A Support Vector Machine based Naive Bayes Algorithm for Spam Filtering

Weimiao Feng
Institute of Information Engineering
Chinese Academy of Sciences
Beijing, China
Email: fengweimiao@iie.ac.cn

Jianguo Sun, Liguo Zhang, Cuiling Cao
Department of Computer Science
and Technology
Harbin Engineering University
Harbin, Heilongjiang, China
Email: {sunjianguo, zhangliguo, sjg}@hrbeu.edu.cn

Qing Yang
Department of Computer Science
Montana State University
Bozeman, MT, USA
Email: qing.yang@montana.edu

***Abstract*—Naive Bayes classifiers are widely used to filter spam emails, however, the strong independence assumptions between features limit their performance in accurately identifying spams. To address this issue, we proposed a support machine vector based naive Bayes – SVM-NB – filtering system. The SVM-NB first constructs an optimal separating hyperplane that divides samples in the training set into two categories. For samples located nearby the hyperplane, if they are in different categories, one of them will be eliminated from the training set. In this way, the dependence between samples is reduced and the entire training sample space is simplified. With the trimmed training set, the naive Bayes algorithm is applied to classify emails in the test set. The SVM-NB system is evaluated with the dataset obtained from DATAMALL. Experiment results demonstrate that SVM-NB can achieve a higher spam-detection accuracy and a faster classification speed.**

***Index Terms*—Naive Bayes, support vector machine; SVM trimming technique, spam filtering**

I. Introduction

While online social networks are now recognized as popular communication channels, survey results reveal that email still reigns as the most popular form of Internet communications in our daily life. One of the long-last problem in email systems is the existence of spam emails. Spam emails, also known as junk emails, are unsolicited bulk emails received by massive recipients. The sender's addresses of spam emails are usually concealed and the spams are not requested by any of the recipients. Spam emails may contain malware, in the form of scripts or executable files, or contain disguised links that lead to phishing web sites. According to the latest survey data published by the China Anti-Spam Alliance[1], a regular Internet user receives on average 35 emails per week, however, spam emails account for 41% of them. Spam emails may always exist as long as emails are around, therefore, how to efficiently and accurately detect and filter spams is a critical and important research issue.

There are many solutions to spam filtering, e.g., the black-list and white-list filtering techniques [1], decision tree based approaches [2], [3], and machine learning based methods [4], [5]. Among various solution, machine learning based ones are receiving more and more attentions, due to its high accuracy in detecting spams. Most of existing machine learning based solutions are based on either the support vector machine (SVM) or Naive Bayes (NB) methods. Typically, an SVM based spam filtering solution has great advantages on high precision and recall rate, while an NB based solution has faster classification speed and requires a smaller training set. Overall, existing solutions are either too slow or inaccurate in solving the spam filtering problem. Another technical challenge in spam detection lies in the dependence of the features extracted from spam emails, which may cause inaccurate classifications in NB based algorithms. Therefore, removing the dependence of features is critical in improving a spam filtering system's performance.

Taking the advantages of both SVM and NB methods, we propose an innovative spam filtering algorithm, namely support vector machine - naive Bayes (SVM-NB). The SVM-NB algorithm first makes attempt to classify training samples/emails using the original NB algorithm. Then, it constructs a hyperplane in the space containing all training samples so that the training set is divided into two parts. For the samples around the hyperplane, their corresponding classification results (i.e., whether spams or not) will be further checked. For a particular sample, if its nearest neighbor (with the shortest Euclidean distance in the sample space) has a different classification result, this sample will be rejected. In other words, we adopt the trimming technique provided by SVM to eliminate bad samples from the training space. In this way, we are able to not only reduce the size of training set but also improve the independence of samples in each category. Finally, the refined training set will be used by the NB algorithm to learn the classification model and thus detect spam emails. Based on the dataset provided by DATAMALL, we carried out intensive experiments and found that the proposed SVM-NB system outperforms benchmark algorithms, including SVM- and NB-based algorithms, in terms of precision, recall rate and classification speed.

The rest of this paper is organized as follows. In section II, we introduce the related work. In section III, we briefly discuss the principles of SVM and NB algorithms and then introduce the SVM-NB algorithm. In section IV, we evaluate

[1]http://www.anti-spam.org.cn/

978-1-5090-5252-3/16/$31.00

the precision, recall rate, and execution time of the SVM-NB algorithm and further compare its performance to the benchmark algorithms. We conclude our work in section V.

II. Related Work

The problem of spam filtering has been widely studied in the last decade. Currently, most Internet service providers (ISPs), as well as email service providers, are providing junk email detection and filtering service. Existing solutions can be categorized into three groups, based on the techniques used to detect spams.

In the first group, spam emails are detected based on senders' identifications [1]. In a white list based email system, a user needs to actively mark regular emails and add the senders' addresses into the white list, i.e., future emails from these senders will be considered regular ones. For the emails sent from others that are not in the white list, they will be treated as spam or junk emails. On the other hand, a user can also put the sender's email address into the blacklist list, if a spam is identified. In practice, both black and white lists can be applied. One limitation of this type of approach is that the sender may change its identity by using dynamic IP, IP proxy, and IP spoofing techniques.

In the second category, spam filtering is realized via rule-based approaches. A typical rule-based method is the decision tree based technique. The earliest decision tree based learning system was developed by Hunt, dateing back to 1966. He created a concept learning system that uses, for the first time, a decision tree to learn concepts. It built the foundation for other decision tree based learning algorithms. For example, R. Quinlan proposed an iterative decision tree based classification algorithm ID3. To address the limitations of the ID3 algorithm, i.e., it cannot handle both continuous and discrete attributes, he later proposed an improved algorithm, the C4.5 algorithm [2]. One significant improvement in C4.5 lies in the feature selection method that is based on information gain theory. In 2002, S. Ruggieri proposed the EC4.5 algorithm [3] that uses binary search, instead of linear search, to identify the threshold in the whole training set. To generate the same decision tree, the efficiency of EC4.5 is about five times better than the original C4.5 algorithm. The memory space requirement of EC4.5, however, is much larger than that of C4.5. The principle of decision tree based methods is to classify emails based on pre-defined rules. These rules are set by regular users and thus cannot be changed frequently. One limitation of these methods is the pattern about spam emails can hardly be identified by a regular user. In addition, the configuration and maintenance of these rules could a cumbersome task.

In the last category, spam emails are detected by machine learning based algorithms. One popular solution is based upon the support vector machine (SVM) technique. A SVM is a supervised learning technique for classification that is formally defined as a separating hyperplane in the space composed of training samples. The hyperplane essentially divides the training samples into different categories. Because it is able to handle small training set, non-linear and high-dimension classification problem, it is widely applied in text classification and spam email classification. The classification speed of SVM, however, depends highly on the number of support vectors extracted from training samples. In other words, larger the number of support vectors, slower the classification speed. When the number of samples are huge, implying large number of support vectors, SVM's classification speed becomes very slow. To address this issue, Scholkopf *et. al.* proposed a method to construct new vectors, and thus reduce the computational complexity of support vector decision functions [6]. Although the simplification greatly accelerates an SVM's classification speed, the SVM's precision and recall rate are significantly reduced, due to the difference between the new vectors and the original vectors. Another type of methods is based on the Naive Bayes algorithm. NB is a simple and effective classification algorithm, and its performance is as good as other algorithms, e.g., decision trees and neural networks. In certain applications, NB can outperform other methods. One important assumption in NB is the independence of training samples, which is usually not true in real-world applications. Therefore, the applications of NB algorithm are limited to a small set of problems. To address this issue, people proposed models to relax this unrealistic assumption and thus improve NB's classification performance. For example, Friedman et. al. proposed the TAN (tree augmented naive Bayes) classifier that relaxes the independence assumption in the Naive Bayesian algorithm [5]. In the model, a node in the tree can rely on no more than one non-class node. TAN achieve a great tradeoff between the classification speed and classification accuracy. Later on, BAN (Bayesian network tree augmented naive Bayes) model is proposed in [7]. BAN further expands the structure of TAN and supports arbitrary directed graph between features. Experimental results show, however, it is difficult to learn the structure of BAN. The learning process is equivalent to solving an NP-Complete problem. In summary, SVM based algorithms usually offer high precision and recall rate but NB based solutions have faster classification speed and require fewer number of training samples. In summary, none of existing solutions can quickly and accurately classify spam emails.

III. Spam Filtering with SVM-NB

Both NB and SVM techniques are currently mainstream text classifiers, although they are different in various aspects, e.g., classification principle, efficiency, and accuracy. In this section, we first discuss the fundamentals of these two methods and then propose a new solution to spam filtering, a unification taking the advantages of both NB and SVM.

A. Overview of Spam Filtering

The contents in regular emails are usually unstructured and text-based data. In order to be analyzed by computers, the data need to be first processed so that they can be represented in a structural manner. Typically, the texts in an email are represented as a vector, based on the space vector model. In the vector, an element corresponds to a term that could be

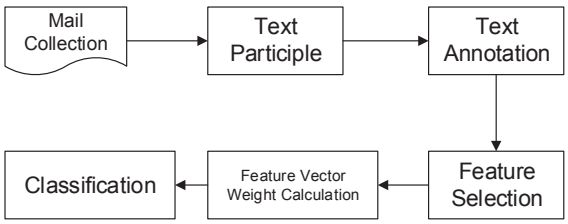


Fig. 1. Overview of the proposed spam filtering technique.

either a word or a phrase. If a term shows in the contents, the corresponding element in the vector in a non-zero value. Before classifying an email, its contents/texts need to be preprocessed by the following modules, including text participle, text annotation, feature selection, and weight calculation with feature words.

As shown in Fig 1, the contents from an email will be first processed by the *text participle* module. This module breaks a regular Chinese sentence into several meaningful units such as words or phrases, according to certain rules. There are many tools for text segmentation in the natural language processing field. The units generated from the text participle module will then be labeled by the *text annotation* module.

Text annotation module tags each word/segment based on its nature, e.g., a noun, verb, adjective, or something else. In addition, this module will remove less important words, e.g., prepositions and auxiliary verbs. After all words are annotated, a vector can be constructed to contain all segments produced from the text annotation module. The *feature selection* module selects several dimensions from the vector to form a feature vector. The feature vector largely reflects the meaning of contents in the email.

Different weights can be assigned to each term in the feature vector. The purpose of the *feature vector weight calculation module* is to identify an appropriate weight assignment scheme. For example, the term frequency-inverse document frequency (TF-IDF) method can be used to weight a certain term. The IF-IDF method basically puts a weight to each term, based on their occurrence frequency. After the preprocessing, the feature vector will be used to represent the email and fed into the *classification* module. The classification module determines whether the input feature vector indicates a spam or non-spam email. We design an innovative algorithm, called SVM-NB, to realize efficient and accurate classification that will be introduced later.

B. Naive Bayes Classifier

Before introducing the proposed SVM-NB algorithm, we first briefly discuss the Naive Bayes and SVM methods. Let $\vec{x} = (x_1, x_2, \cdots, x_n)$ denote a feature vector and $C = (c_1, c_2, \cdots, c_m)$ denote all possible categories that the vector may belong to. The principle of a Naive Bayes classifier is to compute the probabilities $p_1, p_2, \cdots, p_m$ for $\vec{x}$ where p_j is the probability that $\vec{x}$ belongs to category c_j. Determining the value of $\max(p_1, p_2, \cdots, p_j)$, we will know which category the feature vector $\vec{x}$ belongs to. Therefore, the classification problem can be considered as finding the maximum value of the following equation

$$P(c_j|x_1, x_2, \cdots, x_n) = \frac{P(x_1, x_2, \cdots, x_n|c_j)P(c_j)}{P(c_1, c_2, \cdots, c_n)}, \quad (1)$$

where $P(c_j)$ denotes the probability that a random sample belongs to category c_j. $P(x_1, x_2, \cdots, x_n|c_j)$ is the probability that category c_j contains the feature vector $\vec{x} = (x_1, x_2, \cdots, x_n)$, if we already know the training sample is in c_j. $P(c_1, c_2, \cdots, c_n)$ is the joint probability of all possible categories.

For all given categories, the denominator $P(c_1, c_2, \cdots, c_n)$ is a constant, so equation 1 can be simplified as

$$c_{NB} = \arg\max_{c_j \in C} P(x_1, x_2, \cdots, x_n|c_j)P(c_j) \quad (2)$$

According to the assumption of Naive Bayes theorem, the terms in the feature vector are identically distributed. Therefore, their joint probability distribution is equal to the production of every term's probability. In other words, we have

$$P(x_1, x_2, \cdots, x_n|c_j) = \prod_{i=1}^{n} P(x_i|c_j).$$

It means that we are only interested in finding

$$c_{NB} = \arg\max_{c_j \in C} P(c_j) \prod_{i=1}^{n} P(x_i|c_j). \quad (3)$$

The assumption of applying Naive Bayes method is the independence between the feature vectors. In reality, however, there are always many dependent vectors presented in the training set. It can be seen from equation 3 that all probabilities $P(x_i|c_j)$ must be independent to each other. If not, NB will yield incorrect classification results. It is critical to have a mechanism to eliminate the dependency among feature vectors as much as possible. Fortunately, SVM is such a tool that can efficiently classify non-linear or dependent training samples into different categories. Therefore, we combined the NB and SVM methods to proposed an innovative classification approach, called SVM-NB.

C. Support Vector Machine (SVM)

Due to its ability of solving nonlinear classification problems by applying the kernel trick, SVM recently becomes a popular. The principle of SVM is to construct a hyperplane to ensure that the distance between the two types of structure is maximized. While maximizing the distance, SVM still needs to reduce the chance that points are classified into wrong groups to minimize punishments. In this way, we can see SVM is essentially a quadratic optimization method.

Given a separable training set, composed of vectors, SVM can construct an optimal hyperplane

$$(w \cdot x) + b = 0,$$

where x is a vector.

If we use (x_i, y_i) to denote a sample from the training set, we know $x_i \in R^d$ and $y_i \in \{-1, +1\}$ where d is the dimension of the vector x_i, and y_i denotes the classification result of feature vector x_i. If the hyperplane can divide the sample set, we have the following condition satisfied

$$y_i \left[(w \cdot x_i) + b\right] - 1 \geq 0, i = 1, 2, \cdots, n.$$

To maximize the distance between these two sets of samples, we need to minimize

$$\phi(w) = \frac{1}{2}||w||^2 = \frac{1}{2}(w \cdot w).$$

By solving the optimization problem, we have the following result

$$\sum_{i=1}^{n} = y_i a_i^0 (x \cdot x_i) + b = 0,$$

where a_i^0 is a Lagrange multiplier.

If the training set is not dividable, a relaxation factor $\epsilon_i \geq 0$ and a penalty parameter C will be introduced. The optimal objective function is then transformed into

$$\phi(w) = \frac{1}{2}(w \cdot w) + c \sum_{i=1}^{n} \epsilon_i,$$

with the following constraints:

$$y_i \left[(w \cdot x_i) + b\right] - 1 + \epsilon_i \geq 0, i = 1, 2, \cdots, n.$$

SVM takes the signs of $\{y_i\}$ to distinguish samples from different categories.

$$I(x) = sgn\left(\sum_{i=1}^{n} y_i a_i^0 (x \cdot x_i) + b\right).$$

Using kernel functions, SVM is able to build a connection between a low-dimension vector set and a high-dimension vector set. For problems that are not dividable in low-dimensional space, SVM transforms it into a high-dimension space, and thus makes it dividable, if possible. A kernel function is defined as $K(x, y) = \phi(x) \cdot \phi(y)$ where ϕ is the mapping function between the low-dimension vector x and high-dimension vector y. If the kernel function is appropriately selected, SVM obtains a classification function as

$$I(x) = sgn\left(\sum_{i=1}^{n} y_i a_i^0 (\phi(x) \cdot \phi(x_i)) + b\right),$$

where $\phi(x)$ is a higher dimensional vector. Because the kernel function $K(x, y)$ only relates to x and y where x and y are both low dimensional vectors, there is no increase in the computational complexity at all.

D. SVM-NB Algorithm

From the previous discussions, we know the major limitation of Naive Bayes classifier is the assumption of independence of the feature vectors extracted from training samples. On the other hand, SVM is capable to find a perfect hyperplane to divide training samples into two categories. Because the goal of SVM is to maximize the distance between the boundaries of two categories, the mixing of these two boundaries will not appear.

Because the independence assumption does not hold in real applications, the recall rate and accuracy of NB based classifications are significantly affected. To address this issue, we apply an SVM based trimming technique to eliminate samples that are classified into wrong categories by NB. In this way, the dependence of feature vectors are reduced and the independence among training samples are enhanced. Considering the fast classification speed of NB, we propose the SVM-NB algorithm that can efficiently and accurately classify emails, and thus achieve effective spam filtering.

In the SVM-NB algorithm, the training samples are first processed by the original NB algorithm. For every feature vector extracted from the training set, there will be a corresponding category generated by the NB algorithm. Therefore, we will have a set of results

$$(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m),$$

where $x_i \in R^n$, $y_i \in \{-1, +1\}$, $i = 1, 2, \cdots, m$, and n denotes the dimension of the feature vectors.

Next, we will find the nearest neigbor(s) for every feature vector. For a feature vector, if its nearest neighbor and itself belong to the same category, then the vector will be kept. Otherwise, the vector (and the corresponding category information) will be dropped from the training set. The vector can be eliminated because it is classified into a wrong category, due to the dependence between itself and the nearest-neighbor vector. In other words, we remove the dependent samples from the training set. This will further help to improve the classification accuracy.

Given two feature vectors $\vec{u} = (u_1, u_2, \cdots, u_n)$ and $\vec{v} = (v_1, v_2, \cdots, v_n)$, their distance is defined as

$$D(\vec{u}, \vec{v}) = \sqrt{\sum_{k=1}^{n} (u_k - v_k)^2}.$$

Based on the above equation, the nearest neighbor of a feature vector can be easily identified. According to the principle of SVM, two closed (in distance) feature vectors are very likely in the same category. Otherwise, the classification results must be refined.

The pseudo-code of the SVM-NB algorithm is shown in Algorithm 1. In the algorithm, we use $\vec{X} = \{x_1, x_2, \cdots, x_m\}$ and $\vec{Y} = \{y_1, y_2, \cdots, y_m\}$ to denote the feature vectors and the corresponding categories. Let $\vec{V} = \{v_1, v_2, \cdots, v_m\}$ denote the classification results generated by the original NB algorithm. The classification results obtained from the original NB algorithm will be refined by the SVM-NB algorithm, i.e., dependent feature vectors are removed from the training set. The resulting vector $\vec{V}$ will be used by the original NB algorithm to perform the classification task.

E. Implementation of SVM-NB

The detailed implementation of SVM-NB algorithm is illustrated in Fig. 2. The system's input is a set of training

TABLE I
STATISTICS OF THE DATASET OBTAINED FROM DATAMALL.

Type of Emails	Number of Samples	Number of Features	Number of Classes
Junk advertising	2245	27	3
Porn violence	1520	15	2
Special garbled characters	836	10	2
Special remarks	399	16	3
Normal emails	2546	31	4

Algorithm 1 SVM-NB($\vec{X}$, $\vec{Y}$)

Require: $\vec{X} = \{x_1, x_2, \cdots, x_m\}$, $\vec{Y} = \{y_1, y_2, \cdots, y_m\}$ and $\vec{V} = \{v_1, v_2, \cdots, v_m\}$.
Ensure: $\vec{V}'$

for all $i = 1$ to m **do**
  for all $j = 1$ to m **do**
    if $i \neq j$ **then**
      $D_{ij} \leftarrow D(x_i, x_j)$
    end if
  end for
end for
for all $i = 1$ to m **do**
  $Min \leftarrow \infty$
  $NN \leftarrow i$
  for all $j = 1$ to m **do**
    if $Min > D(x_i, x_j)$ **then**
      $Min \leftarrow D(x_i, x_j)$
      $NN \leftarrow j$
    end if
  end for
  if $y_i \neq y_j$ **then**
    Remove v_i from $\vec{V}$
  end if
end for
return $\vec{V}$

samples/emails that include both spam and non-spam emails. The contents of every email in the training set will be first segmented and annotated, based on ICTCLAS, a Chinese lexical analysis system developed by the Institute of Computing Technology, Chinese Academy of Sciences. Then, the feature vector of every email will be extracted by the feature selection module. Because most of the features present redundancy and inconsistency, we adopt a feature selection method that is based on the information gain (IG). Specifically, we compute the IG for every feature vector, no matter whether it corresponds to a spam or a regular email. These feature vectors are then ordered based on their IG values, in a decreasing order. Only the top K features will be selected to represent an email.

After the text segmentation, annotation and feature selection, the obtained feature vectors are fed into the NB algorithm to obtain the original training results. These results are then refined by the SVM-NB algorithm, i.e., dependent features are trimmed from the training results. Finally, the NB algorithm uses the trimmed training set to build a classification model, and this model will be used to detect future spams.

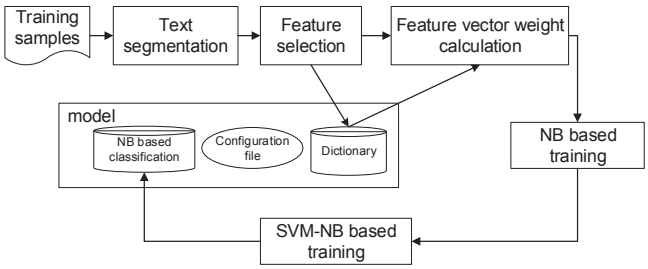


Fig. 2. Detailed implementation of the SVM-NB spam filtering system.

IV. EXPERIMENT AND RESULT ANALYSIS

To evaluate the performance of the proposed SVM-NB algorithm, we implement the spam filtering system on a regular PC with an Intel CORE 7i 2.60 GHz CPU and 8.0 G RAM. The main algorithm, SVM-NB, is implemented by MATLAB. Other modules such as text segmentation and annotation are implemented by Java. The dataset is obtained from the DATAMALL at http://www.datamall.com/. As far as we know, DATAMALL provides the largest Chinese spam emails dataset, so we use it to validate and evaluate SVM-NB. The statistics of the dataset we obtained from DATAMALL are listed in TABLE I. From the dataset, we randomly select 4000 spam emails and 4000 non-spam emails to form the training set. The rest of the dataset are used as the test set.

Spam emails are further divided into four groups: junk advertising, porn violence, emails with special garbled characters and special remarks. Junk advertising emails usually contain a lot of abnormal information on marketing, training and promotions. Emails with special garbled characters tend to be spams as most receipts will not read or understand them. Special remarks may contain sensitive words that may cause threat to the national security. The normal emails are usually used for basic communications between people, of course, these emails may also include normal advertising and marketing information.

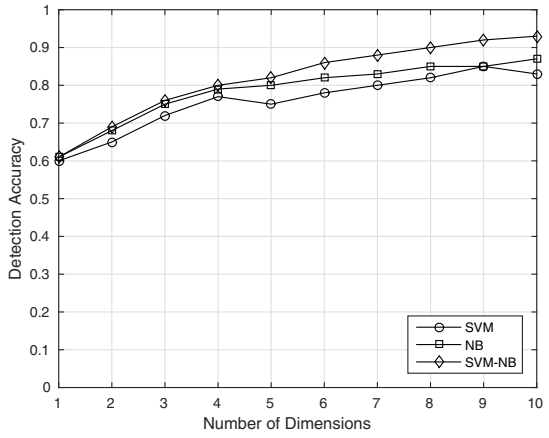
Based on the dataset from DATAMALL, we evaluate the performance of NB, SVM and SVM-NB algorithms, in terms of precision and recall rate, and execution time. The precision of a classification algorithm is defined as

$$\frac{\textit{Number of correctly classified emails}}{\textit{Total number of emails}}$$

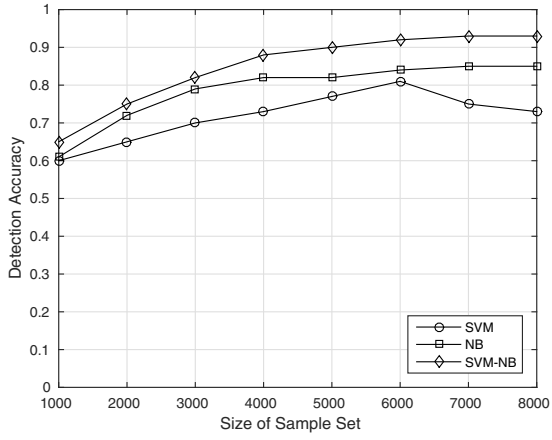
On the other hand, the recall rate is given as

$$\frac{\textit{Number of detected spam emails}}{\textit{Number of all spam emails}}$$

As shown in Fig. 3(a), when the dimension of feature vectors increases, the precision of all algorithm increases. In Fig. 3(b), the precision of every algorithm increases as the size of training set increases. This is expected because a large number of training samples always means a large number of support vectors, which significantly improves classification precision. Overall, we can see the number of support vectors generated in SVM-NB is the smallest one, resulting a faster classification speed.



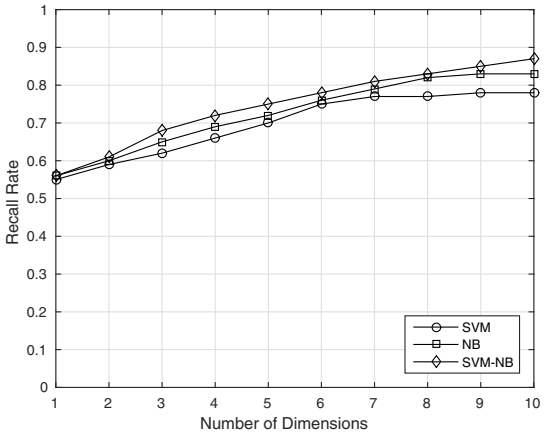
(a)



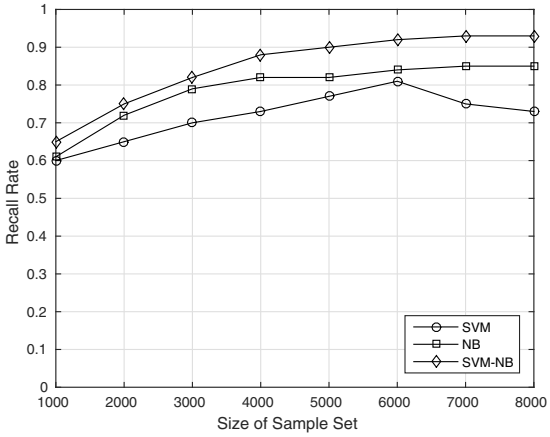
(b)

Fig. 3. Spam detection accuracies of SVM, NB and SVM-NB algorithms with (a) different numbers of dimensions and (b) different sizes of sample set.

We plot the recall rates of various algorithms in Fig. 4. Similar to the precision results, when the number of training samples increases, the recall rate increases as well. From Fig. 3 and Fig. 4, we can see that both precision and recall rate of SVM are lower than the other two methods. In fact, when the number of training samples or dimensions increases to a certain value, both precision and recall rate drop. This is because SVM is ineffective in handling large training sets. Compared to SVM, NB offers better precision and recall rate. The proposed SVM-NB, however, provides the best precision and recall rate because it takes the advantage of both SVM and NB.



(a)

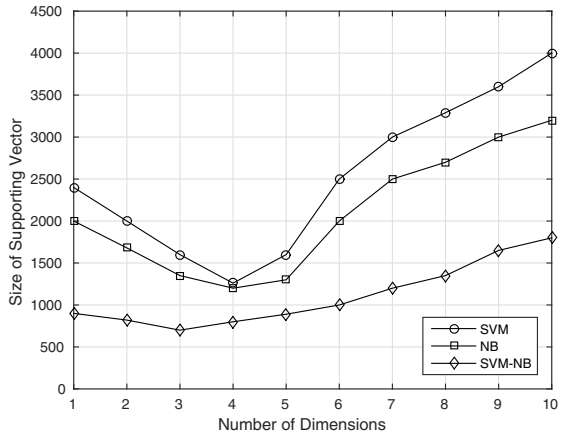


(b)

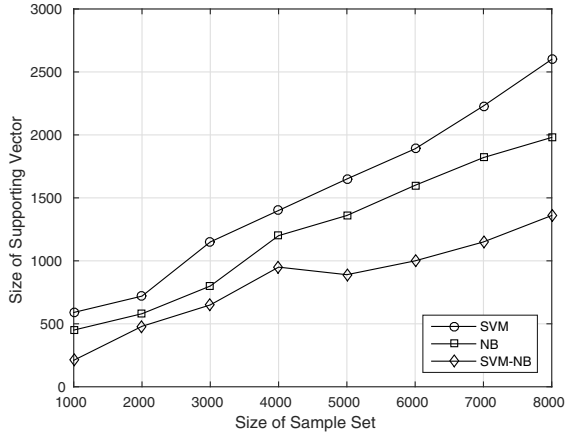
Fig. 4. Recall rates of SVM, NB and SVM-NB algorithms with (a) different numbers of dimensions and (b) different sizes of sample set.

Although large number of training samples can help to improve precision, it also generates more redundant or meaningless vectors. This will significantly decrease the classification speed. As shown in Fig. 5(a), when the number of dimensions increases, the number of support vectors decreases at the beginning and then increases drastically. On the other hand, the number of support vectors increases almost linearly when the number of samples increases, as shown in Fig. 5(b).

The execution times of various algorithms are shown in Fig. 6 where only the classification times are plotted. As the training times are one-time overhead, we omit them in this paper. We can see that the execution times slightly change when the size of training sets is relatively small. With larger training sets, the classification speed of both NB and SVM

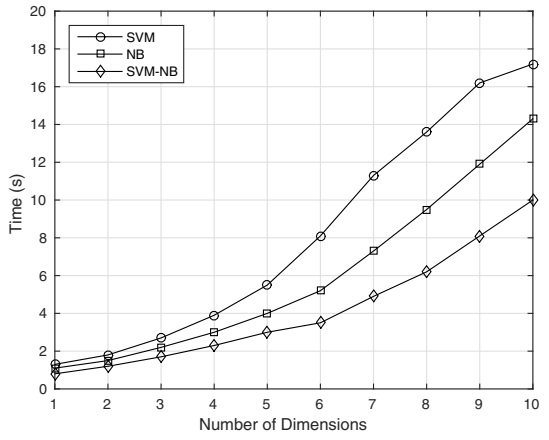


(a)

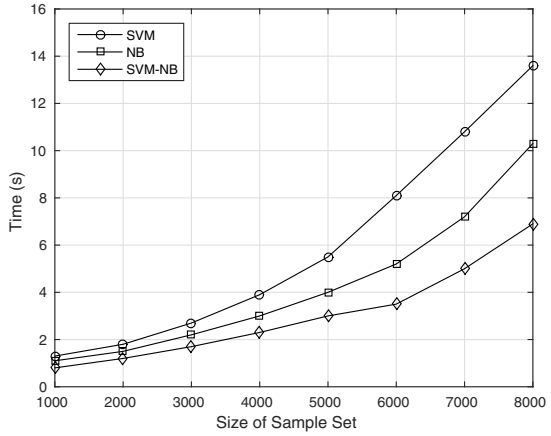


(b)

Fig. 5. Sizes of the supporting vectors in SVM, NB and SVM-NB algorithms with (a) different numbers of dimensions and (b) different sizes of sample set.



(a)



(b)

Fig. 6. Execution times of SVM, NB and SVM-NB algorithms with (a) different numbers of dimensions and (b) different sizes of sample set.

drops drastically, i.e., the execution time used for classification rises quickly. Although SVM-NB's execution time also increases (it is almost certain that classification speed decreases with larger feature dimensions), its classification speed does not decreases as quickly as the others. Because SVM-NB trims feature vectors, i.e., it eliminates redundant and useless vectors, it reduces the number of training vectors, so the classification speed is improved. In summary, we conclude that the proposed SVM-NB algorithm yields better precision, recall rate, and classification speed.

V. Conclusion

We proposed an SVM-NB system to achieve effective and efficient spam email filtering. SVM-NB aims at removing the assumption of independence among features extracted from training set, when the NB algorithm is applied. The solution leverages SVM technique to divide training samples into different categories and identifies dependent training samples. Removing those samples results in a more independent training set with few overlapping features. Based on intensive experiments, we find the SVM-NB algorithm offers better precision and recall rate in detecting spams. In addition, we confirm that SVM-NB is a more efficient classification system, compared to the pure SVM based solution. The proposed system is designed for emails with strong dependence between each other, which is typically true in current email system.

With the rapid development of email systems, spams are not only limited to text based emails. In fact, there are junk emails containing various formats of data, e.g., pictures and other types of multimedia files. Currently, the SVM-NB algorithm is only applicable to text=based spams detection, we will extend SVM-NB and make it suitable to filtering spams with different formats of data.

References

[1] J. W. Yoon, H. Kim, and J. H. Huh, "Hybrid spam filtering for mobile communication," *computers & security*, vol. 29, no. 4, pp. 446–459, 2010.

[2] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[3] S. Ruggieri, "Efficient c4. 5 [classification algorithm]," *IEEE transactions on knowledge and data engineering*, vol. 14, no. 2, pp. 438–444, 2002.

[4] B. Sch Ikopf, S. Mika, C. Burges *et al.*, "Input space versus feature space in kernel-based method," *IEEE Trans Neural Networks*, pp. 1000–1017.
[5] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine learning*, vol. 29, no. 2-3, pp. 131–163, 1997.
[6] B. Scholkopf, S. Mika, C. J. Burges, P. Knirsch, K.-R. Muller, G. Ratsch, and A. J. Smola, "Input space versus feature space in kernel-based methods," *IEEE transactions on neural networks*, vol. 10, no. 5, pp. 1000–1017, 1999.
[7] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, S. Y. Philip *et al.*, "Top 10 algorithms in data mining," *Knowledge and information systems*, vol. 14, no. 1, pp. 1–37, 2008.