信息基础 II **2023 秋**

Homework 1

学生: 刘浩 (202120120312) *时间: 9 月 7 日*

1.1 实验要求

联邦党人文集作者公案分类，找到 85 篇，按照作者两个类别分类，在这 85 篇文章当中有 11 篇作者是“HAMILTON AND MADISON”, 我们要做的就是预测这 11 篇文章的作者是谁，不能直接用 ML 库或开源代码。

1.2 实验原理

基于朴素贝叶斯进行分类，在课上我们学习了朴素贝叶斯的基本原理，在本次作业中即是对这个原理很好的应用，本实验中记 θ_1 事件为作者是 MADISON，θ_2 事件为作者是 HAMILTON，数据集 D 是关键词出现与否，在此我们假设它们是相互独立的, 因此有

$$P(\theta \mid D) = \frac{P(D \mid \theta)P(\theta)}{P(D)} = \frac{P(\theta)}{P(D)} \prod_{i=1}^{d} P\left(D_i \mid \theta\right)$$

那么利用 MAP 准则得到以下表达式：

$$\begin{aligned}
\hat{\theta}_{\mathrm{MAP}} &= \operatorname{argmax} P(\theta \mid D) \\
&= \operatorname{argmax} \log P(\theta \mid D) \\
&= \operatorname{argmax} \left[\log \frac{P(D \mid \theta)P(\theta)}{P(D)}\right] \\
&= \operatorname{argmax} \log P(D \mid \theta) \cdot P(\theta)
\end{aligned}$$

在本次实验中我们只需要计算 $P(\theta_1 \mid D)$ 与 $P(\theta_2 \mid D)$ 再比较二者大小即可

1.3 实验步骤

首先在老师提供的网站:https://avalon.law.yale.edu/subject_menus/fed.asp 找到了这八十五篇文章, 然后我将这八十五篇文章按照顺序整理在 origin_article.txt 中，以此作为基础开始用代码处理这些文章

我定义了数据预处理函数，去除文章中的特殊符号和标点符号，再将所有字母改成小写，然后就可以使用 split 函数进行分词, 这样实现输入一篇文章，输出文章的词汇

```
# 数据预处理
def preprocess_text(text):
    # 去除标点符号和特殊字符
    text = re.sub(r'[^\w\s]', '', text)
    # 转换为小写
    text = text.lower()
    # 分词
    words = text.split()
    return words
```

然后我定义了寻找高频词的函数，输入一篇文章和想要提取前多少位高频词，输出即是寻找到的前 n 位高频词

```
# 计算高频词
def calculate(words, n):
    freq_dict = {}
    for word in words:
        if word in freq_dict:
            freq_dict[word] += 1
        else:
            freq_dict[word] = 1
    key = sorted(freq_dict.items(), key=lambda x: x[1], reverse=True)[:n]
    return key
```

然后是对文章的处理，读取我整理好的 origin_article.txt，由于每篇文章是有序号的，且格式统一为“FEDERALIST No. ”因此我按照这个进行切分即可按顺序得到每一篇文章

```
# 读取文件
with open('origin_article.txt', 'r') as file:
    origin_article = file.read()

# 下面将文章拆分为列表，索引X即代表第X篇文章
split_paper = origin_article.split('FEDERALIST No. ')
```

下面按照作者进行分类，经过研究这 85 篇文章可知，作者一共有以下五种情况

(1)HAMILTON

(2)MADISON

(3)HAMILTON AND MADISON

(4)HAMILTON OR MADISON

(5)JAY

为此我按照采用搜索姓名对文章再次分类：

```
author_M = []
keywords_M = []
author_H = []
keywords_H = []
author_Guess = []
author_all = []
author_index = []

for i in range(1, 86):
    if "HAMILTON AND MADISON" in split_paper[i]:
        pass
    elif "HAMILTON OR MADISON" in split_paper[i]:
        author_Guess.append(split_paper[i])
        author_index.append(i)
    elif "HAMILTON" in split_paper[i]:
        author_H.append(split_paper[i])
    elif "MADISON" in split_paper[i]:
        author_M.append(split_paper[i])
author_all = author_H + author_M + author_Guess
```

下面是关键词的构造，我采用的是提取 HAMILTON 文章的前 250 个高频词，还有 MADISON 前 250 个高频词，再让两个高频词集合取并集得到最终的关键词

```
# 提取H的高频词
words_H = preprocess_text(' '.join(author_H))
features_H = calculate(words_H, 250)
for word, freq in features_H:
    keywords_H.append(word)

# 提取M高频词
words_M = preprocess_text(' '.join(author_M))
features_M = calculate(words_M, 250)
for word, freq in features_M:
    keywords_M.append(word)

# 合并两个作者的高频词汇作为关键词
set_H = set(keywords_H)
set_M = set(keywords_M)
keywords = set_H.union(set_M)
```

根据朴素贝叶斯原理，下面进行先验概率的计算：

```
# 先验概率计算
Pr_H = len(author_H) / len(author_H + author_M)
Pr_M = 1 - Pr_H
```

再根据我选择的关键词结合 numpy 库构造特征向量，分别计算 MADISON 和 HAMILTON 的特征向量，以此计算条件概率：

```
# 构建特征向量
keywords = list(set(keywords))

vector_guess = np.zeros(len(keywords))
vector_H = np.zeros(len(keywords))
vector_M = np.zeros(len(keywords))
vector_list = []

# 计算M的条件概率
for M_word in words_M:
    for i, word in enumerate(keywords):
        if M_word == word:
            vector_M[i] += 1
condition_M = vector_M / np.sum(vector_M)

# 计算H的条件概率
for H_word in words_H:
    for i, word in enumerate(keywords):
        if H_word == word:
            vector_H[i] += 1
condition_H = vector_H / np.sum(vector_H)
```

最后计算未知作者的特征向量，统一添加到 vector_list 中，再利用每篇未知作者文章的特征向量结合朴素贝叶斯原理计算作者为 MADISON 的概率和 HAMILTON 的概率进行大小的比较：

```
# 进行预测
for i in range(11):
    papers = preprocess_text(author_Guess[i])
    vector_guess = np.zeros(len(keywords))  # 创建一个与关键词数量相同的零向量
    for j in papers:
        for k, word in enumerate(keywords):
            if j == word:
                vector_guess[k] = 1  # 对每篇文章单独构建特征向量
    vector_list.append(vector_guess)  # 将特征向量添加到列表中
    # 下面开始计算概率
    guess_M = vector_list[i] * condition_M
    guess_H = vector_list[i] * condition_H
    log_prob_M = 0
    log_prob_H = 0
    for pm in guess_M:
        if pm != 0:
            log_prob_M += np.log(pm)
        else:
            pass
    for ph in guess_H:
        if ph != 0:
            log_prob_H += np.log(ph)
        else:
            pass
    prob_M = log_prob_M + np.log(Pr_M)
    prob_H = log_prob_H + np.log(Pr_H)
```

```
    if prob_M > prob_H:
        print("第", author_index[i], "篇的作者为Madision")
    else:
        print("第", author_index[i], "篇的作者为Hamilton")
    print('----------------------------------------------')
```

1.4 实验结果

程序运行结果如下：

```
第 49 篇的作者为Madision
----------------------------------------------
第 50 篇的作者为Madision
----------------------------------------------
第 51 篇的作者为Madision
----------------------------------------------
第 52 篇的作者为Madision
----------------------------------------------
第 53 篇的作者为Madision
----------------------------------------------
第 54 篇的作者为Madision
----------------------------------------------
第 55 篇的作者为Madision
----------------------------------------------
第 56 篇的作者为Madision
----------------------------------------------
第 57 篇的作者为Madision
----------------------------------------------
第 62 篇的作者为Madision
----------------------------------------------
第 63 篇的作者为Madision
----------------------------------------------
```

经过查阅资料 [4] 可知，未知的这十一篇文章的作者均为 Madision，预测结果相当正确

1.5 实验感想

在本次实验中，由于不能使用 ML 库，让我更好的理解了朴素贝叶斯进行预测的基本原理，课上看很多公式还是迷迷糊糊，但是经过自己通过代码实现一遍这个过程之后清晰了不少，更好的理解了朴素贝叶斯原理，更加体会到了其在数据较少的情况下仍然有效、可以处理多类别问题的优点，收获颇丰！

1.6 参考资料

[1] http://t.csdn.cn/lweNc

[2] http://t.csdn.cn/4ixyB

[3] https://zhuanlan.zhihu.com/p/452205372

[4] https://zhuanlan.zhihu.com/p/75827804