计算机科学与技术学院　　　　　201844911 杨子玉

实验报告 1

1.实验数据来源：20news-18828.tar.gz - 20 Newsgroups

下载:http://qwone.com/~jason/20Newsgroups/

2.相关方法：

1)TF-IDF 是一种用于信息检索与数据挖掘的常用加权技术。TF 意思是词频(Term Frequency)，IDF 意思是逆文本频率指数(Inverse Document Frequency)。

2)VSM：把对文本内容的处理简化为向量空间中的向量运算

3)KNN: 邻近算法， K 最近邻，就是 k 个最近的邻居的意思，说的是每个样本都可以用它最接近的 k 个邻居来代表.

3.预处理文本数据集:

1)将实验数据分成两部分：80%的 data_train 和 20%的 data_test

2)对文本进行分词、大小写进行统一以及词干提取分析，去除停用词等处理

3)对词频大于 9 小于 10000 创建字典 dictionary.csv

4.得到每个文本的 VSM 表示：

遍历文本数据，计算 TF-IDF 值，得到每个文本(包括训练数据和测试数据)的 VSM 向量表示

5. 实现 KNN 分类器，测试其在 20 测试数据上的准确率

对训练数据形成 KNN 分类器，选出其中距离最近的 k=40 个样本，返回类别标签，其中出现次数最多的标签为预测结果。根据预测结果与其本身的类别进行比较，得到准确率。

6.实验结果如下图所示

形成的准确率大都在 0.75 以上

```
In [1]: runfile('C:/Users/Administrator/Desktop/xu/vsm+knn.py',
wdir='C:/Users/Administrator/Desktop/xu')
Divided into two parts
train_set_end
test_set_end

1 Accuracy:     0.7201383346634743
2 Accuracy:     0.7201383346634743
3 Accuracy:     0.7334397446129289
4 Accuracy:     0.7499334929502527
5 Accuracy:     0.7517956903431764
6 Accuracy:     0.759244479914871
7 Accuracy:     0.7613727055067837
8 Accuracy:     0.7648310720936419
9 Accuracy:     0.7749401436552275
10 Accuracy:    0.7770683692471402
11 Accuracy:    0.7741420590582602
12 Accuracy:    0.7773343974461293
13 Accuracy:    0.7781324820430966
14 Accuracy:    0.7799946794360202
15 Accuracy:    0.7826549614259112
16 Accuracy:    0.7834530460228785
17 Accuracy:    0.7805267358339985
18 Accuracy:    0.7826549614259112
19 Accuracy:    0.782388933226922
20 Accuracy:    0.782388933226922
21 Accuracy:    0.7829209896249002
22 Accuracy:    0.782388933226922
23 Accuracy:    0.7813248204309656
24 Accuracy:    0.7834530460228785
25 Accuracy:    0.7829209896249002

25 Accuracy:    0.7829209896249002
26 Accuracy:    0.7815908486299548
27 Accuracy:    0.7831870178238893
28 Accuracy:    0.7821229050279329
29 Accuracy:    0.7839851024208566
30 Accuracy:    0.7866453844107475
31 Accuracy:    0.7855812716147912
32 Accuracy:    0.7842511306198457
33 Accuracy:    0.7818568768289439
34 Accuracy:    0.7826549614259112
35 Accuracy:    0.7847831870178239
36 Accuracy:    0.7842511306198457
37 Accuracy:    0.7858472998137802
38 Accuracy:    0.7855812716147912
39 Accuracy:    0.7845171588188348
40 Accuracy:    0.785049215216813
41 Accuracy:    0.782388933226922
42 Accuracy:    0.7810587922319766
43 Accuracy:    0.7831870178238893
44 Accuracy:    0.7842511306198457
45 Accuracy:    0.7845171588188348
46 Accuracy:    0.7845171588188348
47 Accuracy:    0.7837190742218675
48 Accuracy:    0.7842511306198457
49 Accuracy:    0.7837190742218675
```

实验报告 2

1.实验数据来源：20news-18828.tar.gz - 20 Newsgroups

下载:http://qwone.com/~jason/20Newsgroups/

2.相关方法：

1) **朴素贝叶斯分类器**基于一个简单的假定：给定目标值时属性之间相互条件独立。换言之。该假定说明给定实例的目标值情况下。观察到联合的 $a_1,a_2...a_n$ 的概率正好是对每个单独属性的概率乘积： P(a1,a2...an | Vj) =∏i P(ai| Vj)2)VSM：把对文本内容的处理简化为向量空间中的向量运算。通过以上定理和"朴素"的假定，可以知道：

P(Category | Document) = P (Document | Category) * P(Category) / P(Document)

2)**拉普拉斯平滑处理**：零概率问题，就是在计算实例的概率时，如果某个量 x，在观察样本库（训练集）中没有出现过，会导致整个实例的概率结果是 0。在文本分类的问题中，当一个词语没有在训练样本中出现，该词语调概率为 0，使用连乘计算文本出现概率时也为 0。这是不合理的，所以使用加 1 的方法。

3.处理文本数据集:

1)将实验数据分成两部分：80%的 data_train 和 20%的 data_test

2)对训练集和测试集创建向量[类名，所有单词的长度，出现的概率，字典]

4.进行分类：

对每个待分类的文档，利用公式计算，并统计成功的文件数和失败的文件数，得到准确率

5.实验结果如下图所示

在 NB1 中采取 Homework1 中已经分好的训练集和测试集，计算步骤可能出现问题，在 NB2 中采用 Pythonsklearn 自带的贝叶斯分类器完成文本分类，使用

MultinomialNB，假设特征的先验概率为多项式分布，添加新闻标签 10 个进行分类，可以看见越多的训练类别得到的准确度越高 ，但没有写一个添加标签的函数，直接进行导入的。

NB1

```
In [88]: runfile('C:/Users/Administrator/Documents/Tencent Files/
917956361/FileRecv/NBC.py', wdir='C:/Users/Administrator/Documents/
Tencent Files/917956361/FileRecv')
strat get vector:)
finish
测试集文档总数: 3759
Accuracy: 0.595903165735568
```

NB2

```
In [82]: runfile('C:/Users/Administrator/Desktop/Homework/Homework2/
untitled12.py', wdir='C:/Users/Administrator/Desktop/Homework/
Homework2')
训练集数量:  6113
测试集数量:  1529
Accuracy
0.8639633747547416
```

```
In [83]: runfile('C:/Users/Administrator/Desktop/Homework/Homework2/
NB2.py', wdir='C:/Users/Administrator/Desktop/Homework/Homework2')
训练集数量:  7704
测试集数量:  1926
Accuracy
0.8997923156801662
```

实验报告 3

1.相关资料：https://scikit-learn.org/stable/modules/clustering.html#

实验任务：测试 sklearn 中以下聚类算法在 tweets 数据集上的聚类效果。

使用 NMI(Normalized Mutual Information)作为评价指标。

2.相关方法：

scikit-learn 简称 sklearn，支持包括分类、回归、降维和聚类四大机器学习算法。还包含了特征提取、数据处理和模型评估三大模块。

此次作业主要使用以下几种聚类方法：

Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large `n_samples`, medium `n_clusters` with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters	Distances between points
Affinity propagation	damping, sample preference	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with `n_samples`	Many clusters, uneven cluster size, non-flat geometry	Distances between points
Spectral clustering	number of clusters	Medium `n_samples`, small `n_clusters`	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints	Distances between points
Agglomerative clustering	number of clusters, linkage type, distance	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints, non Euclidean distances	Any pairwise distance
DBSCAN	neighborhood size	Very large `n_samples`, medium `n_clusters`	Non-flat geometry, uneven cluster sizes	Distances between nearest points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation	Mahalanobis distances to centers

3.处理文本数据集:

1)将实验数据的文本和应属于的类别放入两个向量中

2)调用库函数计算每个文本的 tf-idf 值

4.进行聚类：

调用函数聚类，同时采用 NMI(Normalized Mutual Information) 标准化互信息 评价效果

5.实验结果如下图所示

可以看到大多集中在 0.7 左右范围，AffinityPropagation 的效果最好。

```
In [48]: runfile('F:/anacodaa/123/Lib/site-packages/sklearn/
feature_extraction/untitled11.py', wdir='F:/anacodaa/123/Lib/site-
packages/sklearn/feature_extraction')
start cluster!
K-means: 0.7841980308246572
AffinityPropagation: 0.785654609647782
MeanShift: 0.7468492000608158
SpectralClustering: 0.6740829992908092
AgglomerativeClustering: 0.7843154591464184
DBSCAN: 0.7049439626810924
GaussianMixture:0.775646245521511
end
```