
山东大学
SHANDONG UNIVERSITY

# 实验报告

| | |
|---|---|
| 课　　程: | Data Mining |
| 实验题目: | Homework 2 |
| 姓　　名: | 陈昕 |
| 学　　号: | 201814806 |
| 班　　级: | 2018 级计算机学硕班 |
| 实验时间: | 2018/11/6 – 2018/11/25 |

# 一、 实验目的

理解课堂上学习的朴素贝叶斯分类器（Naïve Bayes Classifier）的原理，包括贝叶斯公式、条件独立假设、不同模型的平滑技术及取对数等工程上的tricks。

## 任务：

1. 实现朴素贝叶斯分类器，测试其在 20Newsgroups 数据集上的效果。

# 二、 实验环境

## 硬件环境：

intel i5-7400 CPU @ 3.00 GHz
8.00 GB RAM

## 软件环境：

Windows 10 64 位 家庭版
python 3.6.6
Anaconda custom (64-bit)
Jupyter notebook 5.7.0
nltk 3.3.0
numpy 1.15.4
scikit-learn 0.20.0

# 三、 实验过程

**（我不能保证在报告描述中变量名与实际程序中完全一致。）**

## 准备工作

这一次实验中，并没有什么需要下载的东西。
我新建了 Homework2 文件夹，用于保存本实验中的所用到的程序文件等。
然后，将包含 20Newsgroups 数据集的文件夹复制到 Homework2 下。

## 任务一：实现朴素贝叶斯分类器，测试其在 20Newsgroups 数据集上的效果。

在这个任务中，我依然使用 nltk 进行文本预处理，并使用 scikit-learn 来划分训练集和测试集，并编写了一个 NaiveBayesClassifier 类来封装朴素贝叶斯分类器的实现。

**在 main.ipynb 中的主程序过程如下：**

首先，读取 20Newsgroups 数据集中所有文件的文件名 filepaths。

Scikit-learn 的 model_selection 包提供了一个 train_test_split 方法，可以在划分测试集和训练集的同时进行洗牌。我采用了该方法，把 filepaths 按 8：2 划分成了训练集和测试集，并读取了所有文件。

和 Homework1 一样，我用 nltk 进行了分词、取词干和去除停词的操作。

对于训练集，我将文件夹名作为 label，和预处理后的文档一起作为参数构建了一个 NaiveBayesClassifier，并进行训练。

对所有测试集中的数据，用分类器计算分类结果。

最后，对比分类器生成的 label 和测试集的实际 label（也就是文件夹名），计算错误率。

```
In [9]:
# create a naive Bayes classifier
labels = [os.path.dirname(fn) for fn in training_set_fns]
classifier = NaiveBayesClassifier(tokenized_docs, labels)

In [10]:
# training the classifier
classifier.train()

In [11]:
# classify the testing data for different k
results = []
for doc in whole_test_tokenized_docs:
    results.append(classifier.classify(doc))

In [12]:
error_rate = 0
sum_ = 0
for i, label in enumerate(results):
    if label != os.path.dirname(test_set_fns[i]):
        sum_ += 1

error_rate= sum_/N
print('The accuracy is: %f' % (1-error_rate))

The accuracy is: 0.831121
```

结果显示，我实现的朴素贝叶斯分类器对于所有数据集的正确率大约是 83%。这个结果实际上略低于我在 KNN 分类器中得到的最高正确率（84%），但是 KNN 分类器大约需要一个小时来完成整个分类过程，但朴素贝叶斯分类器的分类过程要快得多，只需要大约一分钟。

**NBC.py 文件中：**

该文件只包括 NaiveBayesClassifier 一个类。

类 NaiveBayesClassifier 中，没有类属性。

对象属性包括：

- training_data（训练数据集列表）
- labels（每个数据对应的标签）
- label_missing_word_prob（每个类对应的不属于该类的词的概率，用于平滑技术）。
- label_word_probability（每个类对应的属于该类的词的概率）。

- label_class_prob（训练集中，各类文档数占总数的比例）。

对象方法包括：

- train（计算对象属性里提到的各种概率，并对它们取对数）
- classify（统计文档中词的次数，求文档属于各类的概率，然后选出使得概率最大的那个类）

可以看出，我在这个朴素贝叶斯分类器的实现中采用了多项式模型。

**在 train 阶段**，我使用 python 内置的 Counter 进行统计词语在每一类中出现的次数总和，并统计词表长度。然后，用平滑技术计算词语概率，并用 math.log 对所有概率取对数。

**在 classify 阶段**，我首先把分词后的文档转化为了一个 Counter，并遍历所有 label，用相应的取对数后的类概率加上所有集合中的词的出现概率乘以出现次数，求出概率最大的类标签作为结果。

# 四、 实验结论

在本次实验中，我动手实现了朴素贝叶斯分类器，其中用到了课堂上学到的贝叶斯公式、条件独立假设、不同模型的平滑技术及取对数等工程上的 tricks，感觉对这些知识点的理解更深入了。

我对上述知识点的理解是，朴素贝叶斯分类器基于贝叶斯公式，将先验知识（训练集中的文档在类中的概率和类在总体中的概率）转换为后验知识（新文档属于哪个类）。它还引入了条件独立假设，也就是假设所有词的出现概率是相互独立的，所以可以把文档在类中的概率转换成词语在类中的概率中的累乘。平滑技术用于处理词不在类中的情况，包括伯努利模型、多项式模型和混合模型；而取对数技术把累乘转化成了累加，解决了浮点数下溢问题并提高了计算速度。

在实验过程中，我还犯过好几个错误，也学到了不少东西。

最初，我是使用迭代器作为 training_data 的来源，这样重复调用 train 方法就会导致统计结果为空，结果正确率只有 20%多。在我用 list 展开迭代器后，重复运行就正常了。

其次，我还忘记了计算类概率时也要取对数，结果正确率不到 60%。

我是通过 debug 工具定位这些问题的。使用 import pdb; pdb.set_trace() 可以设置断点，直接在 jupyter notebook 中检查中间变量的结果。

这次我还采用了 scikit-learn 中的 train_test_split 方法来随机划分训练集和测试集，能够增强结果的健壮性和可信度。

**最终结果显示朴素贝叶斯分类的正确率大概 83%，稍低于 KNN 分类器的最好情况（k=9，正确率 84%）；但是运行速度比 KNN 分类器快得多，训练和分类加起来只要大概一分钟，而 KNN 分类器需要 1 个多小时。**