Columbia International Publishing
International Journal of Computational Intelligence and Pattern Recognition
(2014) Vol. 1 No. 1 pp. 1-26

Research Article

Uninterrupted Approaches for Spam Detection Based on SVM and AIS

Ying Tan[1*], and Guangchen Ruan[1]

Received 28 November 2013; Published online 8 March 2014

Abstract

The proliferation of spam has increasingly caused serious problems to our daily electronic communications. Jupiter Research estimates an e-mail user will on average receive more than 3,900 spam mails in 2007. This number is on average of 40 spam mails in 1999. This paper proposes uninterrupted detection approaches based on Incremental Support Vector Machine and Artificial Immune System for the spam of e-mail stream. These approaches use a window to hold several classifiers, each one classifies the e-mail independently, and the e-mail is labeled according to a majority voting strategy. The exceeding margin update technique of support vector machine (SVM) is also used for the dynamic update of each classifier in the window. A sliding window is used to purge out-of-date knowledge. When a new batch of e-mail arrives, the classifier at rightmost in the window is removed from the window while the remaining classifiers just slide a position to right and the classifier at leftmost is newly generated by the previous batch. These two techniques endow our algorithms with dynamic and adaptive properties as well as the ability to trace the changing content of e-mails and user's interests in an uninterrupted way. Eight methods, possibly regarded as different implementations of the uninterrupted detection of e-mail stream, including Hamming Distance (with and without mutation), Included Angle, SVM and Weighted Voting, are developed and elaborated in this paper. Experiments on two public benchmark corpora PU1 and Ling are conducted to verify the validity of the proposed methods. The eight methods are compared with current methods for accuracy, precision, recall, miss rate and speed of detection. The results demonstrate that the proposed uninterrupted detection approaches are promising way to contain spam.

Keywords: Spam detection; Support Vector Machine (SVM); Artificial Immune System (AIS); Exceeding Margin technique (EM); Sliding window; Majority voting

1. Introduction

E-mail has become a common and convenient medium for daily communication. However, the proliferation of spam, usually defined as unsolicited commercial e-mail (UCE) or unsolicited bulk e-mail (UBE), has caused serious problems in daily communication. Many of us may think of spam as a new problem, but in fact, it goes back at least to 1975, as noted by the late (Postel, 1975). Spam occupies valuable communications bandwidth and storage space, and wastes a lot of a user's time. Spam also causes serious threats to the internet security due to

*Corresponding email: ytan@pku.edu.cn
1 Key Laboratory of Machine Perception (MOE), Peking University, Department of Machine Intelligence, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China

viruses and malicious codes. Jupiter Research estimates the average e-mail user will receive more than 3,900 spam mails in 2007. This number is on average of 40 spam mails in 1999. Ferris Research estimates that spam costs U.S. companies 10 billion in 2003, and a user spends on average 4 seconds to process a spam mail (Wu et al., 2005).

There have been many attempts to detect and filter spam out of the normal e-mail stream. This paper summarizes some of those solutions and classifies them in three categories, simple approaches, intelligent approaches and hybrid approaches.

Simple approaches, including munging, listing, aliasing and challenging, are techniques that are not as effective, but are easily implemented. Intelligent approaches are techniques with self learning ability and good performance, which include Naïve Bayes (Androutsopoulos et al., 2000; Sahami et al. 1998; Li et al., 2006; Shrestha and Lin, 2005; Anayat et al., 2004), Support Vector Machine (SVM) (Drucker et al., 1999; Tan and Wang, 2004), Artificial Neural Network (ANN) (Clark et al., 2003; Stuart et al., 2004; Tan et al., 2000), Artificial Immune System (AIS) (Oda and White, 2003a; Oda and White, 2003b; Oda and White, 2005; Secker et al., 2003; Bezerra et al., 2006; Wang et al., 2006; Tan, 2006) and DNA Computing (Rigoutsos and Huynh, 2004). As a variety of novel techniques are continually developed, some researchers realize that one technique alone can not solve the problem of spam detection fully and completely, since different techniques excel in different ways. Hybrid approaches combine two or more techniques in an attempt to improve overall performance and to overcome the shortcomings of a single approach (Leiba and Borenstein, 2004; Wu et al., 2005).

Support Vector Machine (SVM) has already proved its superiority in pattern recognition for its generalization performance. Natural immune system has some desirable properties for spam detection, including pattern recognition, dynamically changing coverage and noise tolerance, etc. Those properties are considerably desirable for our task of anti-spam.

SVM is a classification algorithm based on the Structural Risk Minimization principle from statistical learning theory formulated by Vapnik (Vapnik, 1982; Vapnik, 1995; Drucker et al., 1997). The goal of SVM is to find an optimal hyperplane for which the lowest true error can be guaranteed.

Artificial Immune Systems (AIS), inspired by the mammalian immune system have become an increasingly popular computational intelligence paradigm in recent years. AIS seeks to use the immune components and processes of mammalian immune system as metaphors to produce artificial systems that encapsulate desirable properties of the natural immune system. These AIS are then applied to solve complex problems in a wide variety of domains (De Castro and Timmis, 2002).

This paper uses SVM and AIS as the basis for uninterrupted detection approaches for stopping spam. The support vectors of the trained SVM are used to generate naïve detectors for the proposed AIS-based approaches. The naïve detectors frequently used for classification are promoted to be memory cells. For the purpose of tracing the dynamic changing property of the content of e-mails and user's interests, a window is used to hold several classifiers, each one classifies the incoming e-mail independently and then the window labels the e-mail through majority voting. The exceeding margin update technique is used to dynamic update the classifier, the detector set, and the memory set. The messages that exceed the margin are used to update the classifier. A sliding window is used to purge the out-of-date knowledge. When new batch arrives, the ``oldest'' classifier in the window will be removed from the window and the remaining classifiers slide a position to right. The ``youngest'' classifier is generated from the previous batch of data. Thus, the proposed approaches are able to trace the changing content of e-mails and user's interests in a continuous way.

Several approaches, including Hamming Distance (with and without mutation), Included Angle, SVM and Weighted Voting, which can be regarded as different implementations to the uninterrupted detection of e-mail stream, are developed in this paper. Furthermore, experiments on two public benchmark corpora PU1 and Ling are conducted to verify the performance of the approaches. Finally, a performance comparison between the different approaches is made for accuracy, precision, recall, miss rate and speed.

The rest of this paper is organized as follows. In Section 2, the related work about anti-spam is presented concisely. Section 3 gives an introduction to SVM and AIS. Section 4 details about the exceeding margin and sliding window. The algorithmic implementations of the approaches are elaborated in Section 5. Experimental results on two public benchmark corpora are reported in Section 6. Section 7 is the discussion of the experimental results. Finally, conclusions about the experiments and future work are given in Section 8.

2. Related Work

The task of spam detection has become increasingly serious and a variety of anti-spam solutions have been proposed in different literature. In order to present these techniques clearly, they are divided into three groups: (A) Simple Approaches, (B) Intelligent Approaches, (C) Hybrid Approaches.

2.1 *Simple Approaches*

Simple approaches do not have the capability to self learning and do not automatically update themselves. Some examples include munging, listing, aliasing, challenging. These approaches are easy to implement, but are not very good for spam detection. They are usually incorporated into more complicated systems and are used as a preliminary filtering layer in a coarse granularity.

1. **Munging** is where an e-mail address is deliberately alternated so that it is no longer usable for e-mail harvesters (Hoanca, 2006). For example, ruangc@gmail.com can be munged as ruangc at gmail dot com.

2. **Listing** is usually used as a tool that permits legitimate senders in a whitelist, but blocks spammers in a blacklist, and as greylist for senders that could be spammers.

3. **Aliasing** is a strategy that offers self-destructing disposable email addresses (DEA) by encapsulating policy in e-mail addresses (Ioannidis, 2003). This approach is used by vendors like Spamgourmet.com, which provides e-mail addresses that look like someword.x.user@spamgourmet.com. ``Someword'' is a word you have never used before, it is used as a way to uniquely identify the sender or mailing list provider. X is the maximum number of email messages you want to receive at this address, and user is your username at spamgourmet.com.

4. **Challenging** adopts a challenge-response mechanism which requires the e-mail sender to provide ``proof-of-words'' (POWs) instead of monetary stamps. Senders are required to spend some CPU processing time resolving POWs puzzles. Some researchers use Completely Automated Public Turing test to tell computers and humans apart (CAPTCHA) to block program bots (von Ahn et al., 2004).

2.2 *Intelligent Approaches*

Compared to simple approaches with preset rules, intelligent approaches have the ability to self learn or have no user intervention. Some of them, like Naïve Bayes, and SVM, are based on theoretical foundations of statistics and structural risk minimization principle. Others, like Artificial Neural Network, Artificial Immune System, and DNA Computing, are inspired by biological processes or phenomena. Intelligent approaches play an increasingly important role in stopping spam due to their ability of self learn and high level of performance. Some of them are briefly presented as follows.

1. **Naïve Bayes** is one of the most commonly used machine learning approach (Androutsopoulos et al., 2000). Anayat et al. employ a probability weight based Bayesian approach for spam filtering. A numerical weight is assigned to each word appearing in an e-mail by using a Bayesian rule (Anayat et al., 2004).

There are a lot of variants of traditional Naïve Bayes. Shrestha et al. utilize the inner-related property of the same word occurring in different parts of the e-mail to obtain an improvement on performance (Shrestha and Lin, 2005). Li et al. propose an improved Naïve Bayes approach based on users' feedback, and achieve relatively lower false positive and promising performance (Li et al., 2006).

2. **Support Vector Machine** is discussed extensively in (Vapnik, 1982; Vapnik, 1995; Drucker et al., 1997). Drucker et al. use support vector machine to classify e-mails into spam or non-spam. Furthermore, a comparison is made with other three classification algorithms, i.e., Ripper, Rocchio, and boosting decision trees. Between the four algorithms tested, SVM performed the best in terms of binary representation features (Drucker et al., 1999).

3. **Artificial Neural Networks** simulate the procedure for information processing in the human brain and become a very important branch of machine learning. Clark and his team use artificial neural networks to classify e-mails into spam and non-spam automatically. Their designed system Linger performs very well on corpus Ling (Clark et al., 2003). Stuart et al. use artificial neural networks for e-mail classification by adopting the descriptive characteristics of e-mail to construct the feature set (Sruart et al., 2004).

4. **Artificial Immune System** based spam solution is inspired by the human immune system to be detailed in section 3.2. Secker et al. propose an AIS based anti-spam algorithm, AISEC, which continuously classifies electronic mail as interesting e-mail and uninteresting e-mail. Only spam e-mail is used for the generation of detectors. He also gives a comparison with Naïve Bayes classifier in (Secker et al., 2003). Later, Oda et al. propose another AIS based approach for spam detection, in which spam mails and legitimate mails are used to generate negative detectors and positive detectors respectively (Oda and White, 2003a; Oda and White, 2003b; Oda and White, 2005; Tan, 2006). Bezerra et al. propose an antibody network called SRABNET (Supervised Real-Valued Antibody Network) as an immunological filter for spam (Bezerra, 2006). Wang et.al propose an immune based peer-to-peer model for anti-spam. In their approach, different peers can collaboratively share knowledge about spam in P2P environment (Wang et al., 2006).

5. **DNA Computing** is a novel computational intelligence paradigm inspired by computational biology like gene finding and protein annotation. Isidore et al. develop a DNA computing based approach by using pattern-discovery as an underlying tool for automatic identification of unsolicited e-mail messages. Their designed system Chung-Kwei can be trained quickly and can be re-trained without interrupting the classification of incoming email. The performance of Chung-Kwei is very promising in real application (Rigoutsos and Huynh, 2004).

2.3 *Hybrid Approaches*

In recent years, many researchers realize that a single technique is not powerful enough to block all kinds of spam. Each technique has its own advantages and disadvantages. A hybrid approach using a multi-faceted architecture can combine more than one technique to improve the overall performance while overcoming the drawbacks of a single technique. Wu et al. analyze the shortcomings of current anti-spam solutions and propose a multi-faceted approach towards spam-resistible mail by coordinating layers of shields. First, they list good senders. Second, they label the message. Third, they collect known spam digests. Finally, they propose challenging a probable spammer in a spam-resistible mail agent (SRMA). Their experiments show that the proposed SRMA is immune to existing spambots and the prototype of SRMA is proved to be effective, feasible and deployable (Wu et al., 2005). Leiba et al. propose another multi-faceted framework. The SpamGuru combines the classifiers of JClassifier, SwiftFile, Plagiarism, Linear and Chung-Kwei together. It is constructed in layers of a particular sequence to provide a filtering granularity. This granularity established the closer the layer is to the user, the more delicate granularity it adopts. The experimental results show that SpamGuru can achieve excellent discrimination between spam and legitimate mail, and offer a tunable tradeoff between detection rates of spam and false positives of legitimate mails (Leiba and Borenstein, 2004).

3. Backgrounds of SVM and AIS

In this section, some basic backgrounds of SVM and AIS are briefly introduced. The focus is on the parts pertinent to our algorithms. For a complete review of them, interesting readers refer to literature (Vapnik, 1982; Vapnik, 1995; Drucker et al., 1997; Hofmeyr and Forrest, 2000).

3.1 *Support Vector Machine*

Support Vector Machine is a classification algorithm based on the Structural Risk Minimization principle in statistical learning theory. Assume there are two classes, $y_i \in \{-1, 1\}$, and N labeled training examples: $\{x_1, y_1\}, \ldots \{x_N, y_N\}$, $x \in \mathbb{R}^d$ where d is the dimension of the vector x.

If the two classes are linearly separable, one can find an optimal weight vector w^* such that $\|w^*\|^2$ is minimum, subjected to

$$y_i(w^* \bullet x_i - b) \geq 1 \tag{1}$$

The training examples satisfying the equality in Eq. (1) are termed support vectors. Support vectors define two hyperplanes in feature space to separate the two classes. The distance between the two hyperplanes defines a margin which is maximized when the norm of the weight vector $\|w^*\|$ is minimum. An intuitive illustration of the margin is shown in Figure 1. It has been shown that the minimization of $\|w^*\|$ can be carried out by maximizing the following function with respect to the variables α_j.

$$W(\alpha) = \sum_{i=1}^{N} \alpha_i - 0.5 \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j (x_i \bullet x_j) y_i y_j \qquad s.t. \quad \alpha_j \geq 0 \tag{2}$$

where symbol $\bullet$ represents the dot product. x_j with positive α_j is called as a support vector.

To classify a newly incoming unknown vector x_j, one can find

$$F(x_j) = sign\{w^* \bullet x_j - b\} \tag{3}$$

where

$$w^* = \sum_{i=1}^{r} \alpha_i y_i x_i \tag{4}$$

where r is the number of support vectors whose α's are nonzero.

If the two classes are linearly nonseparable, training errors are allowed, instead we should minimize

$$\|w^*\|^2 + C \sum_{i=1}^{N} \xi_i \qquad s.t. \quad y_i(w^* \bullet x_i - b) \leq 1 - \xi \qquad \xi > 0 \tag{5}$$

where ξ is a slack variable.

In this case, we allow training examples to exist between the two hyperplanes that go through the support vectors of two classes. We can equivalently minimize $W(\alpha)$ subject to $0 \leq \alpha_i \leq C$ instead of $\alpha_j \geq 0$. Maximizing $W(\alpha)$ can be carried out by using quadratic programming techniques, some of which are particular to SVM (Platt, 1998; Osuna et al., 1997).

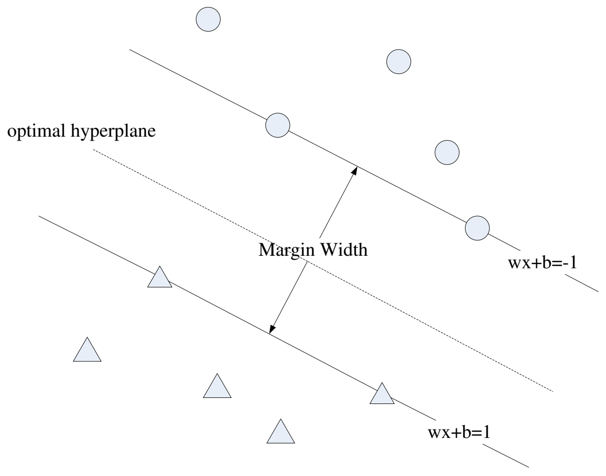


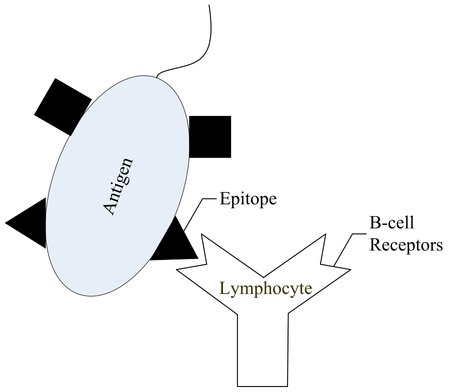
Fig. 1. The optimal hyperplane and margin in feature space, $wx + b = 1$ *and* $wx + b = -1$ represent separating hyperplanes where support vectors locate, respectively.

3.2 *Artificial Immune System*

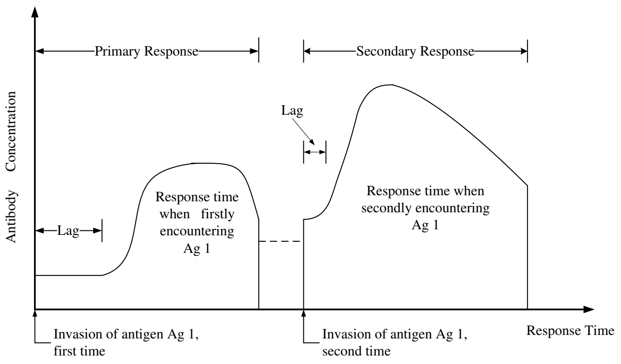
The main goal of human immune system (HIS) is to distinguish self from non-self such as bacteria, viruses, fungi, etc. For this purpose, HIS uses the specialized white blood cells, e.g., B and T *lymphocytes*. On the surface of each lymphocyte there are *receptors*. The lymphocyte can be activated by binding the receptor to the patterns presented on the surface of *antigens*. The binding process is schematically shown in Figure 2(a).

Lymphocytes are created in the bone marrow. The match between a receptor and an antigen may not be exact, so a single lymphocyte may recognize a number of antigenic patterns, which brings HIS the nature of noise tolerance. Generally, the similarity between receptor and antigen is measured by *affinity*. When the binding of the receptor to antigens takes place, the binding stimulates an immune response of *clone selection*. In this process, the cloning takes place with a rate proportional to the affinity, while the mutation rate is inversely proportional to the affinity (De Castro and Timmis, 2002; Tan and Xiao, 2007). The binding receptor making a correct decision, i.e., the matched antigen is real pathogenic, is promoted to be a memory cell for an immune memory of the encountered antigen. It takes longer for HIS to mount a response to an antigen for the first time when a new pathogen is encountered, e.g., the process of *Primary Response*. Yet, when the same pathogen or its likeness appears again, the HIS can respond very quickly because of the memory cells and their mutants, e.g., the process of *Secondary Response*. Figure 2(b) depicts the immune response processes in HIS.

Over the last few years, a novel paradigm of computational intelligence called Artificial Immune System (AIS) has received extensive attention from researchers. Inspired by the human immune system, AIS seeks to use the observed immune components and processes as metaphors to build artificial systems to encapsulate the desirable properties of HIS. The AIS have already been applied to a wide variety of domains, including: machine learning, pattern recognition, optimization, data mining, and computer security (De Castro and Timmis, 2002; Tan and Xiao, 2007; Dasgupta and Attoh-Okine, 1997).



(a) Binding of the receptor to patterns presented on the antigen


Primary Response
Secondary Response
Antibody Concentration
Lag
Lag
Response time when firstly encountering Ag 1
Response time when secondly encountering Ag 1
Invasion of antigen Ag 1, first time
Invasion of antigen Ag 1, second time
Response Time

(b) Primary response and secondary response in HIS

Fig. 2. Schematic diagrams of binding and immune response processes of human immune system (HIS)

4. Principle of EM-Update and Sliding Window

In this section, the three main principles employed in our algorithms are elaborated, i.e., exceeding margin update (EM-Update) technique, sliding window technique and immune response process of HIS.

4.1 *EM-Update*

In pattern recognition, one would like to consider more data or examples simultaneously to estimate the underlying class distribution as accurately as possible. However, for spam detection, there is a very small amount of training data available. Furthermore, data of emails are in stream mode and arrive successively. Thus incremental techniques are required to update of the stream data.

SVMs have been successfully used as a classification tool in a large variety of practical domains. The representation of the data is given by the set of support vectors along with corresponding weights. The number of support vectors in a SVM is much smaller than the total number of training examples, therefore, the support vectors provide a compact representation of the data (Domeniconi and Gunopulos, 2001). At each incremental step, the support vectors which describe the essential class boundary information together with the new incoming data, are used as training set for the SVM update.

There are several techniques for the incremental learning of SVM, including: *Error-driven technique (ED)*, *Fixed-partition technique (FP)*, *Exceeding-margin technique (EM)*, and *Exceeding-margin+error technique (EM+E)* (Mitra et al., 2000; Syed et al., 1999). The experimental results on Large-noisy-crossed-norm data and real data set Pima taken from UCI Machine Learning Repository, show that *EM* technique achieves similar error rate compared to the other three techniques, yet, the number of support vectors is small by comparison (Syed et al., 1999). Therefore, *Exceeding-margin technique (EM)* is adopted to update the SVMs in algorithm.

Given the model SVM$_t$ at time t, the algorithm checks if new data (x_i, y_i) exceeds the margin defined by SVM$_t$, i.e., $y_i(w^* \bullet x_i - b) \leq 1$. If the condition is satisfied the data point is kept, otherwise, it is discarded. Once a given fixed number n_e of data which exceed the margin is collected, the update of SVM$_t$ takes place, i.e., the support vectors of SVM$_t$, together with the n_e data points, are used as training data to obtain a new model SVM$_{t+1}$ at time $t+1$.

4.2 *Work Process of Sliding Window*

In one hand, the EM technique is employed to construct the incremental learning algorithm. On the other hand, data points which are no longer effective or usable have to be purged. The characteristic of the data in stream mode changes rapidly with time, and historical data may not be a good predictor for future data points. Here, a *sliding window* is used to purge the out-of-date knowledge (Domeniconi and Gunopulos, 2001).

We consider the incoming data in batches with size b, and maintain w models to represent the previous $1, 2, \ldots, w$ batches in the window simultaneously. Thus, the window hold W $(= wb)$ data points. The w models are updated incrementally and independently according to the EM technique once data become available. Let us denote the w models, at time t, as SVM$_1^t$, SVM$_2^t$, . . ., SVM$_w^t$, respectively. When a new batch of data arrives at time $t+1$, SVM$_w^t$ is discarded, meanwhile, the remaining SVM$_1^t, \ldots,$SVM$_{w-1}^t$ become SVM$_2^{t+1}, \ldots,$SVM$_w^{t+1}$, sequentially. SVM$_1^{t+1}$ is newly created by only using the data of the previous batch. This process can be formulated as

$$SVM_{i+1}^{t+1} = \begin{cases} SVM_i^t & 1 \leq i \leq w-1 \\ \text{created by batch at time t} & i = 0 \end{cases}$$

where w is the size of the window. Figure 3 depicts an intuitive illustration of this process of the sliding window.

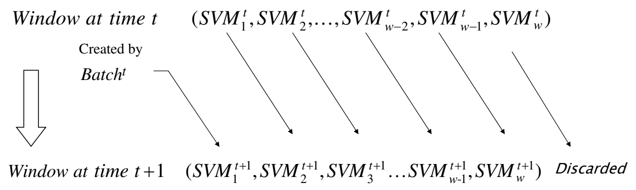

Window at time t
$(SVM_1^t, SVM_2^t, \ldots, SVM_{w-2}^t, SVM_{w-1}^t, SVM_w^t)$
Created by
Batch^t
Window at time $t+1$
$(SVM_1^{t+1}, SVM_2^{t+1}, SVM_3^{t+1} \ldots SVM_{w-1}^{t+1}, SVM_w^{t+1})$
Discarded

Fig. 3. The slide process of the window with size w

Each SVM in the window represents the recent batches of data seen so far. Specifically, SVM$_1^t$ represents the latest batch of data while SVM$_w^t$ represents the previous w batches of data. When a new data point arrives, each SVM in the *window* classifies the data independently and has an equal weight. The label of the data is decided by the strategy of majority voting. Here, SVMs in the window can be considered as ``experts'' with different knowledge and work together to give out the final decision for the new data.

A more complicated strategy of voting, weighted majority voting, could be adopted to weight each SVM in the window. The weight could be set at different values according to the feature of the data and could be adjusted dynamically. When the characteristic of the data stream drifts dramatically, we can increase the weights of ``younger'' classifiers such as SVM$_1^t$,SVM$_2^t$, to reflect the immediate change of the data. On the contrary, when the variation of the data is slight, it will benefit from increasing the weights of ``older'' classifiers because they have seen more data and have more amount of knowledge. In practice, the changing trend may be smooth or not, therefore, the update of weights of SVMs must be a dynamic process.

4.3 *Primary Response and Secondary Response*

The HIS takes longer time to mount a response a pathogen encountered for the first time. This process is called *Primary Response*. The detectors that detect the pathogens successfully undergo the immune response of clone selection. They are promoted to be a long-lived memory cell to memorize the encountered pathogen. After a primary response, the HIS can react immediately when a similar pathogen intrudes again. This process is called the *Secondary Response*. The concept of a memory cell is used as a metaphor in algorithm. The purpose is to recognize the previously met spam. There are three ways to generate initial detector set. First, use a negative selection algorithm. Second, use a positive selection algorithm. Third, use a positive-negative selection algorithm. The negative selection algorithm constructs the initial detector set from negative samples. Some of representative samples and their mutants can directly serve as detectors. The positive selection algorithm only uses the positive samples for this purpose. The positive-negative selection algorithm is a combination of the two ways mentioned above. Both positive samples and negative samples are used to generate detectors. The size of positive detector set relative to the size of the negative set is set according to the real application.

5. Implementation of Algorithms

5.1 *Motivation*

We consider HIS and SVM as inspiration because their properties are particularly suitable for developing a spam filter. The similarities between the immune system and a spam detection system are:

1. **Pattern Recognition**: The goal of spam detection is to distinguish spam mails from non-spam mails while HIS differentiates self from potentially dangerous non-self.
2. **Dynamical Change**: The formats of the e-mail vary, and the content of e-mail and user's interest are always changing. Tracing these changes is also the goal of spam detection. Similarly, HIS is capable of recognizing a variety of invaders dynamically.
3. **Noise Tolerance**: A desirable feature in pattern recognition is noise tolerance, while the immune system is capable of recognizing different mutants of pathogens.

SVM is a prominent classifier with a solid theoretical foundation of statistical learning and superior generalization of performance. Moreover, there are many successful applications based on AIS and SVM techniques. Thus, the proposed algorithms in this paper employ the incremental SVM technique and the concepts of detectors, as well as, memory cells for uninterrupted spam detection.

5.2 *Algorithm Overview*

The incoming e-mails arrive successively in a stream mode. For the processing of this kind of data, an incremental update technique is required for the incorporation of new knowledge from the data stream. Since the content of the user's e-mail and interest constantly drift, we have to forget the data points that are no longer in active. The key idea of algorithm is to employ a sliding window to hold several classifiers simultaneously, meanwhile, each classifier in the window updates independently through the EM technique. When a new e-mail arrives, it is classified independently through the classification criterion, which will be presented in details in section 5.6. The label of the new e-mail is determined by majority voting. The out-of-date knowledge is purged when a new batch of e-mail arrives. The ``oldest'' classifier, the one farthest right, is removed from the window while the remaining classifiers slide to the right. The ``youngest'' classifier, the one farthest left, is newly created by using the previous batch. The window must be initialized to start the work. Each email needs to be transformed to a feature vector before it is classified. The overview of the algorithm is presented in Algorithm 1, each step of it is described in detail in the following sections.

Algorithm 1 Algorithm for Spam Detection

```
Preprocess each message in the Training Set
Initialize the window, namely generate the $classifier_1$

while Algorithm is running do
  if message is received then
    for each classifier in the window do
      classify the message independently
      if the message exceeds the margin then
        the message is kept
      else
        the message is discarded
      end if
      if a given number of messages exceeding the margin is collected then
        update the classifier
      end if
    end for

    the label of the message is given by classifiers in the window in terms of voting
    if the received message is the last message of current batch then
      Slide the window
      generate the $classifier_1$ by current batch only
    end if
  end if
end while
```

5.3 *Message Representation*

The first stage of pattern recognition is the preprocessing of the data. The problem of e-mail classification is how to represent the message. A word in the message is defined as a feature, and the message can be easily represented as a feature vector through a bag of words that is formed by analyzing a set of training samples. There are serval methods to construct a feature vector, such as, TF (Term Frequency), TF-IDF (Term Frequency--Inverse Document Frequency), and binary representation. Drucker et al. showed that SVM with binary features outperforms other methods (Drucker, 1999). Here, we adopt the binary representation of a message in algorithm. When we use w_i to represent a word, the message can be expressed as

$$message = (w_1, w_2, \cdots, w_m) \tag{6}$$

where m is the number of words in the bag, and w_i, $(i = 1, 2, \ldots, m)$ takes binary value of 1 or 0 to indicate whether it occurs in this message or not.

5.4 *Dimension Reduction*

If we extract each word appearing in a message to form a dictionary, the size of this dictionary will be too large. Thus, a dimension reduction of the feature space is required to avoid the curse of dimension. According to (Zuchini, 2003), the features that appear in most of the documents are not relevant to the separation, because all the classes contain those same features. On the contrary, the words used rarely also give us few information during classification either. Therefore, for simplification, we adopt the processing method in (Bezerra et al., 2006), which discards the features that appear less than 5% and more than 95% in all messages of the corpus.

5.5 *Initialization of the Window*

At beginning, the window is empty. The initialization of the window is to generate the $classifier_1$. For each classification criterion, there is a different construction of the classifier. The different initialization processes correspond to specific classification criteria described in the following paragraphs. Here, some parts of the corpus are used as training samples to generate the $classifier_1$.

For the classification criteria of **Hamming Distance** and **Included Angle**, the classifier is comprised of *Detector Set* and *Memory Set*. At first, each mail is transformed into a binary vector in the above mentioned way. After pre-processing, the training vectors are used to train the SVM. The *support vectors* of the trained SVM are used as *naïve detectors* to form the *Detector Set*.

The *Memory Set* is a set of *memory cells* that have better performance. The *Detector Set* is used to classify the training samples according to the classification criterion of Hamming Distance or Included Angle. The *naïve detector* is promoted to a memory cell when the number of its correctly classified messages exceeds the presetting threshold n_m, and it is then removed from current *Detector Set*. Each member of *Detector Set* and *Memory Set* is labeled identically with its corresponding *support vector*. Each memory cell is assigned a *lifespan*. There are two sorts of support vectors, one sort represents non-spam and the other sort represents spam. The way of the generation of detectors adopted here is an implementation of a positive-negative selection algorithm. An overview of the generation process of *Detector Set* and *Memory Set* is described in Algorithm 2.

Algorithm 2 Generation of Detector Set and Memory Set

```
Detector Set ⇐ $\phi$
Memory Set ⇐ $\phi$
Convert each message in the Training Set to a binary vector
Use the preprocessed Training Set to train the SVM
Detector Set ⇐ Support Vectors of the trained SVM

for each msg $\in$ Training Set do
    Use Detector Set to classify msg in terms of certain criterion
    for each naïve detector participating in decision making and judging correctly do
        Increment the number of correctly classified messages of naïve detector
    end for
end for

for each naïve detector $\in$ Detector Set do
    if the number of correctly classified messages exceeds the threshold $n_m$ then
        Assign a lifespan to this naïve detector
        Memory Set ⇐ Memory Set $\cup$ {naïve detector}
    end if
end for

Detector Set ⇐ Detector Set $-$ Memory Set
```

For classification criterion of **SVM**, there are no *Detector Set* and *Memory Set*. We just need to train the SVM by using the samples only.

5.6 *Classification Criterion*

Four kinds of classification criteria are developed as uninterrupted detection for spam. There are detailed performance comparisons made in the experiments. The criteria are described as follows.

1. **Hamming Distance.** This approach calculates the *Hamming Distance* between the message to be classified and each *naïve detector* and each *memory cell*. Each one that meets the minimum distance is added to a set called the *Committee*. Each member of the *Committee* votes through its label. The final label of the message is determined by majority vote of the *Committee*. In circumstances where the votes of the two sides are equal, the message is classified as non-spam. This action is taken because misclassifying a non-spam as a spam is much more serious than the opposite. Under the circumstance of using the binary representation of the feature vector, the minimum *Hamming Distance* is equivalent to the minimum *Euclidean Distance*. Thus, this decision making is equivalent to *Nearest Neighbor* (NN). Also, there is an optional procedure called *mutation*, in which each member in the *Committee* can be mutated. The committee member that makes one correct decision is mutated to get closer to the message in the feature space. The committee member that makes a mistake is moved farther away from the message. This idea is implemented by changing some entries of the vector to be identical with or different from the corresponding entries of the message. The number of mutated entries is proportional to the mutation *rate* which is preset in advance. The procedure of classification using *Hamming Distance* with mutation is outlined in Algorithm 3.

2. **Included Angle.** In this approach, the included angle is used as the measure of classification. We compute the cosine value of included angle between the message to be classified and each *naïve detector* and each *memory cell*. The one with maximum cosine value is added to the *Committee* set. The following process is just the same as *Hamming Distance* without mutation procedure.

3. **SVM.** This method directly uses SVM to classify the message. It locates and classifies the message on the *optimal hyperplane* as shown in Figure 1.

4. **Weighted Voting.** This method is a combination of *Hamming Distance*, *Included Angle* and *SVM*. For *Hamming Distance* and *Included Angle*, *Committee* voting can be considered voting at the first level. *SVM* is regarded as the voting of *support vectors* in Eq. (3). The idea of weighted voting is to use the results of above three to vote at the second level, while the weight of each single method can be preset in advance and updated dynamically according to its performance. Specifically, the approach that performs best can be weighted higher the others lower.

Algorithm 3 Classification using Hamming Distance with mutation

```
Input: The message waiting for classification.
Output: The label of the message.

for each detector $\in$ (Detector Set $\cup$ Memory Set) do
    Compute the hamming distance between detector and message
    Elements_Iden $\Leftarrow$ positions of entries identical with message
    Elements_Diff $\Leftarrow$ positions of entries different from message
end for

Add the ones with minimum hamming distance to set Committee

if Number of detectors with non-spam label $\geq$ those with spam label then
    labelofMsg $\Leftarrow$ non-spam
else
    labelofMsg $\Leftarrow$ spam
end if
{The following is the process of mutation, it is optional}
for each detector $\in$ Committee do
    if detector correctly classifies the message then
        num_mutate $\Leftarrow \lceil size\ of$ Elements_Diff $*\ ratio \rceil$
        repeat
            pos $\Leftarrow$ Randomly pick up a position not picked previously from Elements_Diff
            Take NOT operation to the entry of vector of detector at pos
        until num_mutate times
    else
        num_mutate $\Leftarrow \lceil size\ of$ Elements_Iden $*\ ratio \rceil$
        repeat
            pos $\Leftarrow$ Randomly pick up a position not picked previously from Elements_Iden
            Take NOT operation to the entry of vector of detector at pos
        until num_mutate times
    end if
end for
```

5.7 *Update of the Classifier*

The task of on-line e-mail classification is actually to process data in stream mode. The content of e-mails and user's interests change constantly, and a classifier built from previous data may not always be suitable for future data. Therefore, *EM technique* is adopted for incremental update of the classifier.

Given a model $classifier_t$ at time t, once the number of messages exceeding the margin is equal to or greater than n_e, the update of $classifier_t$ takes place. For classification criteria of *Hamming Distance* and *Included Angle*, the *Detector Set* and *Memory Set* of $classifier_t$, together with n_e messages, are used as training data to obtain a new model $classifier_{t+1}$. The *memory cells* with positive *lifespan* are reserved during each update, regardless whether the memory cells are the newly derived support vectors or not. When SVM is used as the classification criterion, we only need to construct the SVM$_{t+1}$ of $classifier_{t+1}$, by using the *support vectors* of SVM$_t$ and n_e

messages as training data, because there are no *Detector Set* and *Memory Set* in this case. Algorithm 4 lists this process in details.

Algorithm 4 Update of the $classifier_t$

```
Input: $classifier_t$
Output: $classifier_{t+1}$

$DetectorSet_{t+1} \Leftarrow \phi$
IncrementalMem $\Leftarrow \phi$
$MemorySet_{t+1} \Leftarrow$ memory cells in $MemorySet_t$ whose lifespan $> 0$
Training Set $\Leftarrow DetectorSet_t \cup MemorySet_t \cup \{n_e\ messages\}$
SVM$_{t+1} \Leftarrow$ SVM constructed from Training Set
$DetectorSet_{t+1} \Leftarrow$ Support Vectors derived from SVM$_{t+1}$

for each msg $\in$ Training Set do
  Use $DetectorSet_{t+1}$ to classify msg in terms of certain criterion
  for each naïve detector participating in decision making and judging correctly do
    Increment the number of correctly classified messages of naïve detector
  end for
end for

for each naïve detector $\in DetectorSet_{t+1}$ do
  if the number of correctly classified messages exceeds the threshold $n_m$ then
    IncrementalMem $\Leftarrow$ IncrementalMem $\cup$ {naïve detector}
  end if
end for

$MemorySet_{t+1} \Leftarrow MemorySet_{t+1} \cup$ IncrementalMem
Assign a lifespan to each member of $MemorySet_{t+1}$
$DetectorSet_{t+1} \Leftarrow DetectorSet_{t+1} -$ IncrementalMem
```

Through the duration of two successive EM updates, there will be some slight updates for the *Detector* and *Memory Sets*. The number of its correctly classified messages is increased one when *naïve detector* makes a correct classification. The one whose number of correctly classified messages exceeds the threshold n_m is added to *Memory Set* immediately. The lifespan of each memory cell decreases one after each classification. Therefore, slight updates are not needed for SVM classification criterion.

5.8 *Purge of Out-Of-Date Knowledge*

As sliding window is used for tracing the changes in the content of e-mail, user's interests, and purging knowledge no longer in active use. In this paper, the incoming e-mail stream is considered batches with a given size of b. A window includes w classifiers which are built by the previous $1, 2, \ldots, w$ batches, respectively. Each classifier in the window has a corresponding weight, which denotes the relative importance. Each classifier updates independently according to EM technique when the number of messages exceeding the margin is equal to or greater than n_e. When a new e-mail arrives, each classifier classifies it independently using some criterion described in section 5.6 and give a label to this e-mail accordingly. The final label of the e-mail is determined by a strategy of majority voting. To purge of out-of-date knowledge, when a new batch arrives, the ``oldest'' classifier, the $classifier_w$, is removed from the window while remaining classifiers move one step to the right. The ``youngest'' classifier, the $classifier_1$, is newly generated by using the data of previous batch only.

6. Experiments

6.1 *Corpora Used In Experiments*

Two corpora used to test the proposed approaches in this paper are the PU1 corpus (Androutsopoulos et al., 2000) and Ling corpus[1] (Androutsopoulos et al., 2000).
PU1 corpus consists of 1,099 messages, with spam rate 43.77%, which includes:

- 481 spam messages. These are all the spam messages over a period of 22 months, excluding non-English messages and duplicates of spam messages sent on the same day.
- 618 legitimate messages, all in English, over a period of 36 months.

Ling corpus consists of 2,893 messages, with spam rate 16.63%, which includes:

- 481 spam messages. Duplicate spam messages received on the same day are not included.
- 2,412 legitimate messages, obtained by randomly downloading digests from the archives, messages are separated from digests, and texts added by the server are removed.

All the messages in both corpus have header fields, attachment and HTML tags removed, leaving only subject line and mail body text. In PU1, each token is mapped to a unique integer to ensure the privacy of the content while keeping its original form in Ling. Each corpus is divided into ten partitions with approximately equal amount of messages and spam rate. There are four versions of the corpus: with or without stemming and with or without stop-word removal. Stop-word removal is a procedure to remove most frequent used words such as `and', `for', `a', and the stemming is the process of reducing a word to its root form, e.g., `teacher' becomes `teach'. Androutsopoulos et al. demonstrate that stop-word removal and stemming may not promote a statistically significant improvement of spam detection (Androutsopoulos et al., 2000). Therefore, the original version without stemming and stop-word removal is adopted in the experiments.

6.2 *Performance Measures*

For performance evaluation, the following classic measures listed in Table 1 are adopted throughout this paper. Accuracy is defined as the percentage of messages classified correctly. Precision is defined as the proportion of the number of correctly-classified spam messages to the number of messages classified as spam. Recall is defined as the proportion of the number of correctly-classified spam messages to the number of messages originally categorized as spam. When filtering spam misclassifying a legitimate mail as spam is much more severe than letting a spam message pass the filter. Miss Rate is used to represent the proportion of misclassified legitimate messages to the number of messages originally categorized as legitimate. We adopt notations in (Yeh et al., 2005) and list measures of accuracy, recall, precision, and miss rate in Table 1. The speed index is the time of training and testing as speed index.

[1]The PU1 corpus and Ling corpus may be downloaded from http://www.iit.demokritos.gr/skel/iconfig/

Table 1. Performance measures in the experiments. TP is the number of spam emails which are correctly predicted as spam; FN is the number of spams which are predicted as non-spam; TN is the number of normal e-mails which are predicted as non-spam; and FP is the number of normal e-mails which are predicted as spam.

Measure	**Expression**
Accuracy	(TP + TN) / (TP + FP + FN + TN)
Precision	TP / (TP + FP)
Recall	TP / (TP + FN)
Miss Rate	FP / (FP + TN)

6.3 *Experimental Results*

The parameters of algorithm and its values used in the following experiments are all shown in Table 2. A legal range for each parameter is also given. The values of these parameters are obtained by trial and error during the initial verification. A range of values for each parameter is tested in the initial experiments. The values of these parameters shown in Table 2 are the ones that work well in the initialization, which then are adopted in the experiments in this paper. The initial experiments of parameters with different values led to some differences in performance. Thus, a certain analysis is given for the influence of each parameter on the performance.

The size of the window is set to be odd to avoid an equal number of votes one each side. When the size of the window is larger than some threshold, the computational speed will decrease because the update of the classifiers consumes more CPU time (because there are more classifiers held in the window), yet the performance does not improve on noticeably or even gets worse. Therefore, a window with good performance and fast speed is desirable. A relatively small window (with window size 3 or 5) is adopted with competent performance in the experiments.

The size of the batch should be set based on the characteristics of the data stream. When the data changes dramatically, the size of the batch should be small to purge of out-of-data knowledge (i.e., the slide process of the window) can be carried out in time. A large batch size can be adopted for a smooth data stream. The variations of the corpus PU1 and Ling used in the experiments are not big and obvious, therefore, a relatively large batch size (i.e., 60) is adopted in the experiments.

The number of messages exceeding the margin (n_e) should not be too small or too large. When n_e is too small, noises will cause vibration and the frequent update of the classifiers is too time consuming. When n_e is too big, the classifier can not be updated in time to reflect or capture the novel data distribution. The vibration and the delayed update degrade the performance severely. Therefore a proper n_e should be chosen for the experiments. A ne of 30 balances the frequency of classifier update and the vibration. Thus, $ne = 30$ is used in all experiments.

Similar to n_e, the threshold of promotion for a naive detector (n_m) should be set properly. When n_m is too s-mall, it is very easy for a naive detector to satisfy the threshold, but many of the promoted naive detectors are not representative (those matching only a small amount of messages). This fact lowers the effectiveness of the memory set. When n_m is too large, it will be too difficult for a naive detector to become a memory cell, in turn, the size of the memory may be zero. The goal is to maintain an efficient memory set with relatively small size. Testing showed the value 5 meets this requirement well.

The lifespan of the memory cell Lifespan controls the aging process of memory cells. This value should not be too small, otherwise, memory cells will make no contribution to the classification because they have been purged before matching with similar messages. If the lifespan is too long, the useless memory cells can not be expelled in time. The value of lifespan and batch size are set to be same in these experiments. This fact implies that a memory cell correctly classifying only one piece of email in a batch can survive.

Furthermore, the ratio for mutation must be set properly in experiments. A ratio that is too small will lead to the coverage of a mutant nearly the same as the mutated one, which only increases the space used without any improvement in performance. Yet, the mutants generated in too large of a ratio may be dramatically different from the original one but still have the same label. This fact leads to a wrong e-mail classification. The ratio is set at 5% in these experiments.

LIBSVM software package is used for the implementation of the SVM (Chang and Lin, 2001). The linear kernel, polynomial kernel, RBF kernel and sigmoid kernel are tested in the initial experiments. The difference in the performances of different kernels is slight. The included angle classification criterion is only feasible on linear kernel, in turn, the linear kernel is adopted in the experiments. The parameter C of SVM is determined as follows. The range of the parameter C for the test is in the integer interval [1, 65]. C is picked where it achieves the highest classification accuracy on the training samples as the final C value. The accuracy on the training samples is measured by 10-fold validation.

The weights of classifiers in the window are set to be identical. All experiments are conducted on a PC with CPU of AMD Athlon 3200+ and 448M RAM. The experiments compare the different classification criteria described in section 5.6. The eight methods to be compared in the experiments are listed in Table 3.

Table 2. Parameters and its values of proposed approaches

Parameter	Value	Range
w (size of the window)	3 or 5	≥ 1
b (size of the batch)	60	≥ 1
n_e (number of messages exceeding the margin)	30	≥ 1
n_m (threshold of promotion for a naïve detector)	5	≥ 1
lifespan (lifespan of the memory cell)	60	≥ 1
ratio (the ratio for mutation)	5%	$[0, 1]$

The SVM on the farthest right of the window only (M5) is the approach mentioned in (Ahn et al., 2004). A *window* is maintained and each *classifier* in the *window* updates independently according to EM technique. Yet, only the $classifier_w^t$ is used to predict the label of the new coming e-mail.

The SVM without a window (M6) can be regarded as a special case because the *window* size is one. The classification is performed successively by the unique SVM classifier, which updates itself according to EM technique.

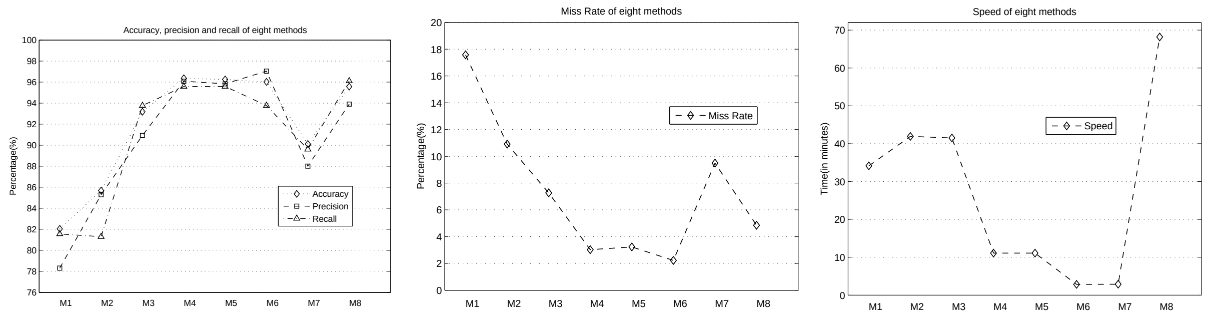
The SVM with 90% Supports Vectors without window (M7) is a method which makes a tradeoff between performance and the speed of execution. The *support vectors* are sorted in descending order according to the corresponding coefficients, which indicate the relative importance of the *support vectors*. The 10% of *support vectors* at the rear of the sorted queue are discarded and the other 90% support vectors are used to compute w^* by Eq. (4). The other processes are just the same as M6.

We choose different combinations of partitions in each corpus to construct training set and testing set, the window size is 3 or 5, respectively. Figure 4 shows the accuracy, precision, recall, miss rate and speed of the eight

Table 3. Eight methods with different classification criterion for comparison in the experiments

Method	**Description**
M1	The Hamming as classification criterion without mutation
M2	The Hamming as classification criterion with mutation
M3	The included angle as classification criterion
M4	The SVM as criterion
M5	The SVM on the farthest right of the window only
M6	The SVM without a window
M7	The SVM with 90% supports vectors without window
M8	The weighted voting as classification criterion

methods on PU1. Partitions 1-2(219 messages) are used as training set. The partitions 3-10(880 messages) are used as testing set, and 5 is used as the window size. Figure 5 shows the performance measures of the eight methods on Ling. The partitions 9-10(580 messages) are used as training set. Partitions 1-8(2313 messages) are used as testing set, and 3 is used as the window size. The performances of the experiments are averaged on different combinations of partitions of corpus and are listed in Table 4 to Table 7.

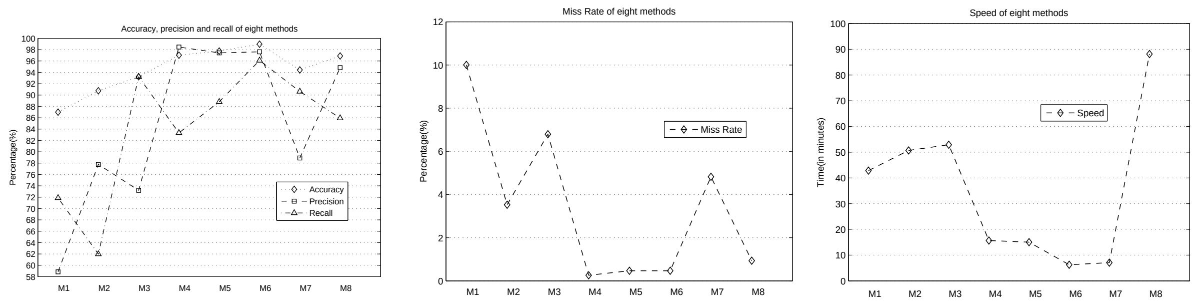


(a) Accuracy, Precision and Recall (b) Miss Rate (c) Speed

Fig. 4. Accuracy, Precision, Recall, Miss Rate and Speed on Testing Set of partitions 3-10 (880 messages), using partitions 1-2 (219 messages) as Training Set with window size 5, on corpus PU1

Figure 6(a) shows the variations of classifier's *Detector Set* and *Memory Set* during its lifetime from initially generated by batch 1 to being removed from *window* on PU1. The *included angle* is used as classification criterion and 5 as window size. Figure 6(b) shows, for SVM classification criterion, the variation of classifier's *support vectors* during its lifetime from initially generation by batch 1 to removed from the *window*. Abscissa 1 denotes the generation of the classifier and other abscissas indicate each time when the *EM-Update* technique is used to update the classifier.

Table 8 and Table 9 show the performances of Naïve Bayesian, Linger-V and SVM-IG on corpus PU1 and Ling reported in (Androutsopoulos et al., 2000; Clark et al., 2003; Androutsopoulos et al., 2000; Chang and Lin, 2001). Linger-V is a NN-based system for automatic e-mail classification. The Naïve Bayesian, the original version of corpus is adopted, while Linger-V and SVM-IG use stemming version. All these results are obtained by using 10-fold validation. In (Androutsopoulos et al., 2000; Clark et al., 2003; Androutsopoulos et al., 2000; Chang and



(a) Accuracy, Precision and Recall (b) Miss Rate (c) Speed

Fig. 5. Accuracy, Precision, Recall, Miss Rate and Speed on Testing Set of partitions 1-8 (2313 messages), using partitions 9-10 (580 messages) as Training Set with window size 3, on corpus Ling

Table 4. Performances of eight methods on corpus PU1 with window size 3

Methods	**Accuracy** (%)	**Precision** (%)	**Recall** (%)	**Miss Rate** (%)
M1	80.758	77.2524	79.471	18.239
M2	83.4483	82.8473	78.4758	12.6785
M3	91.0182	88.1228	91.8624	9.6393
M4	95.7811	95.4235	94.9131	3.5426
M5	95.7345	95.0696	95.1905	3.8418
M6	96.2926	96.5779	94.9422	2.6567
M7	92.9538	90.646	93.7673	7.6819
M8	94.3057	93.309	93.7322	5.2467

Table 5. Performances of eight methods on corpus PU1 with window size 5

Methods	**Accuracy** (%)	**Precision** (%)	**Recall** (%)	**Miss Rate** (%)
M1	81.5696	79.2014	78.5389	16.0689
M2	86.2257	87.3313	80.2468	9.1166
M3	91.6421	89.4713	91.6916	8.397
M4	96.1644	96.3969	94.7746	2.7533
M5	96.3495	95.9985	95.708	3.1519
M6	96.2926	96.5779	94.9422	2.6567
M7	92.9538	90.646	93.7673	7.6819
M8	95.2396	95.2084	93.8697	3.6927

Table 6. Performances of eight methods on corpus Ling with window size 3

Methods	**Accuracy** (%)	**Precision** (%)	**Recall** (%)	**Miss Rate** (%)
M1	86.0903	56.4391	77.309	12.1604
M2	90.8851	77.0192	64.3103	3.8227
M3	92.4888	71.1464	93.1514	7.643
M4	97.0434	97.5746	84.2888	0.4167
M5	97.7805	96.4355	90.7653	0.657
M6	98.7186	96.0549	96.2479	0.7897
M7	96.3315	84.6912	95.5456	3.5119
M8	96.6121	91.5655	87.9770	1.6677

Table 7. Performances of eight methods on corpus Ling with window size 5

Methods	**Accuracy** (%)	**Precision** (%)	**Recall** (%)	**Miss Rate** (%)
M1	87.0358	59.1022	77.9555	11.1557
M2	92.2556	83.6818	66.1891	2.5532
M3	92.8725	72.0113	94.0351	7.3591
M4	97.6485	97.0959	88.483	0.5264
M5	98.0584	95.4202	92.7775	0.8865
M6	98.7186	96.0549	96.2479	0.7897
M7	96.3315	84.6912	95.5456	3.5119
M8	97.1328	91.9936	90.7708	1.6

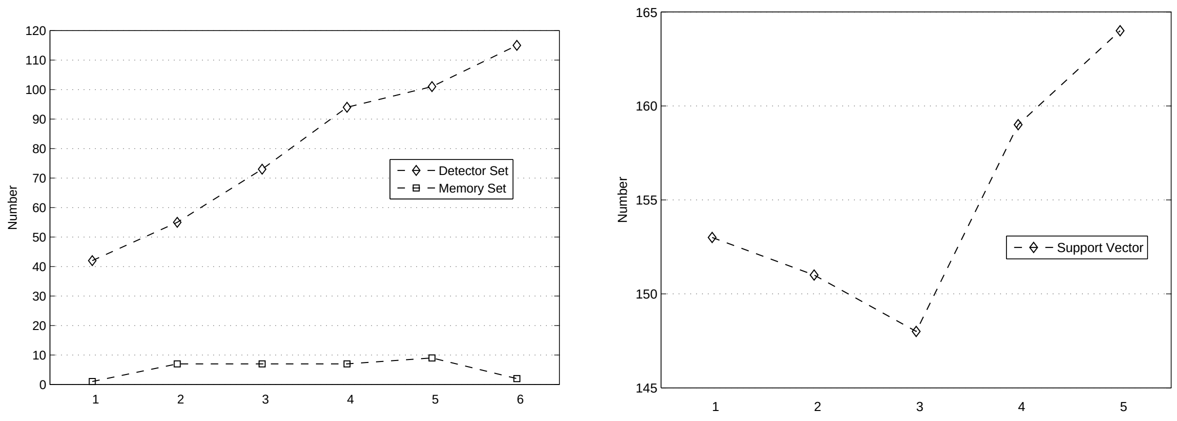
Table 8. Performances of Naïve Bayesian (NB), Linger-V and SVM-IG on corpus PU1, using 10-fold cross validation

Methods	**Accuracy** (%)	**Precision** (%)	**Recall** (%)	**Miss Rate** (%)
NB	91.076	95.11	83.98	3.4
Linger-V	93.45	96.46	88.36	2.588
SVM-IG	93.18	95.7	88.4	3.1

Table 9. Performances of Naïve Bayesian (NB), Linger-V and SVM-IG on corpus Ling, using 10-fold cross validation

Methods	**Accuracy** (%)	**Precision** (%)	**Recall** (%)	**Miss Rate** (%)
NB	96.408	96.85	81.10	0.539
Linger-V	98.2	95.62	93.56	0.875
SVM-IG	96.85	99	81.9	0.17

Lin, 2001). , the authors only report the accuracy, precision and recall, the miss rate given here is derived by the given three indexes.



(a) Dynamic Change plots of the Sizes of Detector Set and Memory Set

(b) Dynamic Change plot of the Size of Support Vectors

Fig. 6. Variation of Detector Set, Memory Set and Support Vectors, using partitions 1-5 (549 messages) as Training Set, partitions 6-10 (550 messages) as Testing Set, on corpus PU1

7. Discussions

In this section, we analyze the experimental results in detail and give explanations for them. The eight methods are divided into three groups shown in Table 10.

Each method in AIS group has a *Detector Set* and a *Memory Set*, which is the inspiration for the HIS. Methods in SVM group use SVM as the basic classification tool. The Weighted Voting (WV) group contains the M8 approach, which combines the classification criteria of *hamming distance*, *included angle* and *SVM*.

Table 10. Groups of methods compared in experiments

Group	**Methods**
AIS	M1, M2, M3
SVM	M4, M5, M6, M7
WV	M8

The results on corpus PU1 show that methods M4, M5 and M6 in SVM group obtain a good performance on accuracy, precision, recall, and a miss rate below 5%. Performance of M7 degrades dramatically compared to M6, while an advantage in speed is not obvious. A possible explanation for this fact is that the number of *support vectors* is small as shown (Figure 6(b)). As a result, the number of the 10% small support vectors is also small, the absolute number of *support vectors* is not reduced remarkably, therefore, there is no improvement in speed.

When the number of *support vectors* is tremendously large, the advantage in speed may be increased.

The strategy of using classifiers in the *window* to vote (M4) does not show in advantage when compared to M5 and M6. This fact suggests that changes to the contents and user's interests do not present problem, or the variation of samples is too little in the PU1 corpus. When there are notable drifts (for example, a healthy person might have no interests on pharmaceutical at first, but he changes his interests on the pharmaceutical when he faces a health problem), the ``younger'' experts with latest knowledge, such as $classifier_1^t$, $classifier_2^t$, can reflect the immediate variation. Thus the drift can be tracked and a good performance can be achieved when a sliding window is used.

SVM group outperforms the other groups in performance and speed. The method for SVM M6 is the best one for performance and speed.

The performance of AIS group has been unsatisfactory. This is especially true for the *hamming distance* method. The original intention to construct a *Detector Set* and a *Memory Set* via SVM is to utilize the ``representative'' feature of *support vectors*. The *detectors* would be regraded as ``boundary detectors'' and the size of the set would be relatively small. When using *hamming distance* as the classification criterion, using minimum distance to label the e-mail (Nearest Neighbor) may not reflect the distribution of two classes as the optimal hyperplane in SVM does. Performance of *included angle* is superior to that of *hamming distance*. The limitation of this method is that only linear kernel can be used for SVM, otherwise, the cosine value cannot be computed for the included angle between two vectors. Another characteristic of AIS group is to have a *Memory Set*. Figure 6(a) shows that the size of the *Memory Set* is small and stable. The number of *memory cells* is proportional to the presetting threshold. The lower the threshold, the more *memory cells*. The value of lifespan and batch size are set to be same, which implies that a *memory cell* correctly classifying only one piece of mail in a batch can survive. The intention introducing a *Memory Set* is to maintain those ``outstanding'' *memory cells*, which are not updated according to EM-Update. Also lifespan is introduced to age and purge those *memory cells* no longer in active use. The curve depicting the size of *Detector Set* in Figure 6(a) shows the number of *detectors* increases after each EM-Update since the classifier is initially generated from batch 1. The explanation is based on the more data the classifier used, the more knowledge that is being accumulated, and more *detectors* are generated. Figure 6(b) shows the variation of *support vectors* for illustration. The curve drops then goes up. The reason is that SVM has the chance to use relatively more examples (549 messages) in the training stage, therefore, more *support vectors* can be generated. The first time the EM-Update, the number of *support vectors* together with messages exceeding the margin is relatively smaller than the training examples. Thus, the number of newly generated *support vectors* decreases accordingly. During the filtering process of the spam, the amount of the data increases, and the number of *support vectors* will increase as the data come in.

The M8 method in WV group is not better than M4 in SVM group. The reason for this is that the performance of the *hamming distance* and the *included angle* is not in accordance with SVM.

When the size of *window* increases from 3 to 5, the performance of AIS group and WV group raises, while the improvement of performance is not noticeable for the SVM group. Augmentation of the *window* makes the classifiers at the right of the window (the ``older'' experts) have more data. Furthermore, the bigger the size of the window, the more the data that can be used. As a result, if the content of the data does not drift dramatically, a considerable improvement of performance can be achieved.

In corpus Ling there is an unbalance between the numbers of negative examples and positive examples. As a consequence, the results on Ling differ from that on PU1 in several aspects. Each method gets higher accuracy and lower miss rate on Ling than on PU1. In particular, SVM group has lower than 1% miss rate. As the spam rate of Ling is only 16.63%, misclassifying a spam to legitimate mail has much more negative influence on precision and recall than that on accuracy.

Methods of AIS group do not give a better balance between accuracy, precision and recall as shown in Figure 5(a).

The proposed methods show competent performances compared to current methods listed in Table 8 and Table 9. It is noticable that the reported results of Naïve Bayesian, Linger-V and SVM are all obtained by using 10-fold cross validation. In each iteration, 90% of the data is used for training while the other 10% is used for its test. Therefore, the classifiers can estimate the underlying class distribution as accurately as possible. This action will achieve better performance. However, these methods may not work in practice because the data available for training is limited and only accounts for a very small percentage of the data. As shown in Figure 4 and Figure 5, when using only 20% of the data of the corpus for training, the performance of proposed methods is promising. Moreover, the proposed methods in this paper are dynamic, the update of the classifier and the slide of the window are carrying out constantly throughout the total running process. Therefore, these methods are remarkably suitable for most of the real applications of anti-spam programs.

8. Conclusions and Future Work

In this paper, uninterrupted detection approaches based on incremental SVM and AIS are proposed for spam in the e-mail stream. A sliding window is used to label e-mail and trace the dynamic change of the content of e-mail and user's interests. The final label of a new e-mail is given by majority voting. The exceeding margin update technique is also used for dynamic update of classifiers in the window. The sliding window is employed to purge of out-of-date knowledge. Eight methods for the uninterrupted detection are developed, these include: hamming distance (with or without mutation), included angle, SVM and weighted voting. Experiments have been conducted on two public benchmark corpora PU1 and Ling. Comparisons of performance among eight methods are made for accuracy, precision, recall, miss rate and speed. Extensive experimental results show that the proposed approaches give promising performances and will be effective and efficient tools in many practical applications for spam detection.

The ways of feature selection and dimension reduction in these approaches is simple. Some more sophisticated complicated techniques may be preferred in the future, these include: information gain, mutual information, term strength, and Chi squared (χ^2). Moreover, the e-mails in picture format and with attached pdf files have become more popular but there are few techniques for filtering such kinds of e-mails. Filtering these types of e-mails will be the work in future.

Acknowledgement

The authors are grateful to the Associate Editor and anonymous reviewers for providing invaluable comments and suggestions that greatly improved the quality of this paper.

This work was supported by the National High Technology Research and Development Program of China (863 Program), with grant number 2007AA01Z453, and partially supported by National Natural Science Foundation of China (NSFC), under grant number 61375119, 61170057 and 60875080.

References

von Ahn, L., Blum, M., Langford, J., 2004. Telling humans and computers apart automatically. *Communications of the ACM*, vol. 47, no. 2, pp. 56-60.

Anayat, S., Ali, A., Ahmad H. F., 2004. Using a probable weight based bayesian approach for spam filtering. in *Proc. IEEE International Multitopic Conference (INMIC'04)*, pp. 340-345.

Androutsopoulos, I., Koutsias, J., Chandrinos, K. V., Spyropoulos C. D., 2000. An experimental comparison of naìve bayesian and keyword-based anti-spam filtering with personal e-mail messages. in *Proc. of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 160-167.

Androutsopoulos, I., Koutsias, J., Chandrinos, K. V., Paliouras, G., Spyropoulos, C. D., 2000. An evaluation of naѷe bayesian anti-spam filtering. in *Proc. European Conference on Machine Learning (ECML'00)*.

Bezerra, G. B., Barra, T. V.*et al.*, 2006. An immunological filter for spam. *Lecture Notes in Computer Science*, vol. 4163, pp. 446-458.

Chang, C.-C., Lin C.-J., 2001. *LIBSVM: a Library for Support Vector Machines*, [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm

Clark, J., Koprinska, I., Poon, J., 2003. A neural network based approach to automated e-mail classification. in *Proc. IEEE International Conference on Web Intelligence (WI'03)*, Halifax, Canada, pp. 702-705.

Dasgupta, D., Attoh-Okine, N., 1997. Immunity based systems: A survey. in *Proc. IEEE International Conference on System, Man, and Cybernetics, Computational Cybernetics and Simulation*, vol. 1, Orlando, FL, USA, pp. 369-374.

De Castro L. N., Timmis, J., 2002. *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer.

Domeniconi, C., Gunopulos, D., 2001. Incremental support vector machine construction. in *Proc. IEEE International Conference on Data Mining (ICDM'01)*, San Jose, CA, USA, pp. 589-592.

Drucker, H., Burges, C. J. C., Kauffman, L., Smola, A., Vapnik, V. N., 1997. Support vector regression machines. in *Advances in Neural Information Processing System (NIPS)*, M. C. Mozer, M. I. Jordan, and T. Petsche, Eds. Cambridge, MA: MIT Press, vol. 9, pp. 155-161.

Drucker, H., Wu, D., Vapnik, V. N., 1999. Support vector machines for spam categorization. *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 1048-1054.

Hoanca, B., 2006. How good are our weapons in the spam wars. *IEEE Technol. Soc. Mag.*, vol. 25, no. 1, pp. 22-30.

Hofmeyr, S. A., Forrest, S., 2000. Architecture for an artificial immune system. *Evolutionary Computation*, vol. 8, no. 4, pp. 443-473.

Ioannidis, J., 2003. Fighting spam by encapsulating policy in email addresses. in *Proc. of Network and Distributed System Security Symposium (NDSS'03)*, San Diego, CA.

Koprinska, I., Poon, J., Clark, J., Chan J., 2007. Learning to classify e-mail. *Information Science*, vol. 177, no. 10, pp. 2167-2187.

Leiba, B., Borenstein, N., 2004. A multifaceted approach to spam reduction. in *Proc. of the first Conference on Email and AntiSpam (CEAS'04)*, Mountain View, CA.

Li, Y., Fang, B., Guo, L., Wang, S., 2006. Research of a novel anti-spam technique based on users's feedback and improved naìve bayesian approach. in *Proc. IEEE International Conference on Networking and Services (ICNS'06)*, Silicon Valley, California, pp. 86-91.

Mitra, P., Murthy, C. A., Pal, S. K., 2000. Data condensation in large databases by incremental learning with support vector machines. in *Proc. IEEE International Conference on Pattern Recognition*, vol. 2, pp. 708-711.

Oda, T., White, T., 2005. Immunity from spam: An analysis of an artificial immune system for junk email detection. *Lecture Notes in Computer Science*, pp. 276-289.

Oda, T., White, T., 2003a. Increasing the accuracy of a spam-detecting artificial immune system. in *Proc. IEEE Congress on Evolutionary Computation (CEC'03)*, vol. 1, Canberra, Australia, pp. 390-396.

Oda T., White, T., 2003b. Developing an immunity to spam. *Lecture Notes in Computer Science*, vol. 2723, pp. 231-242.

Osuna, E., Freund, R., Girosi, F., 1997. An improved training algorithm for support vector machines. in *Proc. IEEE Workshop on Neural Networks for Signal Processing (NNSP'97)*, pp. 276-285.

Platt, J. C., 1998. Sequential minimal optimization: A fast algorithm for training support vector machines. in *Advances in Kernel Method: Support Vector Learning*, B. Scholkopf, C. J. C. Burges, and A. J. Smola, Eds. Cambridge, MA: MIT Press, p. 185¨C208.

Postel, J., 1975. On the junk mail problem. RFC 706, Internet Engineering Task Force.

Rigoutsos, I., Huynh, T., 2004. Chung-kwei: a pattern-discovery-based system for the automatic identification of unsolicited e-mail messages(spam). in *Proc. of the first Conference on Email and AntiSpam (CEAS'04)*, Mountain View, CA.

Sahami, M., Dumais, S., Heckerman, D., Horvitz, E., 1998. A bayesian approach to filtering junk e-mail. in *AAAI Workshop on Learning for Text Categorization*, pp. 55-62.

Secker, A., Freitas, A. A., Timmis, J., 2003. AISEC: An artificial immune system for email classification. in *Proc. IEEE Congress on Evolutionary Computation (CEC'03)*, vol. 1, Canberra, Australia, pp. 131-139.

Shrestha, R., Lin, Y., 2005. Improved bayesian spam filtering based on co-weighted multi-area information. *Lecture Notes in Artificial Intelligence*, vol. 3518, pp. 650-660.

Stuart, I., Cha, S.-H., Tappert, C., 2004. A neural network classifier for junk e-mail. *Lecture Notes in Computer Science*, vol. 3163, pp. 442-450.

Syed, N. A., Liu, H., Sung, K. K., 1999. Incremental learning with support vector machines. in *Proc. International Joint Conference on Artificial Intelligence (IJCAI'99)*, Stockholm, Sweden.

Tan, Y., Xiao, Z. M., 2007. Clonal particle swarm optimization and its applications. in *Proc. IEEE Congress on Evolutionary Computation (CEC'07)*, Singapore, pp. 2303-2309.

Tan, Y., and Wang, J., 2004. A support vector network with hybrid kernel and minimal vapnik-chervonenkis dimension. *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 4, pp. 385-395.

Tan, Y., 2006. Multiple-point bit mutation method of detector generation for snsd model. *Lecture Notes in Computer Science*, vol. 3973, pp. 340-345.

Tan, Y., Xia, Y., Wang, J., 2000. Neural network realization of support vector machines for pattern classification. in *Proc. IEEE International Joint Conference on Neural Networks (IJCNN'00)*, vol. 6, Como, Italy, pp. 411-416.

Vapnik, V., 1995. *The Nature of Statistical Learning Theory*. New York: Springer-Verlag.

Vapnik, V. N., 1982. *Estimation of Dependencies Based on Empirical Data*. New York: Springer-Verlag.

Wang, F., You, Z., Man, L., 2006. Immune-based peer-to-peer model for anti-spam. *Lecture Notes in Bioinformatics*, vol. 4115, pp. 660-671.

Wu, M.-W., Huang, Y., Lu, S.-K., Chen, I.-Y. Kuo, S.-Y., 2005. A multi-faceted approach towards spam-resistible mail. in *Proc. IEEE Pacific Rim International Symposium on Dependable Computing*, pp. 208-218.

Yeh, C.-Y., Wu, C.-H., Doong, S.-H., 2005. Effective spam classification based on meta-heuristics. in *Proc. IEEE International Conference on System, Man, and Cybernetics*, vol. 4, pp. 3872-3877.

Zuchini, M. H., 2003. Aplições de mapas auto-organizáveis emmineração de dados e recuperação de informação. Master's thesis, UNICAMP.