新闻分类——朴素贝叶斯分类器

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。 在 sklearn 中，该模型有两种装载方式： 第一种是 sklearn.datasets.fetch_20newsgroups，返回一个可以被文本特征提取器（sklearn.feature_extraction.text.CountVectorizer）自定义参数提取特征的原始文本序列；第二种是 sklearn.datasets.fetch_20newsgroups_vectorized，返回一个已提取特征的文本序列，即不需要使用特征提取器。

导入工具包

```
#import sys
#from importlib import reload
import matplotlib.pyplot as plt
#reload(sys)
plt.rcParams['font.sans-serif'] = ['Arial Unicode MS']
#sys.setdefaultencoding('utf-8')

import pandas as pd
import numpy as np
from sklearn import metrics

import matplotlib.pyplot as plt
%matplotlib inline
```

读取数据

```
from sklearn.datasets import fetch_20newsgroups

twenty_news = fetch_20newsgroups()
y = twenty_news.target
X = twenty_news.data


from sklearn.feature_extraction.text import TfidfVectorizer

# 初始化 TFIV 对象，去停用词，加 2 元语言模型
tfv = TfidfVectorizer(min_df=3,  max_features=None, strip_accents='unicode', analyzer='word',token_pattern=r'\w{1,}', ngram_range=(1, 2), use_idf=1,smooth_idf=1,sublinear_tf=1, stop_words = 'english')

# 提取特征，会有点慢
X = tfv.fit_transform(X)
#将数据分割训练数据与测试数据
```

```
from sklearn.model_selection import train_test_split

# 随机采样 20%的数据构建测试样本，其余作为训练样本
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=3
3, test_size=0.2)
X_train.shape
print(np.max(y_train))

X_test.shape
```

模型训练

```
# 多项朴素贝叶斯
from sklearn.naive_bayes import MultinomialNB

MNB = MultinomialNB()
MNB.fit(X_train, y_train) #特征数据直接灌进来
```

测试

```
#输出每类的概率
#y_test_pred = MNB.predict_proba(X_test)
y_test_pred = MNB.predict(X_test)
```

性能

```
print(metrics.classification_report(y_test, y_test_pred, target_names=tw
enty_news.target_names))
#print(metrics.confusion_matrix(y_test, y_test_pred))
```

```
print(metrics.accuracy_score(y_test, y_test_pred))
```

```
# 和逻辑回归比较
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV

#设置超参数搜索范围
penaltys = ['l1','l2']
Cs = [ 0.1, 1, 10, 100, 1000]
tuned_parameters = dict(penalty = penaltys, C = Cs)
```

```
# LR 学习器实例
lr_penalty= LogisticRegression(tol=0.0001)

#GridSearchCV 实例
grid= GridSearchCV(lr_penalty, tuned_parameters, cv=5, scoring='neg_log_
loss',n_jobs = 4,verbose=5)

# 模型训练
grid.fit(X_train,y_train)

#输出结果

grid.grid_scores_
grid.best_estimator_
```

最佳超参数在搜索范围边界，扩大超参数搜索范围

```
#设置超参数搜索范围
penaltys = ['l1','l2']
Cs = [10000,100000]
tuned_parameters = dict(penalty = penaltys, C = Cs)

# LR 学习器实例
lr_penalty= LogisticRegression(tol=0.0001)

#GridSearchCV 实例
grid= GridSearchCV(lr_penalty, tuned_parameters, cv=5, scoring='neg_log_
loss',n_jobs = 4,verbose=5)

# 模型训练
grid.fit(X_train,y_train)

#输出结果
grid.grid_scores_
```

测试

```
y_test_pred = grid.best_estimator_.predict(X_test)
print(metrics.classification_report(y_test, y_test_pred, target_names=tw
enty_news.target_names))
print(metrics.accuracy_score(y_test, y_test_pred))
```