

K-NN 文档分类实验报告

姓 名：余 平 学 号：201814848

一、重点知识：

1、**向量空间模型 VSM** (Vector Space Model, 简称 VSM): 表示通过向量的方式来表征文本。将一个文档抽象为一系列关键词的向量, 该向量由 n 个关键词组成, 每个词都有一个权重 (Term Weight), 不同的词根据自己在文档中的权重来影响文档相关性的重要程度。

2、**TF-IDF**: 表示 TF (词频) 和 IDF (逆文档频率) 的乘积。特征抽取完后, 因为每个词语对实体的贡献度不同, 所以需要对这些词语赋予不同的权重。一个文档的 TF-IDF 与一个词在该文档中的出现次数成正比, 与该词在整个语言环境 (训练集) 中的出现次数成反比。TF-IDF 计算权重越大表示该词条对这个文本的重要性越大。

3、**余弦相似度**: 两篇文章间的相似度通过两个向量的余弦夹角 \cos 来描述。Cosine 值越大, 文章相似性越高。

二、实验步骤：

1、读出数据集文档, 按照训练集与测试集 4: 1 的比例进行分类, 并为每篇文档打上分类标签。

2、对数据集进行预处理: 去特殊符号、分词、转换成小写字母、去掉非英语单词、词形还原、去停用词。

分词使用 `TextBlob()` 分词工具;

词形还原只是将名词还原成单数形式, 将动词转换成原型, 其余词性的词未作变换。

3、计算的词频: 使用 `collections.Counter()` 类对每篇文档进行词频统计。

4、使用训练集构建字典: 根据统计词频的情况去除一些低频词和高

频词，以减小文档向量空间的维度。

5、用训练集的词频统计计算 IDF：

$$IDF(t) = \log\left(\frac{N}{df(t)}\right)$$

为避免 $df(t) = 0$ ，在公式分母加 1，即：

$$IDF(t) = \log\left(\frac{N}{df(t) + 1}\right)$$

6、分别计算训练集和测试集文档的 TF：

$$tf(t, d) = \begin{cases} 1 + \log c(t, d), & \text{if } c(t, d) > 0 \\ 0, & \text{otherwise} \end{cases}$$

7、计算训练集和测试集文档的 TF*IDF 值：

$$w(t, d) = TF(t, d) \times IDF(t)$$

8、计算测试集文档与训练集文档向量的 cosine 值，用两个文档向量的 cosine 值表示其 similarity。

$$\cosine(d_i, d_j) = \frac{V_{d_i}^T V_{d_j}}{|V_{d_i}|_2 \times |V_{d_j}|_2}$$

9、设置 K 值，找到与测试文档相似文档最多的类，判断分类是否正确。

三、实验结果：

实验中所有的数学运算使用矩阵进行计算。

将数据集的 80 作为训练集并构建词典，20%作为测试集测试模型，在该数据集上，K 取值在区间[5,41]上滑动时，对结果的影响不大，如下图(横轴为 K 值)：



当 $K=25$ ，字典词频为 2-5000 时，结果如下：

```
the type of train_VSM_array is (15069, 25010)
the type of test_VSM_array is (3759, 25010)
N*M矩阵的维数: (3759, 15069)
the correct rate of classifier is 78.56%
```

图中第一行为训练集生成的向量空间的维数；

第二行为测试集生成的向量空间的维数；

第三行为测试集与训练集的 cosine 值的维数；

第四行最终分类的正确率。

四、实验总结

分类结果很大程度上取决于数据集的预处理和词典的构建， K 值在一定范围内对结果的影响相对较小。在预处理阶段，判断一个字符串是否为英语单词、去停用词的效果取决于字典的大小，在实验中也发现了词典不够用的情况；词形还原时效果不是很好。预处理函数的选取很大程度上影响了实验结果，完善数据集的预处理能够进一步提高正确率。

使用 VSM 进行文本分类，运算量很大，在本次实验中一个向量的维度达到 25010。

NBC experimental report

name: ping yu ID: 201814848

1. Key point:

1.1 Naïve Bayes

Assume a target function with $f: X \rightarrow V$, where each instance x is described by $\langle x_1, x_2, \dots, x_n \rangle$. Most probable value of $f(x)$ is:

$$\begin{aligned} v_{MAP} &= \arg \max_{v_j \in V} P(v_j | x_1, x_2, \dots, x_n) \\ &= \arg \max_{v_j \in V} \frac{P(x_1, x_2, \dots, x_n | v_j) P(v_j)}{P(x_1, x_2, \dots, x_n)} \\ &= \arg \max_{v_j \in V} P(x_1, x_2, \dots, x_n | v_j) P(v_j) \end{aligned}$$

Using the Naïve Bayes assumption:

$$v_{NB} = \arg \max_{v_j \in V} P(v_j) \prod_i P(x_i | v_j)$$

1.2 Smoothing

To overcome the following issue:

$$P(x_i | v_j) = 0 \text{ or } P(v_j) \prod_i P(x_i | v_j) = 0$$

Smoothing: $P(x_i | v_j) \leftarrow \frac{n_c + mp}{n + m}$

1.3 Simplified Calculation

$$v_{NB} = \arg \max_{v_j \in V} P(v_j) \prod_i P(x_i | v_j)$$

↓

$$\log(v_{NB}) = \arg \max_{v_j \in V} (\log(P(v_j)) + \sum_i \log(P(x_i | v_j)))$$

2. Experiment steps

Step 1: Data preprocessing: Tokenization, Normalization, Stemming,

Remove stopwords.

Step 2: Divide the data into training data and testing data. Testing data account for 20% of the data.

Step 3: On the base of the data set structure, store the training set in the following data structure:

List 1 = [{a:1,..., b:2},..., {a:2, ...,b:3, ...,c=2}]

The list contains 20 objects and each object is a class of training set. Each dictionary in the list shows the number of every word in this class.

Store the testing set in the following data structure:

List 2 = [[{}],..., {}]...[{a:1,..., b:2},..., {a:2, ...,b:3, ...,c=2}]

The outer list contains 20 objects and each object is a class of testing set. The length of each inner list shows the number of the document in this class. Each dictionary demonstrates the weights of all words in a document.

Step 4:

For each document in testing set, compute the weight the document belongs to each class.

The weight(W) a document belongs to a class:

$$W = \frac{P("a class"|"every words in the document")P("a class")}{P("other classes"|"every words in the document")}$$

Classify the document to the class corresponding to the min(W).

Related code:

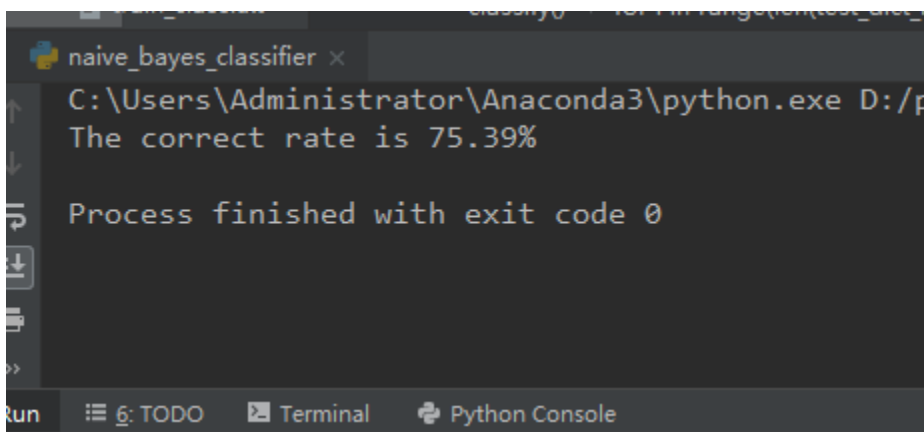
```

correct_num = 0
test_num = 0
for i in range(len(test_dict_list)):
    for j in range(len(test_dict_list[i])):
        test_num += 1
        predict = []
        for k in range(len(train_dict_list)):
            class_p = 0
            class_p1 = 0
            for key in test_dict_list[i][j].keys():
                #计算有一个词在一个类中的概率log值
                if key not in train_dict_list[k].keys():
                    continue
                else:
                    value = train_dict_list[k][key]*test_dict_list[i][j][key]
                    temp_p = math.log(value / class_word_num[k])
                    class_p += temp_p
                #计算一个词在另外19个类中的概率log值
                value1 = 0
                for m in range(len(train_dict_list)):
                    if m == k:
                        continue
                    else:
                        if key not in train_dict_list[m].keys():
                            continue
                        else:
                            value1 += train_dict_list[m][key]*test_dict_list[i][j][key]
                if value1 == 0:
                    temp_p1 = 0
                else:
                    temp_p1 = math.log(value1 / (all_words_num - class_word_num[k]))
            class_p1 += temp_p1
            class_p += math.log(class_document_num[k] / all_document_num)
            class_p1 += math.log((all_document_num - class_document_num[k]) / all_document_num)
            predict.append(class_p / class_p1)
        if predict.index(min(predict)) == i:
            correct_num += 1
predict_rate = correct_num / test_num

```

3. Experiment result:

The final correct rate is 75.39% when I use the function above.



```

naive_bayes_classifier x
C:\Users\Administrator\Anaconda3\python.exe D:/p
The correct rate is 75.39%

Process finished with exit code 0

```

4. Conclusion

Based on the results of this experiment alone, the accuracy of Bayes Classifier is lower than K-NN under the same preprocessing of the same specific data set.

Clustering with Sklearn

姓 名：余 平 学 号：201814848

(一) 实验任务

- (1) 检验 sklearn 中聚类算法在 tweets 数据集上的聚类效果。
- (2) 使用 NMI(Normalized Mutual Information)作为评价指标。

(二) 实验数据集

数据集一共包含 2472 行，代表 2472 个测试样本，cluster 的个数是 89 。

(三) 实验过程

(1) 根据 tweet 构建字典，建立向量空间模型，每一行用向量表示。词典的大小是 5097，将每一个样本表示成一个 5097 维的向量。

(2) 评价标准 (NMI) 在 Sklearn 包中有已经定义好的函数，调用这个函数来评估自己的聚类效果。

(三) 实验结果

(1) K-Means

```
K-means accuracy: 0.7881140298093793
```

(2) Affinity propagation

```
AffinityPropagation accuracy: 0.7834777200368181
```

(3) Mean-shift

```
MeanShift accuracy: 0.7468492000608157
```

(4) Spectral clustering

```
SpectralClustering accuracy: 0.6782978969064626
```

(5) DBSCAN

```
DBSCAN accuracy: 0.7009526046894612
```

(6) Agglomerative clustering

```
AgglomerativeClustering accuracy: 0.7800394104591923
```

(四) 实验总结 通过本次实验，熟悉了 Sklearn clustering 中的各种聚类方法和聚类的过程，不同的聚类方法得到不同的聚类效果。