机器学习概论实验报告

Introduction to Machine Learning

Exp1 - Naïve Bayes Classifier

李文哲 | 2017011468 | 2019.3.15-2019.3.28

问题描述

- **目标：**实现一个朴素贝叶斯分类器并测试其在真实数据集上的效果。数据集为中文邮件数据集，分类器需要从中检测出垃圾邮件。
- **问题导向：**
  - 实现机器学习算法，将其应用在真实数据集上并且改善其性能。
  - 合理的评价算法的表现。
  - 分析实验结果。

整体设计思路

整个程序的组织如上图。其中，数据预处理程序在 prepare_dataset.py 文件中，将从分词后的数据集中提取邮件的头信息（如优先级，发件人的一致性，收件人的一致性，发信时间等）、正文内容和标签。抽取特征词程序在 extract_feature.py 文件中，其主要作用是从正文内容中提取特征词，作为计算似然度的基准。训练集统计、计算似然度、做出预测均在 cross_validate.py 文件中，以便进行五折交叉检验。每一次划分完数据集后，程序会根据训练邮件类别分别统计相应的词频等特征。对于每一个测试样例，会根据其特征词的频率等特征以及先验概率计算似然度，以此预测邮件类型。最后程序会统计准确率等参数。

语言&开发工具

Python + VSCode

算法原理

基于朴素贝叶斯方法：$\hat{y} = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y)$

即$\hat{y} = argmax_y [logP(y) + \sum_{i=1}^{n} logP(x_i|y)]$

这里具体想就$P(x_i|y)$的计算做几点说明：

1. $\{x_i\}$包括邮件头信息和特征词
2. 对于邮件头：$P(x_i|y) = \frac{y\text{类样例中}x_i=k\text{的个数} + 1}{y\text{类样例个数} + x_i\text{的取值数}}$
3. 对于特征词：$P(x_i|y) = \frac{\text{测试样例}x_i\text{出现的次数} \times y\text{类样例中}x_i\text{出现的次数} + 1}{y\text{类样例总词数} + \text{特征词的个数}}$

具体功能的实现

- **数据预处理**

  对邮件的头信息的提取主要是由正则匹配完成的。鉴于在大数据中有可能出现数据不规范的问题，使用 try/except 来规避错误（事实证明确实有的邮件头无法匹配）。

```
hour = re.search('Date: .*([0-9]{2}):.*:.*', head).group(1)
send1 = re.search('Received: from ([0-9a-zA-Z\.]*) ', head).group(1)
send2 = re.search('From: .*@([0-9a-zA-Z\.]*)', head).group(1)
receive1 = re.search('for <([@0-9a-zA-Z\.]*)>', head).group(1)
receive2 = re.search('\nTo: ([@0-9a-zA-Z\.]*)', head).group(1)
p = re.search('X-Priority: ([0-9])', head).group(1)
```

  预处理还需要处理非中文字符。

```
# filtrate the punctuation
filterate = re.compile(u'[^\u4E00-\u9FA5]')
content = filterate.sub(r' ', content)
# simplify the whitespace
content = re.sub('\s\s+', ' ', content)
```

  预处理后的所有特征如下，将存入到 json 文件中以便后续的访问。

```
sample = dict()
sample['input'] = inputs
sample['label'] = labels
sample['priority'] = priority
sample['send_address'] = send_address
sample['receive_address'] = receive_address
sample['times'] = times
```

- **抽取特征值**

  抽取特征值的主要思想是确定一个上下界来保证特征词语的代表性。在后续的实验过程中发现还有一些停用词毫无意义，比如“啊”等。因此在网上寻找了一份停用词表来进行筛选，停用词表可见 stopwords.txt。发现筛选后程序的准确性有了比较大的提升。

```
vec = [k for k, v in features.items() if v > lowerbound and v < upperbound and
len(k) > 1]
for word in stopwords:
    if word in vec:
        vec.remove(word)
```

- **训练集统计**

  遍历训练集统计相应的频数即可。在这里有一些小 tricks，可以提升程序的效率。如果使用词袋模型，我们判断相关词语是否为特征词时可以直接调用 in dict 而不是 in

list。由于规模较大，后者会极大的减慢程序的运行效率。

- **计算似然度**

  根据算法原理中的公式计算即可。在具体实践中，在信头特征的似然度上选择乘以一个因子以获得算法的最优表现。

- **做出预测**

  本程序从五个角度评估算法的表现：

```
# SINGLE: 测试集大小
nss = 0 # spam->spam
nsh = 0 # spam->ham
nhs = 0 # ham->spam
nhh = 0 # ham->ham
recall = nss / (nss + nsh) # 召回率
precision = nss / (nss + nhs) # 精确率
accuracy = (nhh + nss) / SINGLE_SIZE # 准确率
error_rate = (nsh + nhs) / SINGLE_SIZE # 错误率
F = 2 * recall * precision / (recall + precision) # F 值
```

实验结果

```
$ python3 cross_validate_v2.py
Loading input and features
Round   0
Validating
accuracy:   0.9770194986072424
Round   1
Validating
accuracy:   0.9745434849891674
Round   2
Validating
accuracy:   0.9805013927576601
Round   3
Validating
accuracy:   0.9801918910554008
Round   4
Validating
accuracy:   0.9794181367997524
avg_recall:   0.9828316865146925
avg_precision:   0.9845260869207451
avg_accuracy:   0.9783348808418447
avg_error_rate:   0.021665119158155367
avg_F:   0.9836781570607201
```

如上述方法实现的朴素贝叶斯分类器的平均准确率为 97.8%，其他各指标如上图。可以发现朴素贝叶斯的方法在此处效果还是非常好的。

事实上，在实验前后我实现了两种方法。第一种方式的思路是找到特征词后将整个数据向量化，如果特征词出现在某封邮件中则置为 1，否则置为 0，基于此计算每个特征词在垃圾邮件和非垃圾邮件中出现的概率，进而计算似然度。这种方式的优点是将数据集向量化，在计算似然度时会比较快。而缺点是向量化后得到的稀疏矩阵会使整个数据集变得非常巨大，在读写操作时会比较慢。同时由于舍弃了词频这一特征，算法的准确率大幅下降，仅为 90%左右。第二种方式则是提交版本中基于词频的朴素贝叶斯。在这一方法中，我在多处进行了改进，在后续内容中会详细阐释。改进后准确率显著提升，达到了 97.8%。

实验优化与分析

■ ISSUE 1: THE SIZE OF TRAINING SET

由于样例数据较多，下对训练集大小对分类器性能的影响进行分析。分别使用 5%, 50%, 100%的训练集来训练模型并在测试集上进行测试，统计准确率可得下表：

	5%	50%	100%
1	95.25%	97.49%	97.70%
2	95.71%	97.33%	97.45%
3	96.15%	97.79%	98.05%
4	95.93%	97.76%	98.02%
5	96.19%	97.81%	97.94%
min	95.25%	97.33%	97.45%
max	96.19%	97.81%	98.05%
avg	95.85%	97.64%	97.83%

可以发现，随着训练集数量的增大，其在各指标上的表现均有所增长。但是就算仅使用 5%的数据，其准确率仍然高达 95%，由此可以看出朴素贝叶斯算法的优越性。由此可见，朴素贝叶斯算法在训练集不是非常大的时候也可以得到较好的效果，因此当数据集不太大的时候朴素贝叶斯算法可能是一个较好的选择。考虑到它不会导致过拟合的优点后更是如此。

值得注意的是，在训练集规模从 5%到 50%再到 100%的过程中，准确率的提升放缓，说明其收敛速度变慢。因此在使用贝叶斯算法的过程中，我们也需要考虑数据集规模变大造成的运行时间变长的问题，即运行时间和准确率的 trade-off。

■ ISSUE 2: ZERO-PROBABILITIES

在运行时，如果没有对概率进行平滑处理，会发现程序会提示在 np.log 运算中遇到 0。原因是测试集中的数据的某一特征在训练集的一部分中并没有出现。具体来说就是$\#\{y=c, x_i=k\}=0$。解决这一问题的方法是使用拉普拉斯平滑：

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

特别的，我们在这里取$\alpha=1$ $M=$ 特征词数（特征词）/ 总邮件数（信头）。

平滑保证了程序在计算时的准确性。但是在另一方面，概率本身的计算也同样重要。在这里，我想特别的解释一下我对计算特征词条件概率的一些想法。在"算法原理"一节中我给出其计算公式为$P(x_i|y) = \frac{\text{测试样例中}x_i\text{出现的次数} \times y\text{类样例中}x_i\text{出现的次数} + 1}{y\text{类样例总词数} + \text{特征词的个数}}$，比较不同的是分子中乘了一个测试样例中出现的次数（记为n），而没有选择整个概率的n次方。这是出于以下的考虑：如果特征词汇在测试样例中出现的次数越多，那么该项概率应当更大以具有更多的代表性。然而如果选择乘方的形式，概率则在变小，与实际意义不符。最后结果证明效果较好。

■ ISSUE 3: SPECIFIC FEATURES

邮件头中包含了一些有用的信息，我通过查阅相关资料[1]选取了以下的特征：

1. 发信时间 – 这是一个连续变量，我通过设定阈值的方式将其转化为离散变量（00:00:00-06:00:00, 06:00:00-24:00:00）
2. 发信人一致性 – Received 头中发信人邮箱是否与 From 头中发信人邮箱相匹配
3. 收信人一致性 – Received 头中收信人邮箱是否与 To 头中收信人邮箱相匹配
4. 邮件优先级

经过一系列实验，我发现有的特征可以对分类效果有所提升，也有的特征对分类效果没什么影响，甚至产生反效果。同时，由于每一个 specific features 在似然函数中只有一项，与数万个特征值相比还比较小，因此可以考虑施加权值。我们也可以看到权值也对准确率有显著影响。为了更直观的说明，下表展示了一部分实验结果：

special features（括号内为权值）	average accuracy
-	0.978288
Priority(10)	0.978227
Priority(5)	0.978258
Sender(10)	0.978319
Sender(10) Time(1)	0.978335

在多次调整之后，我留下了发信时间和发信人一致性两个特征以及设置了权值。最终，程序准确率由 97.82%上升到 97.83%。有趣的一点是，在 5%的测试集中，附带 special features 的程序表现更差，这说明人为的从邮件头抽取特征有一定的风险性，该方法不够 robust。以上几点也表明了这些 special features 对结果产生的影响是 trivial 的。但是，我们不应该就此否定这一方法。在别的场景中，该方法可能会有更大

[1] 袁国鑫,于洪.一种基于邮件头信息的三支决策邮件过滤方法[J].计算机科学,2017,44(09):74-77+114.

的用处。

■ ISSUE 4: LACK OF INFORMATION

在处理邮件头时遇到了一系列由于形式不一致或者信息缺失而产生的问题。形式不一致的问题形如：“From: "xin" <xin@jdl.ac.cn>”和“From: yuan@163.com”。信息缺失则是单纯的缺少 Date 等信息。

对于形式不一致的问题，最后通过调整正则表达式的形式加以解决。但是一些比较奇怪的信息或者是信息缺失，则需要另外想办法解决。对此，我的解决方法是将信息缺失视为一种新的类别。比如，对于发信时间而言，如果在头信息中没有找到发信时间这一项，则将其赋值为-1。如此计算似然度的时候就不会出现零概率的问题。事实证明，如果使用这一方法，程序的准确率达到 97.83%。

实验小结

以往我对于贝叶斯算法的了解仅仅局限在理论上的推导，而本次实验中我第一次在真实数据集上实现了一个完整的机器学习算法，并且进行了一系列的测试、改进与分析。总体来说，我认为这次实验还是非常有意义的。

在实验过程中确实发现了一些我没有预料到的插曲。在作业布置后的几天内，我其实就已经实现了一个基于特征词是否出现的算法，但是算法的准确率只有 90%左右，令我不太满意。我又引入了一些新的特征，但是效果十分有限。然而，当我考虑到词频时，我发现这一信息能极大的提升算法的效果。而有趣的一点则是，在基于词频计算似然度时，似然度的数学意义仿佛已经不是那么明显。我先后使用邮件数量、邮件长度等作为分母，发现效果越来越好。这带给我的启示就是实践才是检验真理的唯一标准。

在利用信息头方面我的一些改进并没有得到非常明显的效果，特别是考虑到其对小数据集的副作用后。程序的准确率可能随着随机种子的设置而有一定的偏差，也可能导致信息头特征作用的变化。限于时间的原因，在这方面没有进行更加深入的探索，但是我相信邮件头中可能会有更多的信息去挖掘。另外，平滑系数的选取也有更多的可能可以去尝试。

在实验的过程中我体会了许多亲手写代码才能体会到的困难，也因此有了许多收获。比如，如何提高程序的效率（实验中曾出现因选择不合适的数据结构而训练模型花了十几分钟的惨剧），如何优化存储空间等等。我也体会到了对程序的思考和评估所带来的重大作用。在很多时候，我们对程序做出了改进，此时我们应该去思考为什么这些方法会 work 或者 fail，只有这样，我们才能把短暂的灵感转化为经验，运用到下一次的实践中去。同时，我也十分感谢老师和助教的无私帮助。