# COMP9318 Project Fool classifier

Student1: Li Yu z3492782

Student2: Bowen Zhou z5127532

## Abstract

This project implemented the function fool_classifier which first used train data set to obtain a super svm linear classifier to get most important features and then modified test data set (change it from class 1 to class 0). In this project, all the train data set (360 class 0 and 180 class 1) have been used and all features (5718 single features) appear are keeped. The performance of fool the target classifier has been improved from 79.5% to 90.5% by using AdaBoost method and TF-IWF.

## 1. Introduction :

The aim of this project is to use test_data text file and generate a modified_data text file to fool the target classifier. Inorder to change 20 features in test_data text file, a local strong classifier has been implemented to obtain most important features for two different classes.

## 2. Process for feature extraction:

### 2.1 Tf-Iwf

Text files are actually series of words. In order to run machine learning algorithms it need to convert the text files into numerical features vectors.

For text classification, mainly two methods used for extracting features from text files in the machine learning:

1. TF : Term frequency
2. TF - IDF : Term Frequency times inverse document frequency

However both these two method can not achieve a decent accuracy when predict test_data. TF hits a higher accuracy(53%) than TF-IDF(43%) when using linear model.

TF-IDF is a common method for information retrieval which has many disadvantages for text classification:

1. TF-IDF undervalues the term can represent the characteristic of the documents of this class if it only frequency appears in the documents belongs to the same class while infrequently in the documents of the other class.
2. TF-IDF neglects the relations between the features and the class.

TF- IWF stands for Term Frequency - Inverse Word Frequency. It is based on TF and add a logarithm of the reciprocal of one word frequency in all document to replace IDF.

$$TF-IWF_{i,j} \to TF_{i,j} \times IWF_i = \frac{n_{i,j}}{\sum_k n_{k,j}} \times \log \frac{\sum_{i=1}^{m} nt_i}{nt_i}$$

This method can reduce the importance of some common words appear  frequently in both classes than TF, such as “the”, “that”, “.” and it also scale up the rare words which stand for the its own class.

## 2.2 Stop words

Stops word are important for classification in this specific training data set. Same experimental method with one submission remove stops word, the submission accuracy drop from the 79.5% (the first submission accuracy) to 51% (the second submission accuracy).

Furthermore, the table below shows 20 most important features for each class. Some stop words and punctuations showed strong influence in text classification.

| Class 0: (coef_value, feature) | Class 1: (coef_value, feature) |
|---|---|
| (-1.9155216902382541, 'iraq') | (3.9470084505776346, 'mr.') |
| (-1.8889028737432318, 'of') | (3.578183242615772, 'president') |
| (-1.6457950940330959, 'more') | (2.1080146784534617, ',') |
| (-1.4829682218161095, 'three') | (1.9204463042228126, '(') |
| (-1.385938329169019, 'nuclear') | (1.5712903056217882, ')') |
| (-1.3554791517476845, 'than') | (1.2118866543725555, 'russian') |
| (-1.3209031956478656, 'the') | (1.2050458126080748, 'border') |
| (-1.2778104790525637, 'bomb') | (1.1847522498936154, 'hold') |
| (-1.2711321400684263, 'it') | (1.0922080891951886, 'about') |
| (-1.1456224381958362, 'support') | (0.9909113813464278, 'tell') |
| (-1.1100071870325712, 'city') | (0.988358858149318, 'also') |
| (-1.109754789359076, 'national') | (0.9524513023091911, 'minister') |
| (-1.078801961682577, 'force') | (0.9512950512651976, 'rice') |
| (-1.009651886116594, '/') | (0.9240388099290333, 'confirm') |
| (-0.9810821840151156, 'be') | (0.9192669180289094, 'a') |
| (-0.9623757528270855, 'fail') | (0.9098928426400484, 'any') |
| (-0.9573126868362886, 'iran') | (0.9054390773822859, 'opposition') |
| (-0.9518654804374412, '1996') | (0.8969487359684656, 'back') |
| (-0.9251602203994443, 'into') | (0.8960186631685922, 'team') |
| (-0.9095307189863515, 'afghanistan') | (0.8882533290187997, 'spokesman') |

## 3. Process of Experiment:

### 3.1 Data and methods for training

| | |
|---|---|
| Train Data set | 360 sample from class 0 and 180 samples from class 1 |
| Feature process method | TF - IWF |
| Training Model | AdaBoost method<br>SVM linear model<br>Bag of words (BOW) |
| Unique feature | 5718 |
| Word count | 54008 |
| Test data set | test_data.txt, 200 test sample from class 1 |

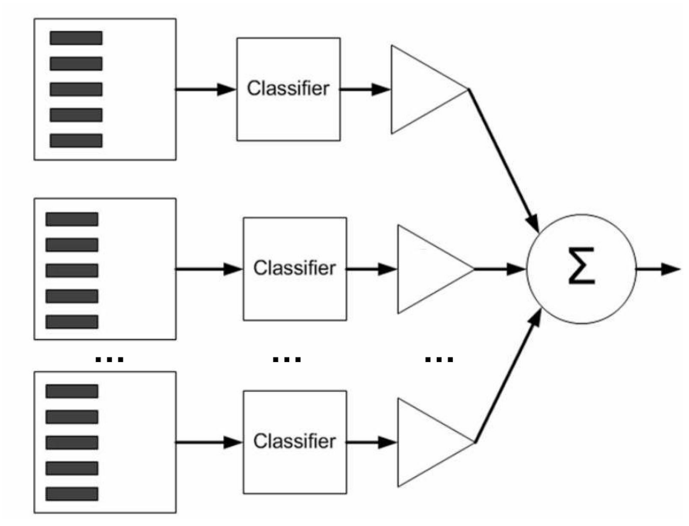
### 3.2 Tackle Imbalanced training data

Imbalanced data typically refers to a problem with classification problems where the classes are not represented equally. In this project , the text files in class 0 is twice larger than that in class 1 (360:180). By given a constraint that the training data sent into svm classfier will not exceed 541， sending 1:1 training data into classifer will imporve the accuracy of predicting test data set.

### 3.3 AdaBoost method improve the test accuracy for linear model

Using a single linear classifier can not obtain a decent accuracy when predicting test data. The highest test accuracy with a single classifier is 70% when implementing TF-IWF.

Adaboost in another ensemble classifier which is make up of multiple weak classifiers and whose output is combined result of output of those classifiers.
In this project , the Adaboost classifier combines 16 weak classifiers to form strong classifier. The difference between each classifier is the random training data set sending into them. Each single weak linear classifier learn part of the training data set. A final voting system is using when predict the test data set which can improve a good accuracy score for overall classifier. When it comes to find the most important features for two classes, the mean of all coefficients of all weak classifiers will represent the final coefficients of the super strong linear AdaBoost classifier.
The image below shows the main idea using AdaBoost method:



## 3.4 Select the parameter

| Kernel | linear |
|---|---|
| C | 4 |

After building the features extraction method and training model, selecting the parameter is the key to reach the higher performance.

Linear kernel is recommended for text classification by machine learning industry. There are 3 reasons for select linearn as a kernel:

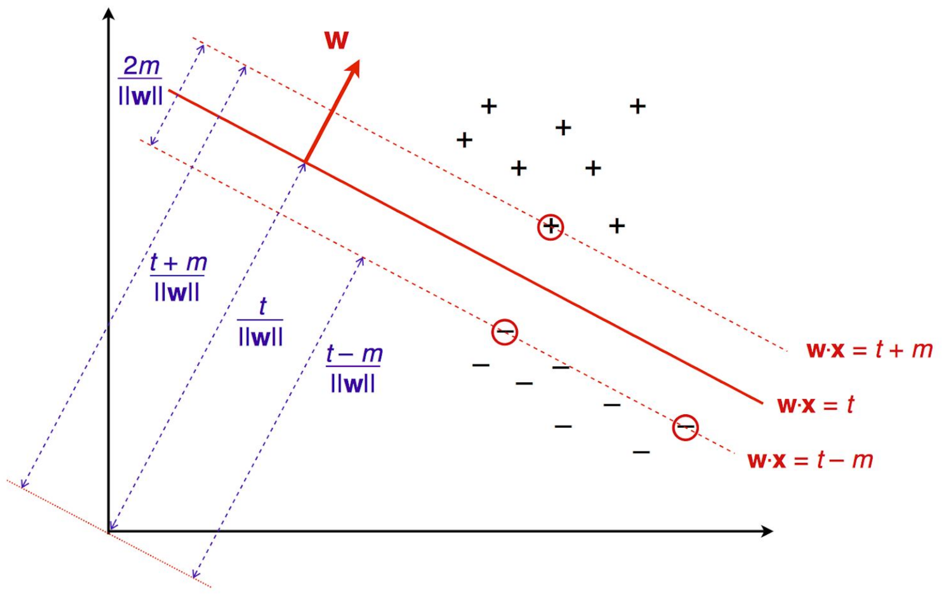
1. Text has a lot of features and linear is good when features numbers larger than training sample numbers (in this case 5718 features > 540 training samples).
2. Training a SVM with a linear kernel is faster than with another kernel.
3. Less parameters to optimize.
4. Text if often linearly separable.

The C parameter tells the SVM optimization how much you want to avoid misclassifying each training example.
sklearn.model_selection.GridSearchCV is used to finding the best C value. This table shows some experiments:

| C | Accuracy of test_data (voting_accuracy) |
|---|---|
| 1 | 0.73 |
| 2 | 0.765 |
| 3 | 0.785 |
| 4 | 0.825 |
| 5 | 0.825 |

## 3.4 Modify test data set

The coefficients of the strong linear classifier show the weights of hyperplane that separates two classes as best as possible. The  weight vector represent this hyperplane, by giving the coordinates of a vector which is orthogonal to the hyperplane. These are the coefficients given by classifier.coef_ in sklearn.



The weight (coefficients) vector give the predicted class. When predict one test sample, taking the dot product of any feature value of the test sample with this weight vector will tell which class this sample is belong to :

1. If the dot product is Positive: class 1
2. If the dot product is Negative: class 0

The absolute value of each coefficient relative to how important the feature was for separation. This table shows the top 20 features of each class.

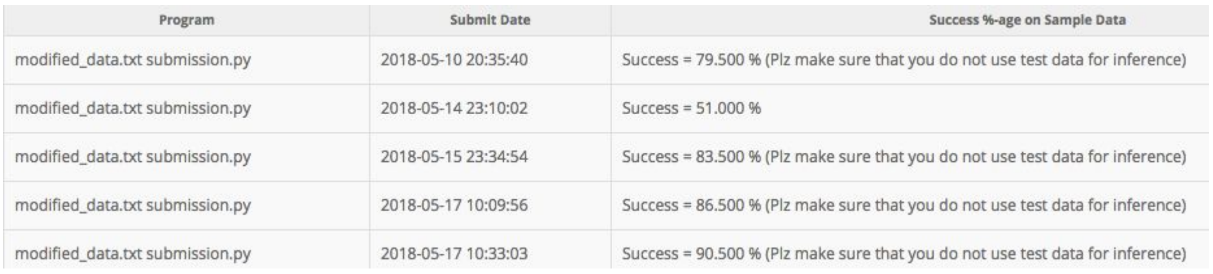
| Class 0: (coef_value, feature) | Class 1: (coef_value, feature) |
|---|---|
| (-1.9155216902382541, 'iraq') | (3.9470084505776346, 'mr.') |
| (-1.8889028737432318, 'of') | (3.578183242615772, 'president') |
| (-1.6457950940330959, 'more') | (2.1080146784534617, ',') |
| (-1.4829682218161095, 'three') | (1.9204463042228126, '(') |
| (-1.385938329169019, 'nuclear') | (1.5712903056217882, ')') |
| (-1.3554791517476845, 'than') | (1.2118866543725555, 'russian') |
| (-1.3209031956478656, 'the') | (1.2050458126080748, 'border') |
| (-1.2778104790525637, 'bomb') | (1.1847522498936154, 'hold') |
| (-1.2711321400684263, 'it') | (1.0922080891951886, 'about') |
| (-1.1456224381958362, 'support') | (0.9909113813464278, 'tell') |
| (-1.1100071870325712, 'city') | (0.988358858149318, 'also') |
| (-1.109754789359076, 'national') | (0.9524513023091911, 'minister') |
| (-1.078801961682577, 'force') | (0.9512950512651976, 'rice') |
| (-1.009651886116594, '/') | (0.9240388099290333, 'confirm') |
| (-0.9810821840151156, 'be') | (0.9192669180289094, 'a') |
| (-0.9623757528270855, 'fail') | (0.9098928426400484, 'any') |
| (-0.9573126868362886, 'iran') | (0.9054390773822859, 'opposition') |
| (-0.9518654804374412, '1996') | (0.8969487359684656, 'back') |
| (-0.9251602203994443, 'into') | (0.8960186631685922, 'team') |
| (-0.9095307189863515, 'afghanistan') | (0.8882533290187997, 'spokesman') |

The requirement of this project is to modify exactly 20 features in each sample in test data set. Method showed as below:

1. Find top 20 candidates which are most contributing features (in class 1) of each sample for test data set
2. From the candidates list as word1, pick one word from the other class (class 0) as word2, compare the abolute value of these two words.
   If abs(word1) >= abs(word2), remove word1
   If abs(word1) < abs(word2), remove word1 and add word2
3. Add with top contributing features (in class 0) which does not appear in this test sample.

## 3.5 Experimental outcome:

With efficient TF-IWF feature extraction method and the strong linear classifier supported by AdaBoost, the success rate has been improved gradually.



| Program | Submit Date | Success %-age on Sample Data |
|---|---|---|
| modified_data.txt submission.py | 2018-05-10 20:35:40 | Success = 79.500 % (Plz make sure that you do not use test data for inference) |
| modified_data.txt submission.py | 2018-05-14 23:10:02 | Success = 51.000 % |
| modified_data.txt submission.py | 2018-05-15 23:34:54 | Success = 83.500 % (Plz make sure that you do not use test data for inference) |
| modified_data.txt submission.py | 2018-05-17 10:09:56 | Success = 86.500 % (Plz make sure that you do not use test data for inference) |
| modified_data.txt submission.py | 2018-05-17 10:33:03 | Success = 90.500 % (Plz make sure that you do not use test data for inference) |

The difference between the success rate:

| Submission counter | Success %-age on sample data | Method |
|---|---|---|
| 1 | 79.5% | TF-IWF for features extraction<br>AdaBoost with 11 weak classifiers,each with random subset of training set |
| 2 | 51% | Remove stop words and punctuations |
| 3 | 83.5% | Increase parameter C from 3 to 4 |
| 4 | 86.5% | Add another 3 classifiers with non-random subset of training data set |
| 5 | 90.5% | Add another 2 classifiers with all training data set |

## 4. Conclusion

Linear model is ideal for text classification with TF-IWF using as the features extraction method. AdaBoost uses subsets of the original data to produce a series of averagely performing models then boosts their performance by combining them together using a voting system. The most contributing features for each each class of SVM linear kernel classifier are represented by the coefficients of the strong linear classifier obtained by AdaBoost method. These coefficients are the weight vector of the hyperplane and the most contributing features which has high absolute value is the key to modify test sample into the other class.

## 5. Reference

1. Improvement of Text Feature Selection Method Based on TFIDF S Qu, S Wang, Y Zou - Published in: Future Information Technology and Management Engineering, 2008. FITME '08. International Seminar on 2008 - ieeexplore.ieee.org
2. Adaboost https://en.wikipedia.org/wiki/AdaBoost