# Stock Market Prediction

Ping Xuanhuang 15307130283

November 17, 2017

## Contents



## 1 Process

Generally speaking, The whole procedure could be divided into 4 steps. The 1st step is to write the Naive Bayes classifier, the 2nd step is to dispose articles and generate dictionary, the 3rd step is to select feature and the last step is to do experimentation.

First, I write a class named as NBclassfier. It is a standard Naive Bayes classifier, which could deal with not only Binary Classification', but also multi-classification. The classifier can generate model by train set, do classification and store data of the model including the logarithmic probability of each class and the logarithmic probability of each feature in each class. The smoothing method it uses is 'add $\alpha$ smoothing' ($\alpha$ could be adjusted).

The function judge() can judge which label(class) a sample belongs to by finding out the maximum probability after adding up the logarithmic probability of each feature and class, it is based on a simplified version of naive Bayes formula(the denominator has been omited ). The function classify() could classify all the samples in the test set with the help of judge(), and if the test set has been labeled, function classify() can score for the classification result(provide the accuracy, recall, precision, F1).

Second, I use jieba to participle sentences in the news corpus and the result are stored in the form of dictionary (i.e. { 'id':...,'content':...,'title':... })(news_2.txt). Related code is in the Prepare.fenci.py. Then I counted the amount of each word in titles, and generated a dictionary for title which includes both the words appearing in the title and the frequency they appears, and the dictionary is sorted by the frequency. And of course, I do the same procedure on the contents.(title/content $dic\_freq$).

There I should mention the file Feature_ex.py, it can help main file to process train set and test set. It means that I can write a 'feature function' in the main file and apply this function on all the train set and test set with the help of function prepare_train_set().The Feature_ex.py could also apply different

feature function in the same sample, it means I can use different feature functions on title and content in the same sample with the help of prepare_train_set().

Third, to efficiently do classification(reduce the dimension of feature space) and to eliminate noise features, I used two used two methods two mark the importance of each word and sorted them by their importance. The first method is 'Mutual information'. It can evaluate the amount of information contributed by the occurrence of an event to the occurrence of another event. If one word is evenly distributes in every class, the mutual information of this word is 0, on the contrary, the higher the mutual information is, the more information the word could provide about a class. And I choose the max mutual information value of each to represent the importance of the word. The file Prepare.K2_MI is to calculate the mutual information of each word and to generate a dictionary, in which words are sorted by their mutual information value.(title/content $dic\_huxinxi$)

The second method I used to evaluate the importance of each word is $\chi^2$test(Chi - square test). It can test whether the two events(word and label) are independent. As above, I used the max $\chi^2$test value to present the importance of a word, and the file Prepare.K2_MI could calculated these value and generate a new dictionary.(title/content $dic\_k2$)

The next step is to start experimentation:

# 2 Experimentation

At this stage, I firstly selected features from either content or title and trained them by my own Naive Bayes classifier(NBclassifier) then I selected features in both content and title. And in each method, I changed the dimensions of feature to see how the amount of feature effect the result of the classification. After these experiments, I tried some different models to do classification.

First, I selected features in content. I did 3 groups of experiment, and used different dictionary(i.e. frequency dictionary, mutual information dictionary, chi-square dictionary) to select features, it means that in each group I choose one of these dictionary and continually extend the dimension of the feature space.

After selecting features, I will do cross validation first, so I separated the train set into 4 pieces randomly and made one of these pieces as validation set each time(the process will repeat 4 times in each experiment), then I will calculate the average score of the validation classification to see how efficient the features are. In the next step, I will train the model by all the train set and test the classifier by test set. Then I did the same thing on title. Figure 1 and Figure 2 shows the result (details could be found in File/data.xlsx).

From these data, I did some analysis. In this report, I will do analysis only based on the data of content, and the analysis about the data of title is similar.

From Figure3 (a), we could find there will be huge difference in the classification result when choose different dictionary to select feature, and it is obvious that use mutual information dictionary to judge the importance of word can make the best grade(When comes to the title dictionary, Chi-square dictionary does the best).

We could also find out that the accuracy of classification is related to the dimensions of the feature space, but we cannot say the more the dimensions is,

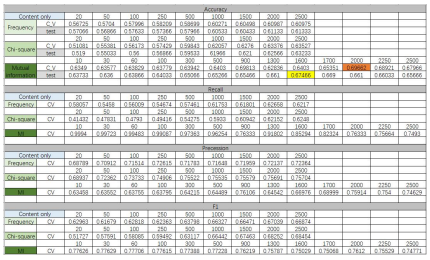


Figure1: Content only

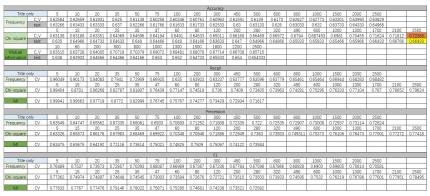


Figure2: Title only

the better the classification is. Now, notice the green line, it is the classification result based on $\chi^2$ dictionary, we could find the result of it is very bad at the beginning, however, as the dimensions increase, the grade of it soared quickly. It is because $\chi^2$test tents to give high value to words which appears only one or two times, so, it is not easy to find these features at the beginning, however, as the dimensions increase, this problem will disappear. And we could suppose that the score of Chi-square will keep increasing as the dimension increasing, however, limited by the memory size of my computer, I cannot do feather research. By the way, from Figure3 (d), we could also suppose that the score will keep going up as the dimensions increasing.

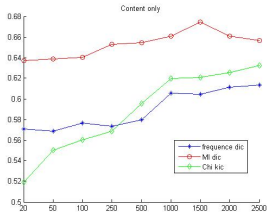
From Figure3 (b), generally speaking, the result tendency of validation and test set is similar, but it is obvious that blue lines don't show this conclusion, and we could say the variation of the result of the validation set is very huge, I think both of this two phenomenons is caused by the partition of the validation set, in other words, the validation set is very special, so we would better not collect information from it. By the way, From Figure3(b), we could also say the tendency of validation and test set is similar in title feature.

In Figure3 (c), blue lines are score of mutual information dictionary and red lines are score of frequency dictionary. We could find out that both of these 4 kinds of score of MI dictionary share the similar tendency, it means as the dimensions increase, comprehensive assessment is becoming better, however, when comes to the score of frequency, we could find the tendency of recall and the accuracy are different, I think it is because the classifier tent to classify sample as '+1'.

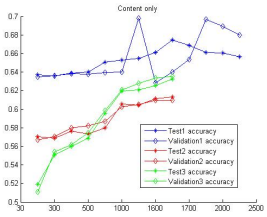
Then I tried to select feature from both title and content. To achieve it, I added 'title_ ' to the feature of title (i.e 'abc' will become 'title_abc'), the same as content. From analysis above, I use MI dictionary to select feature of content and Chi-square dictionary to select feature of title. Limited by the size of memory, I can only set the feature dimensions of content and title as (1000,15),(1000,20),(600,400),(500,500),(400,600)and the accuracy of them are 0.68633,0.6886,0.6926,0.6893, 0.692. So the result is better then just select feature from tittle or content, and I believe if I expend the dimensions of the feature, the result will be better.

Then I will use several classify models from nltk:

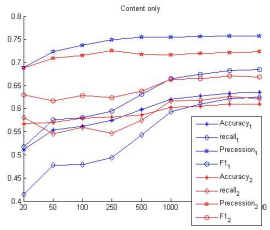
First, I tried the Naive Bayes classifier of nltk on the content only situation. And the dimensions are 100,500,2000, and the accuracy of Naive Bayes classifier of my own and nltk is (0.665,0.6403),(0.6783,0.6526)(0.6946,0.661). So we could find the accuracy of nltk and my own Naive Bayes classifier is difference, it is



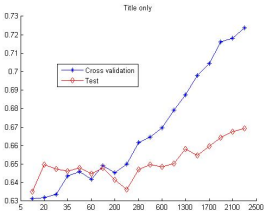

(a) Different dictionary




(b) Accuracy on cross validation & test set




(c) Score for classification




(d) Accuracy on title

Figure3: Analysis of data

because we use different smoothing methods.

Second, I tried the decision-making tree on the title only situation. And the dimensions are 15,20,40 and the accuracy is 0.64366, 0.64466, 0.65066

Third, I tried the Maximum entropy model on the title only situation. And the dimensions is and the accuracy is 35, and the accuracy is 0.65566.

# 3 Thinking and Summary

In the process of this project, I did nearly all the steps of text categorization, including pre process text, build dictionary, build classifier, select feature, cross validation and experimentation. I have to say I have learned a lot. But there is one thing I think I could improve. To pursue perfect, I write a standard Naive Bayes classifier, so I used many dictionary to store feature. It means I waste too much memory on it, so we can can see from above, the result is that I cannot do experimentation on large scale dimensions space. And if i could store feature in list, it will be better.