A PRACTICAL SPAM CLASSIFIER BASED ON SUPPORT VECTOR MACHINE AND NA ÏVE BAYES MODEL

Hongzhi Liu[1], *Yaoshan Liu*[1], *Sikai Ni*[1], *Jing Zheng*[1]

[1] Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115

ABSTRACT

In this paper, a practical spam classifier was built with support vector machine. For a typical email, we first conduct preprocessing and map the content into a critical feature space. Then the training and testing of support vector machine is performed. In this project, 1000 training samples and 100 testing samples are introduced. We collate three kinds of kernel function (Gaussian, Polynomial, and Linear) accordingly, and explore best parameter for each function. We find that, for this spam classification problem, linear kernel has best performance since the feature space is exactly high-dimensional. Besides, we change the penalty parameter exponentially and find reasonable values. Finally, we build another classifier with Na ïve Bayes Model. Result shows that the SVM has better performance in this case.

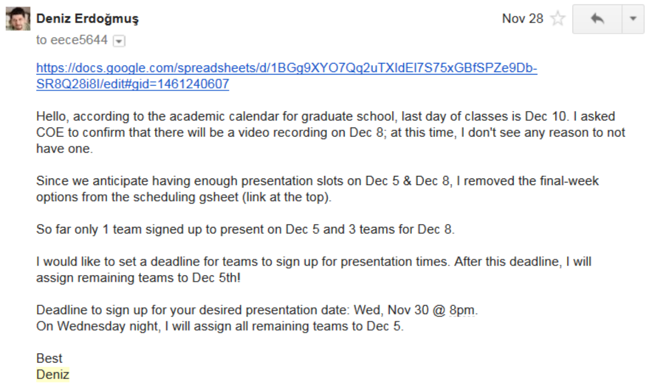
***Index Terms* —** Spam Classification, SVM, Parameter Selection, Na ïve Bayes Model

1. INTRODUCTION

Nowadays, many email services offer spam filters classifying emails into spam and non-spam with high accuracy. This technique, based on several machine learning algorithms, is acknowledged as an ideal application of text classification. In this paper, a typical Support Vector Machines (SVMs) is introduced to build a spam classifier.

2. PREPROCESSING EMAILS

Before conducting SVMs, the first thing we need to know is the exact features of each emails to be labeled. For human, it is nature of us to extract the information in texts and to justify whether the content is useful (i.e. spam or non-spam). However, this may be awkward for a computer to do so. In this section, we introduce a practical routine to solve this problem: normalizing the content of each emails, extracting corresponding features, and passing them to later training as well as testing cycles.



Deniz Erdoğmuş Nov 28

to eece5644

https://docs.google.com/spreadsheets/d/1BGg9XYO7Qq2uTXIdEI7S75xGBfSPZe9Db-SR8Q28i8I/edit#gid=1461240607

Hello, according to the academic calendar for graduate school, last day of classes is Dec 10. I asked COE to confirm that there will be a video recording on Dec 8; at this time, I don't see any reason to not have one.

Since we anticipate having enough presentation slots on Dec 5 & Dec 8, I removed the final-week options from the scheduling gsheet (link at the top).

So far only 1 team signed up to present on Dec 5 and 3 teams for Dec 8.

I would like to set a deadline for teams to sign up for presentation times. After this deadline, I will assign remaining teams to Dec 5th!

Deadline to sign up for your desired presentation date: Wed, Nov 30 @ 8pm.
On Wednesday night, I will assign all remaining teams to Dec 5.

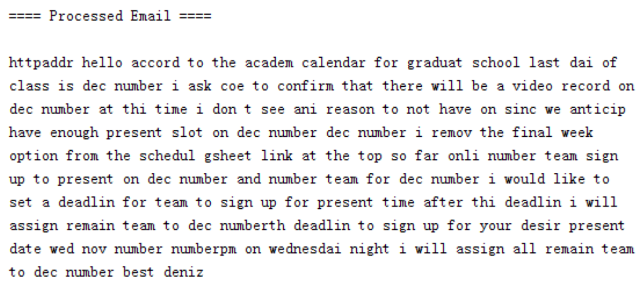
Best
Deniz

Fig. 1 Sample email needs to be classified

Fig. 1 shows a typical email needs to be classified. As we can see, the content includes many entities such as dates, URLs and specific words (e.g. presentation, graduate). Therefore, one method always employed in preprocessing section is to normalize each words into standard form so as to make them easier for subsequent extracting. For instance, when we found some URLs, although they have different lengths and addresses, this approach regards and replaces them as a same text "httpaddr". That is, we focus on whether any URLs are present, rather than whether a specific web address appears. Besides, there are also many other normalization steps are introduced in this preprocessing procedure:

- Lower-casing: convert entire email into lower case.
- Stripping HTML: remove all HTML tags.
- Normalizing URLs: replace all URLs with "httpaddr".
- Normalizing Email Addresses: replace email addresses with "emailaddr".
- Normalizing Numbers: replace numbers with "number".
- Normalizing Dollars: replace signs ($) with "dollar".
- Word Stemming: Words are reduced to their stemmed form.

The result of these steps is shown in Fig. 2, which is much easier to work with for following processes.



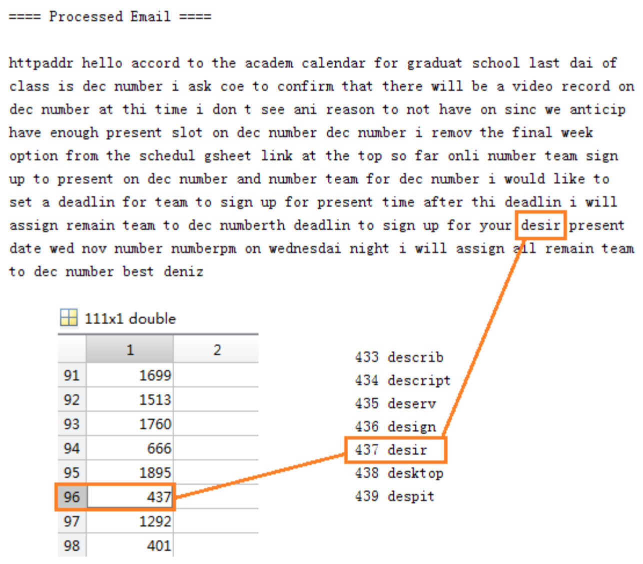
```
==== Processed Email ====

httpaddr hello accord to the academ calendar for graduat school last dai of
class is dec number i ask coe to confirm that there will be a video record on
dec number at thi time i don t see ani reason to not have on sinc we anticip
have enough present slot on dec number dec number i remov the final week
option from the schedul gsheet link at the top so far onli number team sign
up to present on dec number and number team for dec number i would like to
set a deadlin for team to sign up for present time after thi deadlin i will
assign remain team to dec numberth deadlin to sign up for your desir present
date wed nov number numberpm on wednesdai night i will assign all remain team
to dec number best deniz
```

Fig. 2 Normalized emails

Another issue is that we only want to include the most frequently occurring words in our classifier and ignore the least important words, since the rarely occurring words might cause the over-fitting of the training machine. To solve this problem, we introduce a vocabulary list containing 1899 most common words which occur at least 100 times in the spam corpus.

Given the vocabulary list, we can then map each preprocessed words into an indices that contains the index of the word in the vocabulary list. Fig. 3 shows the mapping of the sample email. In the picture, we can see that, the original word “desired” (we can find it in Fig. 1) is reduced to stemmed form and then normalized to “desir” which is included in our vocabulary list, and corresponds to the 437th index. Finally, the sample email was mapped to a word indices which includes 111 index.



```
==== Processed Email ====

httpaddr hello accord to the academ calendar for graduat school last dai of
class is dec number i ask coe to confirm that there will be a video record on
dec number at thi time i don t see ani reason to not have on sinc we anticip
have enough present slot on dec number dec number i remov the final week
option from the schedul gsheet link at the top so far onli number team sign
up to present on dec number and number team for dec number i would like to
set a deadlin for team to sign up for present time after thi deadlin i will
assign remain team to dec numberth deadlin to sign up for your desir present
date wed nov number numberpm on wednesdai night i will assign all remain team
to dec number best deniz
```

111x1 double

	1	2
91	1699	
92	1513	
93	1760	
94	666	
95	1895	
96	437	
97	1292	
98	401	

```
433 describ
434 descript
435 deserv
436 design
437 desir
438 desktop
439 despit
```

Fig. 3 Vocabulary list and word indices

Now for a given email, we have already mapped it into a unique word indices. The final process is reformat this indices as a feature vector. The justification to do so is to generate a sample space on which we will later train our SVM. Since the vocabulary list have 1899 common words, our sample space is 1899-dimensional. For each elements x_i of the feature vector, $x_i = 1$ means this word occur, and $x_i = 0$ otherwise. A typical feature vector is shown in Fig. 4.

$$x = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{R}^n.$$

Fig. 4 1899-dimensional feature vector

3. TRAINING SUPPORT VECTOR MACHINE

3.1. Support Vector Machine

After preprocessing, we convert each emails into a specific sample space whose dimensionality is determined by the vocabulary list. So the next step is training our machine. In machine learning, support vector machines (SVMs) are supervised learning models analyzing data for classification and regression [1].

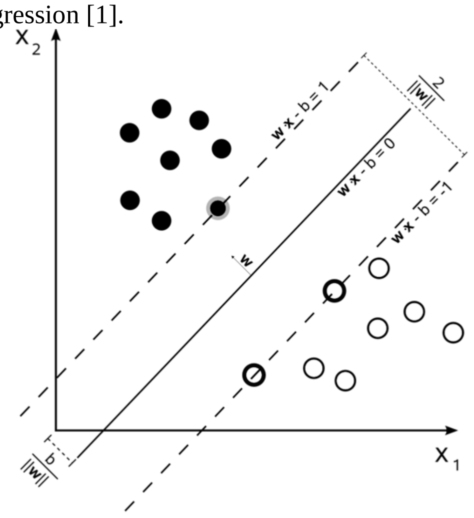

X_2
$\mathbf{w}\cdot\mathbf{x}-b=1$
$\frac{2}{\|\mathbf{w}\|}$
$\mathbf{w}\cdot\mathbf{x}-b=0$
$\mathbf{w}\cdot\mathbf{x}-b=-1$
$\mathbf{w}$
$\frac{b}{\|\mathbf{w}\|}$
X_1

Fig. 5 Support Vector Machine

Fig. 5 intuitively shows the idea of support vector machine. In this case, a data point is modeled as an *n*-dimensional vector, and we want to know whether we can separate such points with an *n-1* dimensional hyperplane [1]. If so, we call it linear separable. So the issue is how to choose the best hyperplane among the large sets of it. One reasonable choice as the best hyperplane is the one the represents the largest margin between the two classes. So we choose the hyperplane so that the distance from it to the nearest data point on each side is maximized. If such a hyperplane exists, it is known as the maximum-margin hyperplane and the linear classifier it defines is known as a maximum margin classifier [1].

3.2. Kernel and Parameter Selection

In this project, 1000 training samples and 100 testing samples are introduced to build this SVM. The dataset included in this exercise is based on a subset of the SpamAssassin Public Corpus [2].

As we all known, the kernel function maps the original sample space into a higher-dimensional feature space, so as to conduct linear classification with ideal accuracy, although this classification maybe non-linear in the original sample space. So the next issue is to justify which kernel has the best classification result for this spam problem. In this project, we test three kernel functions (linear, Gaussian and polynomial) accordingly, and evaluate their performance with testing accuracy. Besides, for each kernels, we explore their suitable parameters, and visualize the results.

1. Gaussian (radial basis function, RBF) Kernel

$$k(\vec{x_i},\vec{x_j}) = \exp\left(-\gamma\left\|\vec{x_i}-\vec{x_j}\right\|^2\right), \gamma > 0 \qquad (2)$$

For Gaussian kernel, the critical parameter is the variance σ (for sometimes, we express it as $\gamma = \frac{1}{2\sigma^2}$). Briefly, this parameter controls the influence of the samples selected as support vectors. On one hand, if it is too large, the area determining support vectors will only influenced by margin samples and the regularization effect of penalty parameter C will be largely diminished. On the other hand, small variance will cause the model so constrained that cannot capture the data complexity. In this section, we change it exponentially to explore its impacts on the classification results.

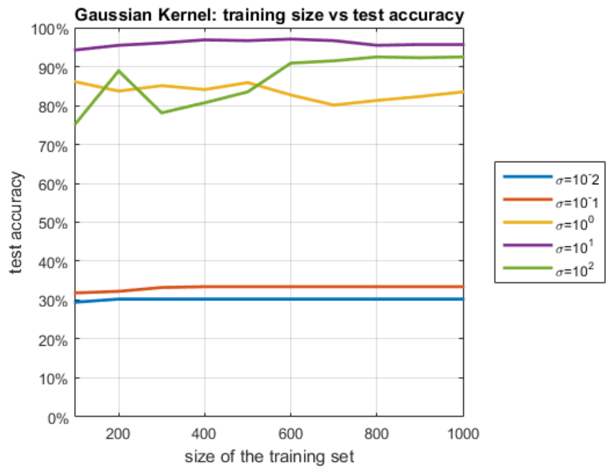


Fig. 6 Gaussian kernel: training size versus test accuracy

Fig. 6 illustrates the testing accuracy as a function of the size of the training set. Here, the penalty parameter C is chosen as 1. Later we will discuss the justification to do so. From the plot, we found that the classification results for $\sigma=10^{-2}$ and $\sigma=10^{-1}$ are rather poor. Apparently, these two curves maintain a good smoothness throughout the testing process, yet this is under the circumstance that the accuracy never grow up to 40%. That is, the classifier is so poor that the size of training set has no influence on the eventual results. As for σ=1 and $\sigma=10^2$, the accuracy jump to about 85%. The best performance in this collation is given by σ=10 whose curve is stationary and the accuracy is 95%.

2. Polynomial Kernel

$$k(\vec{x_i},\vec{x_j}) = (\vec{x_i}\cdot\vec{x_j}+1)^d \qquad (3)$$

For polynomial kernel, the parameter needed to be justified is the degree d. However, one may note that this function may suffer from the issue of numerical instability, namely the value may go to infinity $\left(\vec{x_i}\cdot\vec{x_j}+1\right) > 1$ or zero $\left(\vec{x_i}\cdot\vec{x_j}+1\right) < 1$ with large degrees. Typically, the most common value is d = 2, which is also known as quadratic kernel.

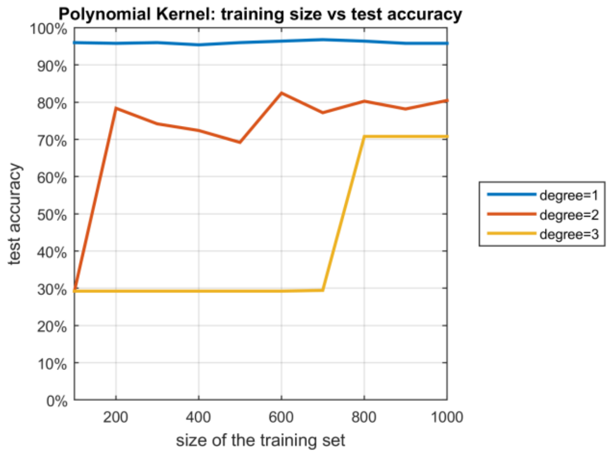


Fig. 7 Polynomial Kernel: training size versus test accuracy

In Fig. 7, surprisingly, it is shown that degree=1 has the best accuracy and smoothness, rather than the typical degree=2.

3. Linear Kernel

To explore this problem, we finally collate the linear kernel, and visualize its results with the other two functions. We note that for this spam classification problem, linear kernel has the best performance. In fact, the degree-1 polynomials just has little difference (i.e. inhomogeneous parameter) with the typical linear kernel function. This interprets why the degree-1 has best accuracy.

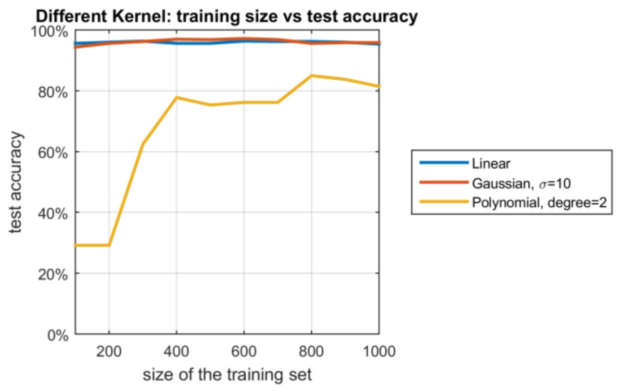


Fig. 8 Different kernel: training size versus test accuracy

In fact, for text classification problems, linear function is always recommended. Firstly, the text has a lot of features, so it is linearly separable in most of case [3]. In this case, it is needless to map data to an even higher dimensional feature space – the dimensionality of original sample space is enough, nonlinear mappings do not improve the performance. Next, it is obvious that training a SVM with linear kernel is much faster than other non-linear functions. Finally, linear kernel has less parameters to justify. For Gaussians, we need to optimize variance. For polynomials, we need to clarify degree. But for linear kernels, we just imply the dot product.

4. Penalty Parameter

After above, we conclude that linear function best suit this problem. So the next issue is to find reasonable penalty parameter C. This parameter is a positive value that controls the penalty of misclassification [4]. Informally, small C improves the smoothness of the decision boundary, while large C is designed to correctly classify all training samples by giving the model freedom to choose samples as support vectors.

One may note that the typical parameter justification process is to find C and γ simultaneously, which called "grid-search". But in this project, it is no need to do so, since we eventually imply linear function that does not have parameter γ. In this case, the only thing is to search C.

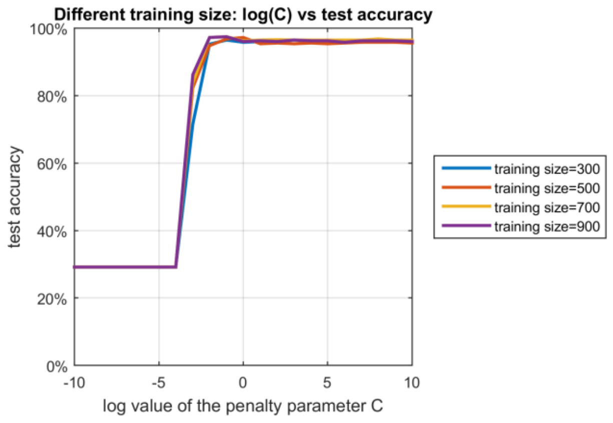


Fig. 9 Different training size: log(C) versus test accuracy

Fig. 9 shows the testing accuracy as a function of log(C). Here we change C exponentially to find best value. Result shows that, when C is large enough, the classifier become linearly separable, so the test accuracy will converge as a specific point. Luckily, as we has discussed above, for spam or text classification, the samples are just happened to be linearly separable. So the convergence will appear at an ideal position. Fig. 9 and Fig. 8, together, justify the consideration to choose linear function.

4. NAÏVE BAYES MODEL VERSUS SVM IN SPAM CLASSIFICATION

In our project, we also establish Naïve Bayes Model to make a contrast. Naïve Bayes Model is a classic, simple but useful model with the solid mathematical fundamental. The basic idea of Naïve Bayes is following:

From the Bayes Rules: $P(w|x) = P(x|w)P(w)/P(x)$, that is, posterior=(likelihood×prior)/evidence.

Given $x = (x_1, \dots x_p)$, our goal is to predict class w by posterior. In this case, we just set two class, class w_1 representing spam and class w_0 representing non-spam. Specifically, we want to find the value of w that maximizes posterior $P(w_i|x)(i = 0, 1)$. Also we can neglect the evidence $P(x)$ because it is just a parameter for normalizing. So $P(w_i|x) \propto P(x|w_i)P(w_i)$.

Now we just need to focus on likelihood and prior. Firstly, we introduce Naïve Bayes Assumption, a strong assumption, which is any features among each class is independent. So

For the likelihood:

$$P(x|w) = P(x_1|w)P(x_2|w) \dots P(x_p|w) = \prod P(x_p|w) \quad (4)$$

For the prior, $P(w_i) =$ frequency of w_i /Number of total samples.

1. Multinomial model

Before we calculating the discrete attributes of likelihood $P(x_p|w_i)$, we should introduce our model. Basically, two models upon the Na ïve Bayes model, one is multinomial model and the other is Bernoulli model. In our project, we choose multinomial model to implement. Because it considers of the word-frequency, that is, the weights of words, which will bring a higher accuracy of classification and matching the characters of natural languages.

In multinomial model

$$P(x_p|w_i)=\frac{frequency\ of\ x_p\ in\ Class\ i}{the\ number\ of\ words\ in\ Class\ i} \tag{5}$$

From now on we have obtained all factors to implement this model.

2. Laplace smoothing

If one feature x_i never occur in any samples of neither spam nor non-spam, the likelihood of this feature will be zero (P($x_i|w$)=0). It will make the whole posterior become zero. We introduce Laplace smoothing to avoid this problem.
That is:

$$P(x_p|w_i)=\frac{frequency\ of\ x_p\ in\ Class\ i{+}1}{the\ number\ of\ words\ in\ Class\ i{+}K} \tag{6}$$

which make this equation always be positive.

3. Logarithm function

Practically, $P(x_p|w_i)$ always is a very tiny decimal. For multiplication continuously, it is easy to make the answer underflow, coming up a wrong answer even becoming zero. A solution is to use logarithm function to convert multiplication operation to addition operation.

$$log\prod P(x_p|w) = \sum log P(x_p|w) \tag{7}$$

We just focus on which one is bigger instead of the specific number of the probabilities. The logarithmic operation will not alter the monotonicity of the original equations. We can still make a decision from the logarithm function.

In our project, we establish a 1000 training-sample dataset, and 100 test-sample dataset. For comparison, we trained SVM and Na ïve Bayes Model for the same training dataset and tested their performance with the same test dataset. For the best performance, we set the most suitable parameters for SVM, which we obtained from the experiments before. The Fig. 10 shows the accuracy of the two methods. The x axis is the number of training samples, from 100 to 1000 and the y axis is the accuracy of each method. From the figure 4.1, we can see the average accuracy of SVM is slightly higher than 95%, and the average accuracy of Na ïve Bayes is lower than 95%, near to 93%. For our training and test samples, SVM is better than Na ïve Bayes for spam classification.

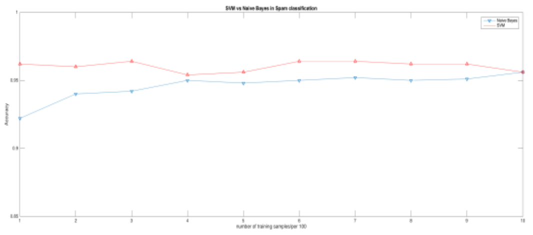


Fig. 10 The accuracy of SVM & Na ïve Bayes

Meanwhile, we also use our own emails for testing. Sample1.txt, the non-spam one, and Sample2.txt, the spam one, both of them are directly copied from our own email box (Please see the code). The result is that both SVM and Na ïve Bayes could correctly distinguish whether the emails spam or non-spam.

5. CONCLUSTION

In this project, we mainly use SVM to implement the spam classification and use Na ïve Bayes to make a comparison. The experiments data show that the performance of SVM is better than that of Na ïve Bayes in classification accuracy. Personally, we suppose that Na ïve Bayes has a weakness for text classification. The fundamental of Na ïve Bayes is that each feature is independent. In this case, it means that each word has no correlation with others. However, we must follow rules called grammar when using language, writing emails and so on. So it is impossible to verify that the correlation degree of words is zero. The performance of Na ïve Bayes largely depends on the correlation degree of samples. But for the SVM, the influence of samples to classify is slight. That is maybe the reason why SVM has the better performance in testing.

6. REFERENCES

[1] Cortes, C.; Vapnik, V. (1995). "Support-vector networks". Machine Learning. 20 (3): 273–297. doi:10.1007/BF00994018
[2] http://spamassassin.apache.org/publiccorpus/
[3] Sch ölkopf, Bernhard, and Alexander J. Smola. Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT press, 2002.
[4] Hsu, Chih-Wei; Chang, Chih-Chung & Lin, Chih-Jen (2003). A Practical Guide to Support Vector Classification (PDF) (Technical report). Department of Computer Science and Information Engineering, National Taiwan University.