alpha$为一个参数，$M$为所有$x_i$的个数，$\#$表示频数。
2. 直接赋值为小概率，如`1e-50`。

对于这两种方法，我分别取参数α与小概率参数从`1e-50`到`1e50`，以`log-scale`，画出参数对准确率的影响如下图：





可以看到，拉普拉斯平滑在`1e-20`到`1e20`之间变化较快，且随着α的增大，平滑处理的效果会逐渐下降直至趋于缓和。因此，为了让拉普拉斯平滑的效果较好，建议α取较小的值。

而取小概率的平滑则对于常数小概率的取值不敏感，取任何值都可以有一个较好的稳定性和准确率。

issue 3. 特定特征的选取

对于特定特征，分析得出，有两种特征可以考虑：

1. `http`字段：在垃圾邮件中`http`的出现频率远大于正常邮件，因为有推销。
2. `From`字段有邮箱域名，垃圾邮件的域名往往出现在某些免费注册邮箱的域名中，且非公司域名。因此，这些域名在垃圾邮件中出现的概率也大于正常邮件。

具体的匹配规则在上面**源代码设计/train.py**中已有详细说明。

而经过训练的计算，发现的确满足如上两个特征：

1. `http`在垃圾邮件中出现了30111次，而在正常邮件中只出现了9419次。
2. `From`字段中几个典型域名：`163.com`在垃圾邮件中出现了5886次，在正常邮件中只出现了671次。此外，还有一些典型域名也有类似的特征。

在加了上面两个特征后，将`issue 2`中的`0.9879`的准确率提升到了`0.9913`，进一步提升了准确率。

由于如上两个feature具有很强烈的特征，因此我将他们的考虑权重扩大10倍，即在取`log`后相加时，将每个计算的结果乘以10（相当于原来连乘时将概率变成10次方），从而更大地提升了预测准确率，将上面的准确率直接提升到了`0.9976765799256505`。

总结

经过如上的讨论以及优化，我的代码最终在朴素贝叶斯上达到了很高的准确率。这说明，就算是最简单的方法，只要观察原理、细节优化、特征提取，也可以得到很高的性能评价。机器学习的魅力并不只是原理的提升，细节优化以及从原理分析可能的优化点同样是很重要的技能和理论。