Introduction to Machine Learning Main assignment - Spam Filter

Boyang Zhao (bz16563), Haoyan Chen (hc16191)

Subtask 1 - simple Naive Bayes spam filter

1.1 Classifier construct and Feature process

The Simple Naive Bayes algorithm is constructed by ourselves in order to make improvement by some interesting ways in the task 3. Each word of the emails in the training list is considered as a feature candidate before feature preprocess in order to best use the Naïve Bayes algorithm theory.

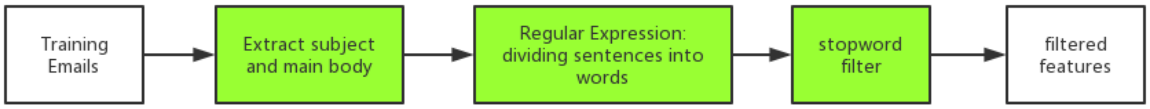

Training Emails
Extract subject and main body
Regular Expression: dividing sentences into words
stopword filter
filtered features

Figure 1

In the task one, we use Regular Expression to preprocess the data. As a result, all the characters except words are removed, which actually is able to simplify the feature list. Stopword reduction is also be processed in the feature preprocess and it can efficiently reduce the emergence of some meaningless nouns, pronouns or prepositions like “on”, “the”, “this”. We import the English Stopword list from python package “nltk” and combine it with the regular expression. This method actually reduces the feature list from around 29,000 to around 24,000 words.

In order to make the classifier more precise, selecting good features and removing redundant ones are more important. We found only the main content and subject of emails have an effect on distinguishing the Spam and Ham emails. The title and ending are almost set as a fixed format and this part of emails can easily generate some noisy features, so it is useless for us to count them as features part. Then we use the package email to process the text content read from emails and only extract the subject as well as the main body of the emails. This obviously reduces the length of feature list from 24,000 to about 22,000 and the time to train the classifier.

During the training, we realize some training samples are written in HTML. Hence after we read the content of the email, we first strip off the HTML tags and punctuation marks if they exit. Because some email samples are encoded by Base64, we need to decode them to extract normal English words. These processing methods result in a decrease of some noisy features and could significantly improve the classifier.

The training process will be taken about 30 minutes and the training result will be saved in files which can be used directly in the test process. The training result includes the prior positive probability, the feature frequency of the Spams and the Hams. It is efficient to extract them during testing.

1.2 Evaluation and Baseline Comparison

Cross-Validation is used as the method to evaluate the classifier. Standard deviation and accuracy are the metrics to calculate the results and show the performance. That is because we only have 500 training examples without any extra test set. Cross-validation can easily solve this problem and achieve a better performance. Accuracy is easy to visualize the test result and standard deviation is able to execute whether the classifier produce a stable performance.

During the implement, we set 10-folds for the evaluation according to the amount of the email training samples. A baseline was set for comparison with the simple Naive Bayes and results are shown in table 1 together.

The baseline is a classifier which will consider the email with few words as spam, because we found most spam emails' contents contain few words for advertisement or disturbance in order to attract people's attention. For example, an email with few words and a link for you to press will have a high probability to be a spam. In the training procedure, we would like to set a word count threshold to distinguish the spam and ham. The simple text preprocessing is also necessary because we only consider the word count in the main content and subject. If we set the Class spam as the positive, the accuracy predicted will be around 40%. Results are shown in the table 1.

Fold	Simple Naive Bayes	Baseline
1	0.8200	0.4000
2	0.8800	0.4000
3	0.8600	0.3800
4	0.8600	0.3400
5	0.9000	0.5400
6	0.8000	0.4800
7	0.7800	0.5800
8	0.8200	0.3400
9	0.9200	0.4600
10	0.8000	0.3600
avg	0.8440	**0.4280**
stdev	0.0470	**0.0839**

Table 1 Compare Naive Bayes Spam filter with Baseline

Obviously, the accuracy of simple naïve Bayes classifier is higher than that of baseline. The result achieved above indicates even the simple Naive Bayes classifier can produce a better accuracy to distinguish the emails. But as the average accuracy of the simple classifier is only 0.8440. There is also the space for improvement because the accuracy of the classifier is not very high.

Subtask 2 - Improve the simple Naive Bayes spam filter

2.1 Smart feature processing techniques

As subtask 1 mentioned, we build a Naive Bayes spam filter with some simple feature processing techniques, such as stopwords. However, a great number of features (around 23,500 features) including many repeated and useless words are generated by our single model. To solve this problem, two smart feature processing techniques - **Word stemming and Weighted Frequency and Odds (WFO)** - are applied to our model.

2.1.1 Word stemming – removes suffixes using Porter's algorithm

Words with a common stem usually have similar meanings, for example: CONNECT, CONNECTED, CONNECTING, CONNECTION, CONNECTIONS. To reduce the number of spam filter features, removing suffixes can help us to build a classifier with best features.

Our model is improved by a word stemming library implemented the Porter's algorithm for suffix stripping, which can delete those repeated terms and reduce the computational complexity.

2.1.2 Weighted Frequency and Odds (WFO)

The following two conclusions are drawn without doubt.

1) Good features are features with high document frequency;

2) Good features are features with high category ratio.

The WFO can reflect these two rules, which is defined as

$when \quad P(t|c_i)/P(t|\overline{c_i}) > 1$

$$WFO(t, c_i) = P(t|c_i)^{\lambda} \left[log \frac{P(t|c_i)}{P(t|\overline{c_i})}\right]^{1-\lambda}$$

$else$

$$WFO(t, \overline{c_i}) = P(t|\overline{c_i})^{\lambda} \left[log \frac{P(t|\overline{c_i})}{P(t|c_i)}\right]^{1-\lambda}$$

where $P(t|c_i)$and $P(t|\overline{c_i})$are the frequency measurements and $log \frac{P(t|c_i)}{P(t|\overline{c_i})}$and $log \frac{P(t|\overline{c_i})}{P(t|c_i)}$ are the logarithm of the ratio measurements.

Before training the Naive Bayes spam filter, we calculate the WFO value for each term in the corpus (simple preprocessed features), which can reflect how good the features are in spam or in ham. And then, those features whose WFO exceeds a preset threshold will survive. In this case, we need to set two parameters in advance: λ and threshold, and we will analyze how the classification results depend on these parameters later.

2.2 Experiment results

The following table 2 gives the results of evaluating our model with and without the feature techniques mentioned above on the 'public' data set with 10-fold cross-validation.

Fold	Without	With
1	0.8200	0.9000
2	0.8800	1.0000
3	0.8600	0.9400
4	0.8600	0.9600
5	0.9000	0.9600
6	0.8000	1.0000
7	0.7800	0.9800
8	0.8200	0.9400
9	0.9200	0.9600
10	0.8000	0.9600
avg	0.8440	**0.9600**
stdev	0.0470	**0.0298**

Table 2

* WFO parameters setting: $\lambda = 0.5$ and threshold = 0.01

The last two lines give the average and standard deviation of accuracy over all the ten folds. Obviously, compared with the model without smart feature selection techniques, the one with those techniques has a higher average accuracy and lower standard deviation, which shows that those techniques truly improve the performance and stability of our model.

At the same time, due to a large amount training features (about 22,000 in each fold), it takes about 30 minutes to run the model without feature selection techniques. However, the model with those techniques extracts fewer features (about 4,000), hence, it spends less time (around 18 minutes) on running 10-fold cross-validation because of lower computation complexity.

2.3 WFO parameter analysis

As 2.1.2 mentioned, two measurements are included in WFO: the frequency measurement and the logarithm of the ratio measurement. The $\lambda$is closely related to the importance of these two measurements. In this case, terms with both high category ratio and high document frequency will survive, therefore, we set λ equals 0.5.

We randomly select 100 emails including 80 hams and 20 spams as the testing data set to find the best threshold when λ equals 0.5.

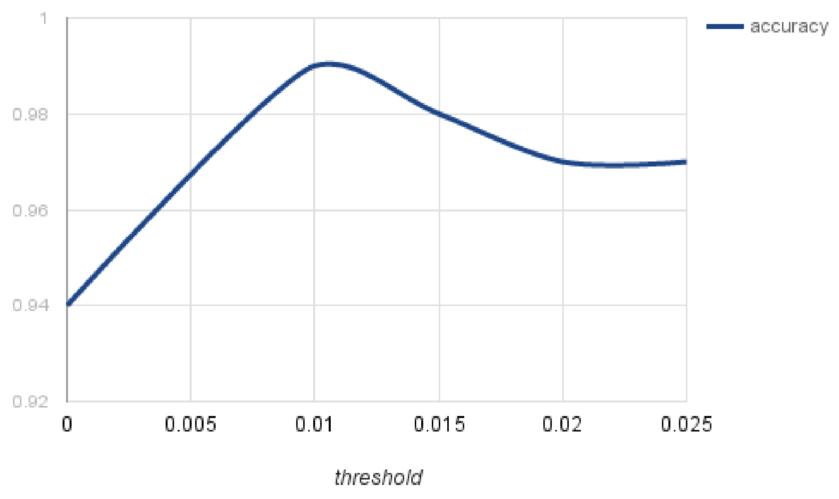


Figure 2

As shown in Figure 2, the accuracy achieves the peak at 0.98 when the threshold is 0.01. If the threshold is above 0.03, the number of features is too small (around 700) to provide enough features to make the prediction more precise. Therefore, we should preset the threshold as 0.01, otherwise, the WFO feature selection cannot work very well.

2.4 Compare with other algorithms

To compare the performance of different algorithms, we use scikit-learn libraries to build SVM and Decision Tree model, and then, these three models are trained by same features and tested by same 10-fold cross-validation data to ensure fair comparison.

False Positives are probably worse than False Negatives for this problem, because most of us are confused about some important emails in the spam folder. For this reason, we evaluate these three models by two metrics: accuracy and FPR.

	Accuracy avg / stdev		FPR avg / stdev	
Naive Bayes	**0.9600**	**0.0298**	**0.1023**	**0.0889**
SVM	0.9120	0.0473	0.1893	0.1165
Decision Tree	0.8780	0.0394	0.2777	0.1015

Table 3

* WFO parameters setting: $\lambda = 0.5$ and threshold = 0.01

As shown in table 3, Naive Bayes algorithm has the highest average accuracy (0.9600) and the lowest False Positive Rate (0.1023), and it is also the most stable one in these three models. SVM is the second suitable one for this problem in these three models, for its average accuracy and FPR are better than the Decision Tree model's, although SVM has a slightly poor stability.

Subtask 3 - Calibration and Naive Bayes extension

3.1 Probability calibration of classifier

3.1.1 DS Method

When we are evaluating a classifier, we want to predict not only the class label, but also the associated probability. In this section, the purpose is to reduce the Mean Squared Error (MSE) of our classifier. When we calculate the posterior probability, we add a new parameter DS to the original classification formula, which can reflect the degree of spamicity of each feature. And this new parameter is generated by Ada-Boost algorithm.

Original formula:

$$\mathrm{P}(\mathrm{spam}|\mathrm{w}) = \log \mathrm{P}(\mathrm{spam}) + \mathrm{sum}(\log\ \mathrm{P}(w_j|\mathrm{spam}))$$

Improved formula:

$$\mathrm{P}(\mathrm{spam}|\mathrm{w}) = \mathrm{logP}(\mathrm{spam}) + \mathrm{sum}(\log\ \mathrm{P}\left(w_j\middle|\mathrm{spam}\right) * DS_j)$$

Ada-Boost, short for Adaptive Boosting, is a meta-algorithm, and can be used in conjunction with many other learning algorithms to improve their performance. In this case, we tend to adjust the DS through many iterations and finally find the best value.

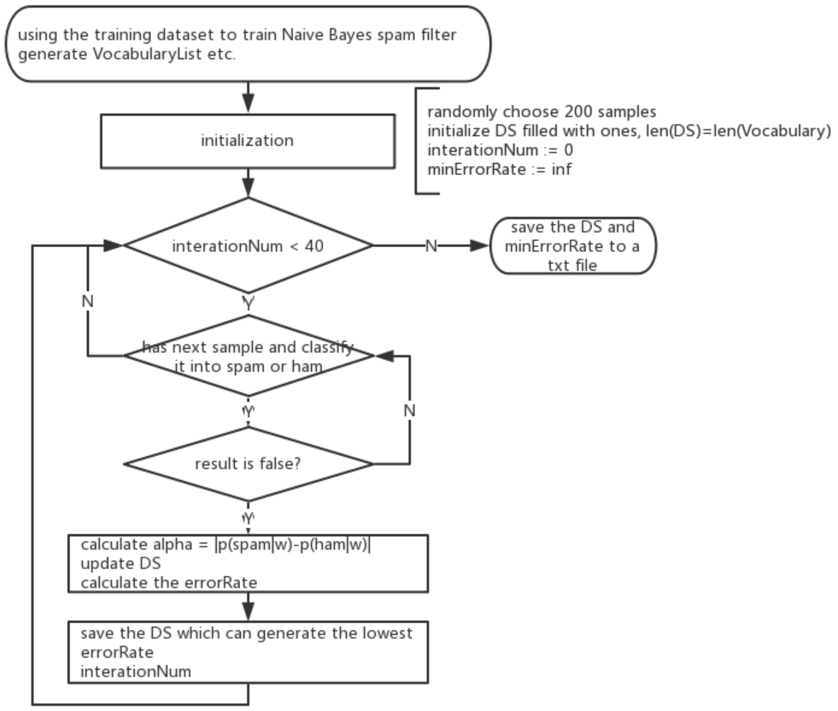

using the training dataset to train Naive Bayes spam filter generate VocabularyList etc.
initialization
randomly choose 200 samples
initialize DS filled with ones, len(DS)=len(Vocabulary)
interationNum := 0
minErrorRate := inf
interationNum < 40
N
save the DS and minErrorRate to a txt file
Y
N
has next sample and classify it into spam or ham
Y
N
result is false?
Y
calculate alpha = |p(spam|w)-p(ham|w)|
update DS
calculate the errorRate
save the DS which can generate the lowest errorRate
interationNum

Figure 3: Flowchart of calculating DS

As shown in Figure 3, DS is a vector initialized by ones, and we update DS when the classification result is false. The updating rules are below:

$alpha = |P(spam|w) - P(ham|w)|$

$when\ \ alpha > 0$:

$$\mathrm{DS[j]} = \mathrm{abs}\big((\mathrm{DS[j]} - \exp(\mathrm{alpha}))/\mathrm{DS[j]}\big)$$

$else$:

$$\mathrm{DS[j]} = \big(\mathrm{DS[j]} + \exp(\mathrm{alpha})\big)/\mathrm{DS[j]}$$

where j is the index of words which appear in the sample emails. At the end of this updating process, the best DS will be saved, and we can get DS directly at the testing process. The DS model helps us to reduce the MSE from 0.5624 to 0.4822.

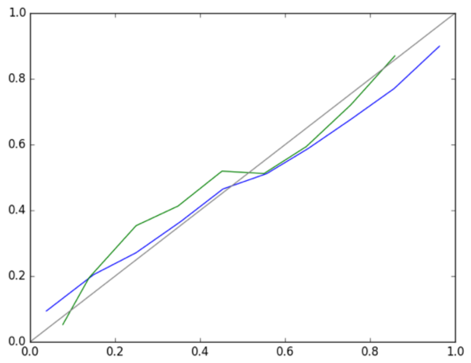
3.1.2 Platt scaling

Platt scaling is a well-known probability calibration method, which can be used in many types of classifier including Naïve Bayes classifier. The equation is below:

$$P(y = 1|x) = \frac{1}{1 + \exp(Af(x) + B)}$$

where A and B are the parameters calculated by maximum likelihood method that optimizes on the same training set as that for the original Naïve Bayes classifier.

After we apply this method talked above in our original classifier. We obtain this plot to show the performance improvement.



The green line represents the result after calibrated and the blue one represents the result before. We can draw the conclusion that the MSE which is 0.3573 is lower than before.

3.2 Modification

3.2.1 Graham's Method

This method focuses on another probability calculation way of Naive Bayes, which is shown below. It is also necessary to calculate the frequency of each word in Spam and Ham respectively. We consider $P(S|w_i)$ as the probability of Spam given by word w_i and $P(S|E)$ as the probability of Spam given by this Email. We denote Sw and Hw as the word's frequency of appearances in spam and ham respectively. From the equation examples below, it can calculate $P(S|E)$ and $P(H|E)$.

$$P(S|w_i) \quad = \quad \frac{Sw/Ssum}{Sw/Ssum + \quad 2 * Hw/Hsum}$$

$$P(H|w_i) \quad = \quad 1 - P(S|w_i)$$

$$\mathrm{P(S|E)} = \frac{\prod_{i=1}^{n} P(S|w_i)}{\prod_{i=1}^{n} P(S|w_i) + \prod_{i=1}^{n} P(H|w_i)}$$

The aim of the digit "2" in the first equation is to reduce false positives in the classification. After doubling the number of appearances in ham for each word in the email. we can find out which word is rarely used in legitimate email and also consider some of the tokens as not appearing in ham at all.

After using this method as the probability calculation way, the classifier avoids the situation that some Ham instances were misclassified into Spam in a kind. As the $P(H|w_i) \quad = \quad 1 - P(S|w_i)$, we set the threshold of probability as 0.9. That means the email will be classified into Spam if the $P(S|w_i)$ larger than 0.9.

We also use the cross-validation as the test method and the fold is 10. Accuracy and standard deviation are the metrics for performance evaluation. The result is shown below:

10-fold cross-validation	Simple Naïve Bayes	Graham's Method
avg	0.9600	0.9500
stdev	0.0298	0.0176

Table 4

The average accuracy is 0.9500 which is almost the same as that of the classified in last chapter, but it is also can be considered as a kind of interesting way to modify the Naïve Bayes classified because it really changes the way to obtain conditional probability. Although it is not necessary to replace the simple Naïve Bayes method proposed before with this Graham's Method, this can really give us another idea and expand our mind to classify the emails.

3.2.2 Unknown tokens

When testing the classifier, there will be a problem occur that maybe training set emails do not consist of some words in the test emails. Therefore, these words are impossible to find the P(w_i/S) or P(w_i/H) in the training result. We define these word as unknown tokens.

In order to improve the accuracy of the classifier, we shouldn't ignore the effect of these unknown tokens. But how to calculate their probability is a challenge. We found an interesting formula called Robinson's smooth method. The formula is based on an assumption that the classification of an email containing word w_i(unknown word) is a binomial random variable with a beta distribution prior. Under the big assumption that all words are independent to each other, this model clearly meets all the criteria for being a binomial experiment. And now we can assume a beta distribution Beta (α, β) for the prior.

$$P(S|w_i) \quad = \quad \frac{\alpha + Sp}{\alpha + \beta + Sp + Sw}$$

Two parameters are introduced to predict the probability of an unknown token:

C: the confidence level of our general knowledge

G: predicted probability of an unknown token being in a spam, based on our general knowledge of the word.

Normally, we set a starting value of 1 for C and a neutral probability of 0.5 as a starting point for$G$.

$$\alpha + \beta = C \quad \text{and} \quad \alpha = C * G \quad \text{so}$$

$$P(S|w_i) \quad = \quad \frac{C * G + Sp}{C + Sp + Sw}$$

Not all the unknown tokens can be considered, some useless words and characters must be eliminated to make sure the classifier maintain a stable performance and a high classification accuracy. Therefore, it is also necessary to preprocess the test emails before classifying them.

References

Korada, Naveen Kumar. "Implementation of Naive Bayesian Classifier and Ada-Boost Algorithm Using Maize Expert System." International Journal of Information Sciences and Techniques 2.3 (2012): 63-75. Print.

Porter, M.f. "An Algorithm for Suffix Stripping." Program 14.3 (1980): 130-37. Print.

Xiong, Zhong-Yang, Jian Jiang, and Yu-Fang Zhang. "New Feature Selection Approach (CDF) for Text Categorization." Journal of Computer Applications 29.7 (2009): 1755-757. Print.