新闻文本分类之传统机器学习

1． 特征提取

新闻文本分类的特征可以选取 tf,tf-idf,fasttext，wor2vec 等特征，本次主要使用 tf-idf。

（1）TF 是词频(Term Frequency)

词频（TF）表示词条（关键字）在文本中出现的频率。这个数字通常会被归一化(一般是词频除以文章总词数)，以防止它偏向长的文件。主要表示为：

公式：$$tf_{ij} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

即：$$TF_w = \frac{\text{在某一类中词条}w\text{出现的次数}}{\text{该类中所有的词条数目}}$$

其中 ni,j 是该词在文件 dj 中出现的次数，分母则是文件 dj 中所有词汇出现的次数总和；

（2） IDF 是逆向文件频率(Inverse Document Frequency)

逆向文件频率（IDF）：某一特定词语的 IDF，可以由总文件数目除以包含该词语的文件的数目，再将得到的商取对数得到。

公式：$$idf_i = \log \frac{|D|}{|\{j : t_i \in d_j\}|}$$

如果包含词条 t 的文档越少，IDF 越大，则说明词条具有很好的类别区分能力。

$$IDF = log(\frac{\text{语料库的文档总数}}{\text{包含词条}w\text{的文档数}+1}),\text{分母之所以要加}1\text{，是为了避免分母为}0$$

其中，|D| 是语料库中的文件总数。 |{j:ti∈dj}| 表示包含词语 ti 的文件数目（即 ni,j≠0 的文件数目）。如果该词语不在语料库中，就会导致分母为零，因此一般情况下使用 1+|{j:ti∈dj}|

（3）TF-IDF 实际上是：TF * IDF

某一特定文件内的高词语频率，以及该词语在整个文件集合中的低文件频率，可以产生出高权重的 TF-IDF。因此，TF-IDF 倾向于过滤掉常见的词语，保留重要的词语。主要实现如下：

```
#特征提取之一：tf
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer()
x=vectorizer.fit_transform(df_train['text']).toarray()

# x_test=vectorizer.fit_transform(text_test).toarray()

print(x.shape)
# print(x_test.shape)
```

```
(50000, 6171)
```

```
#特征提取之一：tf-idf
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_v=TfidfVectorizer(ngram_range=(1,3), max_features=5000,stop_words=["0","2","5","3750"])
x=tfidf_v.fit_transform(df_train["text"]).toarray()
print(x.shape)
```

```
(60000, 5000)
```

2． 分类器

分类器我采取了三个，一个是 navie bayes classifier，ridge classifier，linear discriminant classifer，评判标准为 f1 score。
代码如下：

```
#分类器一：navie bayes
from sklearn.naive_bayes import MultinomialNB
# from sklearn.ensemble import GradientBoostingClassifier#GBDT
from sklearn.metrics import f1_score
gnb = MultinomialNB()
gnb.fit(x_train, y_train)
y_pred=gnb.predict(x_test)
# gbm0 = GradientBoostingClassifier(random_state=10)
# gbm0.fit(x_train, y_train)
# y_pred=gbm0.predict(x_test)
f1=f1_score(y_pred,y_test,average='macro')
print(f1)
#特征维度5000，0.8092610171243083
#PCA+特征维度5000，0.8092610171243083
```

```
0.791156386636155
```

```
#分类器2：ridge classifier
from sklearn.linear_model import RidgeClassifier
from sklearn.metrics import f1_score

clf = RidgeClassifier()
clf.fit(x_train, y_train)
y_pred=clf.predict(x_test)
f1=f1_score(y_pred,y_test,average='macro')
print(f1)
#特征维度3000，0.89
#特征维度3500，0.8955901673803003
#特征维度4000,0.90
#特征维度4000+停用词 0.9023971628901889
#特征维度4500，0.9020835372956402
#特征维度4500+停用词 0.9023971628901889
#特征维度5000，0.9039927784596091
#PCA+特征维度5000，0.7896447449374182
```

```
0.9023971628901889
```

```
#分类器3：qda
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.metrics import f1_score

qda= QuadraticDiscriminantAnalysis()
qda.fit(x_train, y_train)
y_pred=clf.predict(x_test)
f1=f1_score(y_pred,y_test,average='macro')

print(f1)
```

打卡完毕!