武汉大学国家网络安全学院

实验报告

课程名称 机器学习

专业年级

姓　　名

学　　号

协 作 者 无

实验学期 2022-2023 学年 2 学期

课堂时数 课外时数

填写时间 2023 年 5 月 22 日

实验概述

【实验项目名称】： 文本分类

【实验目的】：

通过编程实现基于感知机、k 近邻朴素贝叶斯和支持向量机的文本分类模型

【实验环境】（使用的软件）：

(1) 硬件环境：AMD R9 5900HX

(2) 操作系统环境：Windows 11

(3) 测试脚本编程语言：python 3.10.5

(4) 被测系统编程语言：python 3.10.5

(5) 网络环境：武大校园网

(6) 其他环境：sklearn 0.0.post5

【参考资料】：《统计学习方法》（第二版）李航

https://blog.csdn.net/qq_41856733/article/details/106415101

实验内容

【实验方案设计】：

一、文本预处理

20newsgroups 数据集是用于文本分类、文本挖据和信息检索研究的国际标准数据集之一。数据集收集了大约 20,000 左右的新闻组文档，均匀分为 20 个不同主题的新闻组集合。一些新闻组的主题特别相似，还有一些却完全不相关。此数据集可以通过 sklearn 直接装载。

```
train=fetch_20newsgroups(data_home=None, # 文件下载的路径
                         subset='train', # 加载那一部分数据集 train/test
                         categories=None, # 选取哪一类数据集[类别列表]，默认 20 类
                         #categories=['alt.atheism','comp.graphics','misc.forsale','rec.autos','sci.crypt','soc.religion.christian','talk.politics.guns'],
                         shuffle=True,  # 将数据集随机排序
                         random_state=42, # 随机数生成器
                         remove=(), # ('headers','footers','quotes') 去除标题、结尾或引用
                         download_if_missing=True # 如果没有下载过，重新下载
                         )
    #test=fetch_20newsgroups(categories=['alt.atheism','comp.graphics','misc.forsale',
```

```
    'rec.autos','sci.crypt','soc.religion.christian','talk.politics.guns'],subset='test')
11.     test=fetch_20newsgroups(subset='test')
```

使用 sklearn.datasets.fetch_20newsgroups()可直接下载并使用此数据集，并可以通过参数来进行训练集和测试集的选择。

二、提取向量

TF-IDF(term frequency - inverse document frequency)是一种用于信息检索与数据挖掘的常用加权技术，常用于挖掘文章中的关键词，而且算法简单高效，常被工业用于最开始的文本数据清洗。TF-IDF 有两层意思，一层是"词频"（Term Frequency，缩写为 TF），另一层是"逆文档频率"（Inverse Document Frequency，缩写为 IDF）。

计算 TF-IDF 时，第一步计算词频，词频即某个词在文章中出现的次数，但文章的长度有长有短，为了统一不同文章之间的比较，计算公式如下：

$$\text{词频(TF)} = \frac{\text{某个词在文章中的出现次数}}{\text{文章的总词数}}$$

第二步，计算逆文档频率。这时，需要一个语料库，用来模拟语言的使用环境。如果一个词越常见，那么分母就越大，逆文档频率就越小越接近 0。分母之所以要加 1，是为了避免分母为 0（即所有文档都不包含该词）。

$$\text{逆文档频率(IDF)} = \log\left(\frac{\text{语料库的文档总数}}{\text{包含该词的文档数} + 1}\right)$$

第三步，计算 TF-IDF

$$TF-IDF = 词频(TF) \times 逆文档频率（IDF）$$

此值与一个词在文档中出现的次数成正比，与在语言环境中出现的次数成反比，即一个词平时用的越少而在某篇文章中出现的越多，越说明这个词能代表这篇文章，用于文本分类的价值越高。

sklearn 中，可以使用 CountVectorizer 和 TfidfTransformer 来计算 TF-IDF，其中前者用来计算计数矩阵，后者则使用前者计算出的计数矩阵来计算 TF-IDF 值。
不过我选择使用 TfidfVectorizer 来一步到位地计算出 TF-IDF 来。
参数中的 stop_words=’english’意为使用自带的英语停用词汇表。

```
def TDIDF(train, test):
    #计算TF-IDF
    vectorizer = TfidfVectorizer(stop_words='english')
    train_v=vectorizer.fit_transform(train.data)
    test_v=vectorizer.transform(test.data)
    return train_v,test_v
```

三、模型实现

1. 朴素贝叶斯

```
def beiyesi(train,train_v,test_v):
    clf=[]
    pred=[]
    alpha=1.0
    for i in tqdm(range(5),colour='#00B2FF'):
        clf.append(MultinomialNB(alpha=alpha))
        clf[i].fit(train_v,train.target)
        pred.append(clf[i].predict(test_v))
        print(' alpha=',alpha,'f1_score:',f1_score(test.target,pred[i],average='macro'),'accuracy_score:',accuracy_score(test.target,pred[i]))
        alpha=alpha/10
    #print(classification_report(test.target, pred[2]))
    #print(classification_report(test.target, pred[3]))
```

在 sklearn 中有朴素贝叶斯的方法，直接调用即可，设置不同的参数来选择出准确率最高的哪一个。这里我设置 alpha 为 1, 0.1 ,0.01, 0.001, 0.0001。

2. 感知机

```
def ganzhiji(train,train_v,test_v):
```

```
    clf=[]
    pred=[]
    alpha=['l2','l1','elasticnet','None']
    for i in tqdm(range(4),colour='#00B2FF'):
        if i < 3:
            clf.append(Perceptron(penalty=alpha[i]))
        else:
            clf.append(Perceptron())
        clf[i].fit(train_v,train.target)
        pred.append(clf[i].predict(test_v))
        print(' penalty=',alpha[i],'f1_score:',f1_score(test.target,pred[i],average='macro'),'accuracy_score:',accuracy_score(test.target,pred[i]))
```

直接调用感知机的方法进行训练，这里参数设置为惩罚项，分别为 L2 正则，L1 正则，混合正则，和无罚项。

3. k 近邻

```
def k(train,train_v,test_v):
    clf=[]
    pred=[]
    alpha=1
    for i in tqdm(range(5),colour='#00B2FF'):
        clf.append(KNeighborsClassifier(n_neighbors=alpha))
        clf[i].fit(train_v,train.target)
        pred.append(clf[i].predict(test_v))
        print(' n_neighbors=',alpha,'f1_score:',f1_score(test.target,pred[i],average='macro'),'accuracy_score:',accuracy_score(test.target,pred[i]))
        alpha=alpha+1
```

方法同上，参数为邻居数，设置为 1 到 5。

4. 支持向量机

```
def zhichi(train,train_v,test_v):
    clf=[]
    pred=[]
    alpha=0.0
    for i in tqdm(range(6),colour='#00B2FF'):
        if alpha==0.0:
            clf.append(SVC(C=0.1))
        else:
            clf.append(SVC(C=alpha))
        clf[i].fit(train_v,train.target)
        pred.append(clf[i].predict(test_v))
```

```
        if alpha==0.0:
            print(' C=0.1','f1_score:',f1_score(test.target,pred[i],average='macro'),'accuracy_score:',accuracy_score(test.target,pred[i]))
        else:
            print(' C=%.1f'%alpha,'f1_score:',f1_score(test.target,pred[i],average='macro'),'accuracy_score:',accuracy_score(test.target,pred[i]))
        alpha=alpha+0.2
```

参数 C 根据官方文档，这是一个软间隔分类器，对于在边界内的点有惩罚系数 C，C 的取值在 0-1。0 之间，默认值为 1.0，这里设置为 0.1，0.2，0.4，0.6，0.8，1。

5. 多层感知机

```
def duocengganzhi(train,train_v,test_v):
    clf=[]
    pred=[]
    alpha=[[50,50],[50,100][100,100]]
    for i in tqdm(range(6),colour='#00B2FF'):
        clf.append(MLPClassifier(hidden_layer_sizes=alpha[i]))
        clf[i].fit(train_v,train.target)
        pred.append(clf[i].predict(test_v))
        print(' hidden_layer_sizes=',alpha[i],'f1_score:',f1_score(test.target,pred[i],average='macro'),'accuracy_score:',accuracy_score(test.target,pred[i]))
        #print(classification_report(test.target, pred[i]))
```

参数为隐藏神经元数，参数设置如图。

四、结果分析

结果分析中取 F1 分数和精确率作为指标。F1-score 是分类问题的一个衡量指标。一些多分类问题的机器学习竞赛，常常将 F1-score 作为最终测评的方法。它是精确率和召回率的调和平均数，最大为 1，最小为 0。

1. 朴素贝叶斯

```
请选择模型：1：贝叶斯，2：感知机，3：k近邻，4：支持向量机，5，多层感知机，0：退出
1
  0%|
alpha= 1.0 f1_score: 0.7998199251752561 accuracy_score: 0.8169144981412639
 alpha= 0.1 f1_score: 0.8249084059055034 accuracy_score: 0.8360329261816251
 40%|
alpha= 0.01 f1_score: 0.8278889894475222 accuracy_score: 0.8336431226765799
 alpha= 0.001 f1_score: 0.8180553962537778 accuracy_score: 0.8230217737652682
 80%|
alpha= 0.0001 f1_score: 0.8081916612979654 accuracy_score: 0.8134625597450876
100%|
```

可以看到参数在 0.1 时准确率最高，0.01f1 分数最高

	precision	recall	f1-score	support
0	0.83	0.75	0.79	319
1	0.76	0.74	0.75	389
2	0.76	0.69	0.72	394
3	0.67	0.78	0.72	392
4	0.87	0.85	0.86	385
5	0.86	0.81	0.83	395
6	0.85	0.79	0.82	390
7	0.90	0.90	0.90	396
8	0.92	0.97	0.94	398
9	0.94	0.94	0.94	397
10	0.92	0.98	0.95	399
11	0.84	0.95	0.89	396
12	0.82	0.76	0.79	393
13	0.91	0.84	0.87	396
14	0.85	0.93	0.89	394
15	0.73	0.96	0.83	398
16	0.71	0.94	0.81	364
17	0.95	0.96	0.95	376
18	0.89	0.57	0.70	310
19	0.88	0.40	0.55	251
accuracy			0.84	7532
macro avg	0.84	0.82	0.82	7532
weighted avg	0.84	0.84	0.83	7532

	precision	recall	f1-score	support
0	0.83	0.78	0.80	319
1	0.68	0.74	0.71	389
2	0.72	0.63	0.67	394
3	0.64	0.73	0.69	392
4	0.83	0.83	0.83	385
5	0.84	0.77	0.81	395
6	0.79	0.78	0.79	390
7	0.89	0.90	0.90	396
8	0.93	0.96	0.94	398
9	0.95	0.94	0.95	397
10	0.95	0.97	0.96	399
11	0.90	0.93	0.91	396
12	0.79	0.77	0.78	393
13	0.89	0.83	0.86	396
14	0.88	0.91	0.89	394
15	0.84	0.95	0.89	398
16	0.77	0.91	0.83	364
17	0.97	0.94	0.95	376
18	0.80	0.65	0.71	310
19	0.75	0.61	0.67	251
accuracy			0.83	7532
macro avg	0.83	0.83	0.83	7532
weighted avg	0.83	0.83	0.83	7532

取这两个参数所对应的详细结果可以看到准确率大致是先升后降，说明出现了过拟合现象。

2. 感知机

```
请选择模型：1：贝叶斯，2：感知机，3：k近邻，4：支持向量机，5，多层感知机，0：退出
2
  0%|
penalty= l2 f1_score: 0.7165826171430422 accuracy_score: 0.7258364312267658
 25%|
penalty= l1 f1_score: 0.5526541128652548 accuracy_score: 0.5528412108337759
 50%|
penalty= elasticnet f1_score: 0.7059368054251067 accuracy_score: 0.7202602230483272
 75%|
penalty= None f1_score: 0.7958331922251858 accuracy_score: 0.8023101433882103
100%|
```

可以看到无罚项时准确率最高。取详细数据。

	precision	recall	f1-score	support
0	0.77	0.76	0.77	319
1	0.71	0.77	0.74	389
2	0.70	0.68	0.69	394
3	0.70	0.69	0.70	392
4	0.82	0.78	0.80	385
5	0.79	0.72	0.75	395
6	0.82	0.81	0.82	390
7	0.86	0.88	0.87	396
8	0.86	0.92	0.89	398
9	0.86	0.91	0.88	397
10	0.95	0.95	0.95	399
11	0.83	0.91	0.87	396
12	0.81	0.62	0.70	393
13	0.86	0.80	0.83	396
14	0.78	0.90	0.84	394
15	0.80	0.89	0.84	398
16	0.72	0.88	0.79	364
17	0.89	0.83	0.86	376
18	0.77	0.61	0.68	310
19	0.69	0.62	0.66	251
accuracy			0.80	7532
macro avg	0.80	0.80	0.80	7532
weighted avg	0.80	0.80	0.80	7532

也是先升后降，出现了过拟合现象。

3. k 近邻

```
请选择模型：1：贝叶斯，2：感知机，3：k近邻，4：支持向量机，5，多层感知机，0：退出
3
  0%|
n_neighbors= 1 f1_score: 0.678100167472523 accuracy_score: 0.6805629314922995
 20%|
n_neighbors= 2 f1_score: 0.6489130970392561 accuracy_score: 0.6464418481147106
 40%|
n_neighbors= 3 f1_score: 0.6696370960654086 accuracy_score: 0.6666224110462029
 60%|
n_neighbors= 4 f1_score: 0.6721799071450127 accuracy_score: 0.6716675517790759
 80%|
n_neighbors= 5 f1_score: 0.6745190942721968 accuracy_score: 0.6757833244822092
100%|
```

在邻居为 1 时准确率最高。

	precision	recall	f1-score	support
0	0.71	0.71	0.71	319
1	0.52	0.55	0.54	389
2	0.52	0.53	0.53	394
3	0.56	0.62	0.59	392
4	0.56	0.59	0.58	385
5	0.63	0.57	0.60	395
6	0.54	0.50	0.52	390
7	0.77	0.68	0.72	396
8	0.82	0.87	0.84	398
9	0.69	0.74	0.71	397
10	0.77	0.85	0.81	399
11	0.76	0.83	0.79	396
12	0.67	0.60	0.63	393
13	0.72	0.62	0.67	396
14	0.73	0.82	0.77	394
15	0.78	0.81	0.79	398
16	0.72	0.71	0.72	364
17	0.85	0.73	0.78	376
18	0.71	0.63	0.67	310
19	0.56	0.61	0.59	251
accuracy			0.68	7532
macro avg	0.68	0.68	0.68	7532
weighted avg	0.68	0.68	0.68	7532

4. 支持向量机

```
请选择模型：1：贝叶斯，2：感知机，3：k近邻，4：支持向量机，5，多层感知机，0：退出
4
  0%|
C=0.1 f1_score: 0.2811824656687181 accuracy_score: 0.2703133297928837
 17%|
C=0.2 f1_score: 0.6256600169599948 accuracy_score: 0.5831120552310144
 33%|
C=0.4 f1_score: 0.7575042716733429 accuracy_score: 0.7533191715347849
 50%|
C=0.6 f1_score: 0.7926758995758589 accuracy_score: 0.796202867764206
 67%|
C=0.8 f1_score: 0.8128236095100622 accuracy_score: 0.8173127987254382
 83%|
C=1.0 f1_score: 0.8204625116789279 accuracy_score: 0.8252788104089219
100%|
```

C=1 时准确率最高

	precision	recall	f1-score	support
0	0.84	0.70	0.76	319
1	0.64	0.82	0.72	389
2	0.80	0.74	0.77	394
3	0.71	0.78	0.74	392
4	0.83	0.83	0.83	385
5	0.83	0.74	0.78	395
6	0.75	0.90	0.82	390
7	0.89	0.88	0.88	396
8	0.98	0.93	0.96	398
9	0.92	0.92	0.92	397
10	0.96	0.93	0.94	399
11	0.97	0.86	0.91	396
12	0.64	0.84	0.72	393
13	0.87	0.83	0.85	396
14	0.92	0.88	0.90	394
15	0.80	0.93	0.86	398
16	0.76	0.88	0.82	364
17	0.98	0.80	0.88	376
18	0.82	0.60	0.69	310
19	0.81	0.55	0.66	251
accuracy			0.83	7532
macro avg	0.84	0.82	0.82	7532
weighted avg	0.84	0.83	0.83	7532

5. 多层感知机

```
请选择模型：1：贝叶斯，2：感知机，3：k近邻，4：支持向量机，5，多层感知机，0：退出
5
  0%|
1_score: 0.8330257198540865 accuracy_score: 0.8372278279341476
 33%|
1_score: 0.8350149984080565 accuracy_score: 0.8398831651619756
 67%|
1_score: 0.8375170814302304 accuracy_score: 0.8422729686670207
100%|
```

隐藏层为[100,100]时准确率最高，详细数据：

	precision	recall	f1-score	support
0	0.82	0.79	0.81	319
1	0.66	0.85	0.74	389
2	0.79	0.74	0.76	394
3	0.69	0.77	0.73	392
4	0.87	0.78	0.82	385
5	0.88	0.76	0.82	395
6	0.74	0.89	0.81	390
7	0.82	0.93	0.87	396
8	0.98	0.94	0.96	398
9	0.92	0.96	0.94	397
10	0.97	0.96	0.97	399
11	0.95	0.92	0.93	396
12	0.82	0.76	0.79	393
13	0.91	0.82	0.86	396
14	0.91	0.89	0.90	394
15	0.85	0.92	0.89	398
16	0.76	0.91	0.83	364
17	0.99	0.88	0.93	376
18	0.80	0.63	0.70	310
19	0.80	0.61	0.69	251
accuracy			0.84	7532
macro avg	0.85	0.84	0.84	7532
weighted avg	0.85	0.84	0.84	7532

【小结】：

本次实验实现了五种方法的文本分类，并且准确率尚可，出现的过拟合现象还需要进一步改进。

通过这次实验掌握了五种文本分类方法的代码实现，了解了词向量转化，对 python 编程能力和机器学习的理解更进一步。

指导教师评语及成绩

【评语】：

成绩： **指导教师签名：**

批阅日期：

附件：

实验报告说明

1．实验项目名称：要用最简练的语言反映实验的内容。要求与实验指导书中相一致。

2．实验目的：目的要明确，要抓住重点，符合实验任务书中的要求。

3．实验环境：实验用的软硬件环境（配置）。

4．实验方案设计（思路、步骤和方法等）：这是实验报告极其重要的内容。包括概要设计、详细设计和核心算法说明及分析，系统开发工具等。应同时提交程序或设计电子版。

对于**设计型和综合型实验**，在上述内容基础上还应该画出流程图、设计思路和设计方法，再配以相应的文字说明。

对于**创新型实验**，还应注明其创新点、特色。

5．结论（结果）：即根据实验过程中所见到的现象和测得的数据，做出结论（可以将部分测试结果进行截屏）。

6．小结：对本次实验的心得体会，所遇到的问题及解决方法，其他思考和建议。

7．指导教师评语及成绩：指导教师依据学生的实际报告内容，用简练语言给出本次实验报告的评价和价值。