基于支持向量机、随机森林与 XGBoost 的推特情感分类

——统计机器学习作业一

2019101404 马正一

1 简介

Twitter 是世界上最大的线上社交软件之一。公司与企业可以通过在 Twitter 上搜索与爬取与自己公司相关的推特正文，利用文本分析，自然语言处理技术进行分析，达到监测舆情，维护企业形象的目的。情感分类是自然语言处理中中重要的研究课题之一，随着机器学习、深度学习技术的发展，在日常生活中被广泛使用，极大地便利了人们的日常生活。情感分类也是舆情监测的关键技术之一。通过智能的情感分类技术，我们可以以较高准确率的模型自动识别一段文本的情感极性，从而在大数据上对某一关注实体的舆情做出判断，进而做出相关决策维护舆情。

在本实验中，我们使用 Kaggle 上提供的 Twitter US Airline Sentiment Dataset（美国航空公司推文情感数据集），使用本门课程目前为止学习到的 SVM、Random Forest、XGBoost 等机器学习方法训练情感分类模型，使用交叉验证的方法进行实验，调整超参数，记录模型训练结果，分析结果并比较模型，在实验中加深对学习到的机器学习算法的理解。为了避免复杂的特征工程，更加聚焦分类模型算法，同时尽量提高模型效果，我们使用了 BERT 预训练模型作为句编码器，将每段推文的 BERT Sentence encoding 作为样本的训练特征。实验结果表明我们训练的三个模型效果较好。我们将程序源代码与运行手册上传至 github： 链接。

2 数据集

在本实验中，我们使用 Kaggle 上的开源数据集 Twitter US Airline Sentiment Dataset。该数据集包括 14870 条 Twitter 推文，具体字段包括正文、人工标注的情感极性，负面原因，推文日期等。数据集总共包括了美国六大航空公司的相关推文。每条推文的情感极性可以为正面（Positive）、中性（Neutral）或负面（Negative）。数据集的情感极性分布如图一所示。数据集以 CSV 和 sqlite 两种格式存储，我们在本实验中的代码以 CSV 格式进行数据读入。数据集以附件的形式提交。我们给出数据集中的一条示例样本：

570300616901320704,positive,0.6745,,0.0,Virgin America,,cjmcginnis,,0,"@VirginAmerica yes, nearly every time I fly

VX this “ear worm” won’t go away :)",,2015-02-24 11:13:57 -0800,San Francisco CA,Pacific Time (US & Canada)

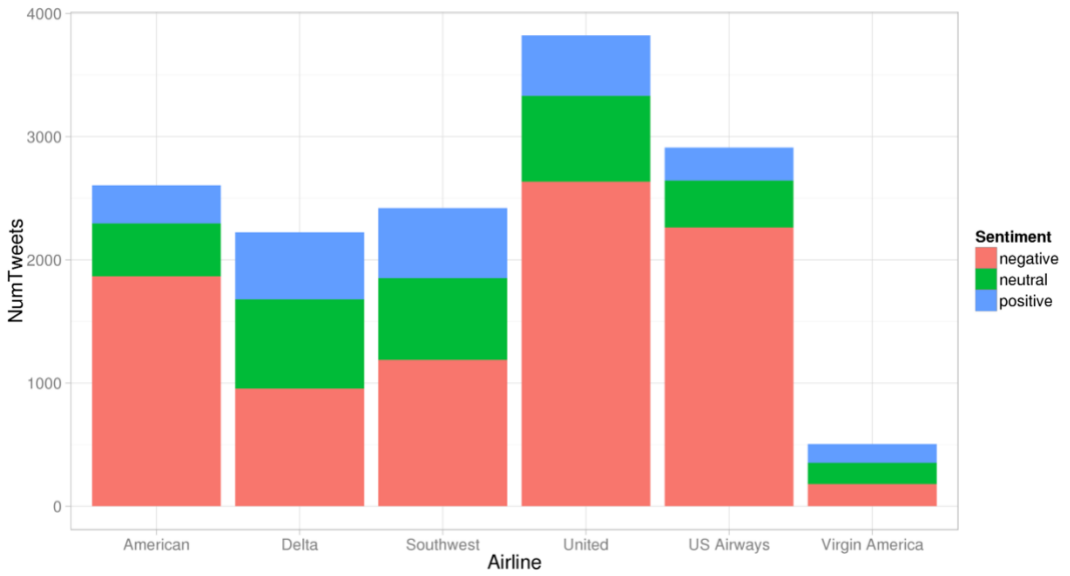

图 1，数据集情感极性分布

3 基于 BERT 的特征工程

在本实验中，为了避免复杂的特征工程，更加聚焦分类模型算法，同时尽量提高模型效果，我们使用 BERT 来处理每条推特正文，直接得到正文的高维分布式句子编码向量。换句话说，我们将每条推文的 BERT sentence encoding 作为特征进行我们的模型训练。值得注意的是，我们并没有对 BERT 进行常见的 Fine-tuning，而是用一种相对“静态”的方式使用 BERT 来进行特征工程。

我们首先使用 Anaconda 进行 Package 管理，安装本实验所需的 Python 环境，主要包括 Scikit-learn，Tensorflow，Numpy，Pandas，BERT 的相关 Package 等。

```
conda create -n tweet_sentiment -c anaconda python=3.7 scikit-learn tensorflow tensorflow-gpu

conda activate tweet_sentiment

pip install xgboost pandas bert-serving-server bert-serving-client
```

之后我们去 BERT 的 Github 官方仓库下载 BERT 模型文件。在本实验中我们选择了 BERT 最新发布的 BERT-Large, Uncased (Whole Word Masking)版本作为模型文件。我们使用以下命令部署 BERT 服务端，以 4 线程方式开启 BERT 服务，供之后的客户端取句子编码向量。由于推文有字数限制，因此我们设置 BERT 的 MAX_SEQ_LEN 为 256 即可。

```
mkdir /tmp/bert_models
```

```
unzip -d /tmp/bert_models/wwm_uncased_L-24_H-1024_A-

16.zip # or another model zip file you download

bert-serving-start -model_dir /tmp/bert_models/wwm_uncased_L-24_H-1024_A-16/ -num_worker=4 -

max_seq_len 256
```

接着我们编写代码，遍历数据集中的每个推特正文，将推特正文输入至 BERT 服务中，得到该推特正文的 1024 维编码向量，以 numpy 格式存储至本地，作为接下来机器学习模型的特征准备输入。

```
from bert_serving.client import BertClient

import pandas

import numpy as np



bc = BertClient("183.174.228.116")

Tweet= pandas.read_csv("./data/tweets.csv")

vecs = bc.encode(Tweet['text'].tolist())

vecs = np.array(vecs)

np.save("textvecs.npy",vecs)
```

4 数据划分

为了支持实验部分的 K-fold 交叉验证，我们首先进行数据的划分。我们使用 numpy 随机生成数列，以此将数据的 80%划分为训练集，10%划分为验证集，10%划分为测试集。由于我们每次跑实验都使用随机数据划分，因此数据中的每个样本都有机会进入训练集、验证集或开发集，保证了模型实验结果的可靠性，使我们的模型不会对数据划分过分敏感。

```
def divide_dataset(X,Y):

    shuffle_list = [i for i in range(0,len(X))]
```

```
np.random.shuffle(shuffle_list)

X_train = [ X[shuffle_list[i]] for i in range(int(len(shuffle_list) * 0.8))]

Y_train = [ Y[shuffle_list[i]] for i in range(int(len(shuffle_list) * 0.8))]



X_dev = [ X[shuffle_list[len(X_train)+j]] for j in range(int(len(shuffle_list) * 0.1)) ]

Y_dev = [ Y[shuffle_list[len(X_train)+j]] for j in range(int(len(shuffle_list) * 0.1)) ]



X_test = [ X[shuffle_list[k]] for k in range(len(X_train)+len(X_dev),len(shuffle_list)) ]

Y_test = [ Y[shuffle_list[k]] for k in range(len(X_train)+len(X_dev),len(shuffle_list)) ]



return X_train,Y_train,X_dev,Y_dev,X_test,Y_test
```

5 模型算法

在本实验中，我们的情感分类问题本质上是一个三分类问题，标签集为“积极”、“消极”与“中性”。我们需要对输入的每个推文的高维编码向量进行分类。我们分别实现了支持向量机、随机森林与 XGBoost 三种分类算法，使用 Scikit-learn 作为开发环境。

5.1 支持向量机（SVM）

我们使用 Scikit-learn 中提供的支持向量分类器类（SVC）来实现 SVM 分类。SVC 定义及所需的超参数表示如下：

class sklearn.svm.SVC(*C=1.0, kernel='rbf', degree=3, gamma='auto_deprecated', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=1, decision_function_shape='ovr', random_state=None*)

在本实验中，我们以 C（惩罚因子）、gamma 两个参数作为待调超参数，来进行超参数选择。

```
def SVM_MODEL(X_train,Y_train,X_dev,Y_dev,X_end,Y_end):
    clf = svm.SVC(C=5, gamma=0.05,max_iter=-1)
    clf.fit(X_train, Y_train)
    Y_pred = clf.predict(X_dev)
    precision = sum(Y_pred == Y_dev)/len(Y_dev)
    print('dev precision: ', precision)
    Y_pred = clf.predict(X_test)
    precision_test = sum(Y_pred == Y_test)/len(Y_test)
    print('test precision: ', precision_test)
```

5.2 随机森林（Random Forest）

我们使用 Scikit-learn 中提供的随机森林分类器（RandomForestClassifier）来实现随机森林分类。随机森林的定义及所需的超参数表示如下：

*class sklearn.ensemble.*RandomForestClassifier*(n_estimators='warn', criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None)*

在本实验中，我们以 n_estimators（森林中树的个数），max_depth（树的最大深度）作为待调超参数，进行超参数选择。

```
def RF_MODEL(X_train,Y_train,X_dev,Y_dev,X_test,Y_test):
    clf = RandomForestClassifier(n_estimators=500, max_depth=32,random_state=8)
```

```
    clf.fit(X_train, Y_train)

    Y_pred = clf.predict(X_dev)

    precision = sum(Y_pred == Y_dev)/len(Y_dev)

    print('dev precision: ', precision)

    Y_pred = clf.predict(X_test)

    precision_test = sum(Y_pred == Y_test)/len(Y_test)

    print('test precision: ', precision_test)
```

5.3 XGBoost

我们使用 XGBoost Package 与 Scikit-learn 提供的 XGBClassfier 实现 XGBoost 分类器。XGBClassfier 的定义及所需的超参数表示如下：

class *xgboost.*XGBClassifier*(*max_depth=3, learning_rate=0.1, n_estimators=100, verbosity=1, objective='binary:logistic', booster='gbtree', tree_method='auto', n_jobs=1, gpu_id=-1, gamma=0, min_child_weight=1, max_delta_step=0, subsample=1, colsample_bytree=1, colsample_bylevel=1, colsample_bynode=1, reg_alpha=0, reg_lambda=1, scale_pos_weight=1, base_score=0.5, random_state=0, missing=None, **kwargs*)*

在本实验中，我们以 learning_rate，max_depth，n_estimators 作为待调超参数，进行超参数选择。

```
def XGBOOST_MODEL(X_train,Y_train,X_dev,Y_dev,X_test,Y_test):

    clf = xgb.XGBClassifier(objective="multi:softprob",random_state=42,learning_rate=0.05,max_depth=8
,n_estimators=100,subsample=0.5,colsample_bytree=0.5)

    clf.fit(np.array(X_train), np.array(Y_train))

    Y_pred = clf.predict(np.array(X_dev))

    precision = sum(Y_pred == Y_dev)/len(Y_dev)
```

```
print('dev precision: ', precision)

Y_pred = clf.predict(np.array(X_test))

precision_test = sum(Y_pred == Y_test)/len(Y_test)

print('test precision: ', precision_test)
```

6 参数选择与模型分析

6.1 参数选择

我们使用 K-Fold 交叉验证法进行参数的选择。对每组候选超参数组合，我们设定 K=5，即对执行 5 次训练，并在验证集上取准确率平均值作为评价指标。表 1-3 给出了三个模型在不同超参数设定下在验证集的准确率结果。

表 1：SVM Model 超参数选择结果表

超参数组合（C,gamma）	验证集上准确率平均值（%,K=5）
（1,0.05）	84.2
(2,0.05)	84.9
(3,0.05)	85.3
(5,0.05)	**88.2(*)**
(10,0.05)	86.4
(20,0.05)	85.2
(5,0.001)	85.1
(5,0.01)	87.2
(5,0.03)	87.5
(5,0.1)	86.7
(5,0.2)	87.0
(5,0.5)	86.2
(5,1)	86.4

根据交叉验证实验结果，我们选取（5,0.05）作为 SVM 模型超参数。

表 2：Random Forest Model 超参数选择结果表

超参数组合 （n_estimators,max_depth）	验证集上准确率平均值（K=10,%）
（10,32）	71.9
(50,32)	75.2

(100,32)	76.0
(200,32)	76.1
(500,32)	**76.6(*)**
(1000,32)	76.5
(100,4)	73.1
(100,8)	74.0
(100,16)	74.3
(100,64)	75.2
(100,128)	75.3

根据交叉验证实验结果，我们选取（500，32）作为 Random Forest 模型超参数。

表 3：XGBoost Model 超参数选择结果表

超参数组合（learning_rate，max_depth，n_estimators）	验证集上准确率平均值（K=10,%）
（0.001,32,100）	78.4
(0.005,32,100)	78.2
(0.01,32,100)	78.1
(0.05,32,100)	**79.4(*)**
(0.1,32,100)	76.3
(0.5,32,100)	73.5
(0.05,4,100)	72.5
(0.05,8,100)	72.9
(0.05,16,100)	74.0
(0.05,64,100)	79.3
(0.05,128,100)	78.9
(0.05,32,20)	76.4
(0.05,32,50)	77.0
(0.05,32,200)	79.3
(0.05,32,500)	79.4

根据交叉验证实验结果，我们选取（0.05,32,100）作为 XGBoost 模型超参数。

6.2 模型分析

在本实验中，我们实现了 SVM、Random Forest、XGBoost 三种机器学习分类模型，并在实际数据集上进行实验。实验结果表示，SVM 在本次实验情感分类任务上表现最佳，其次为 XGBoost 算法，最后是 Random Forest 算法。

对于 SVM 算法，该算法可以达到较高的准确率，这为解决过拟合问题提供了很好的理论保证，并且即便数据在原始特征空间里线性不可分，只要我们对 SVM 模型指定一个恰当的核函数，该模型便能取得较好的效果。因此，SVM 适合解

决维度较高的文本分类问题。也正因如此，在我们本次的情感分类任务中，SVM 表现优于其他模型。SVM 的缺点在于需要的内存空间消耗较大，可解释性较差。综上所述，SVM 的优点在于：（1）适合解决高维度问题，即大型特征空间。（2）能够处理非线性特征的相互作用。（3）无需依赖整个数据。（4）泛化能力较强。缺点在于：（1）训练样本较高时的效率较低。（2）对非线性可分的数据没有通用解决方案，需要人工找一个恰当的核函数。

对于随机森林算法，该算法是一个易于解释的算法。随机森林算法天然适用于非参数化地处理特征间的交互关系，因此使用该模型时无需担心数据中的异常值以及数据的线性可分性。随机森林算法的优点在于：（1）计算简单，可解释性强。（2）适合解决有缺失属性的样本数据。（3）能够处理不相关的特征。（4）效率较高，能够在大型数据源上运行出可行且效果良好的结果。缺点在于：（1）不适合在线学习，对于新数据需要对整个决策树推倒重建。（2）忽略了数据间的相关性。

XGBoost 号称“机器学习竞赛中的大杀器”。该算法准确率较高，并发性较高，效率较高，并且支持自定义损失函数，同时既可以处理回归问题也可以处理分类问题。同时该算法可以像随机森林一样输出特征重要性，适合处理高维特征，适合处理文本相关问题。同时，我们可以在 XGBoost 算法中加入正则性，控制了模型的复杂程度，可以有效缓解过拟合。同时，该算法支持列抽样，随机选择特征，增强了模型的稳定性。该算法同时对缺失值不敏感，可以学习到包含缺失值的特征的分裂方向。

实验结果

我们使用最终选取的超参数，在测试集上运行我们训练的模型，实验结果如下：

模型	SVM	Random Forest	XGBoost
准确率	88.4	77.0	80.3