Robust Anomaly Detection Using Support Vector Machines

Wenjie Hu
Department of Applied Science
University of California, Davis
wjhu@ucdavis.edu

Yihua Liao
Department of Computer Science
University of California, Davis
yhliao@ucdavis.edu

V. Rao Vemuri
Department of Applied Science
University of California, Davis
rvemuri@ucdavis.edu

***Abstract*— Using the 1998 DARPA BSM data set collected at MIT's Lincoln Labs to study intrusion detection systems, the performance of robust support vector machines (RSVMs) was compared with that of conventional support vector machines and nearest neighbor classifiers in separating normal usage profiles from intrusive profiles of computer programs. The results indicate the superiority of RSVMs not only in terms of high intrusion detection accuracy and low false positives but also in terms of their generalization ability in the presence of noise and running time.**

***Index Terms*— Intrusion detection, computer security, robust support vector machines, noisy data.**

I. Introduction

THE rapid increase in connectivity and accessibility of computer systems has resulted in frequent opportunities for intrusions and attacks. Anomaly detection and misuse detection are two general approaches to computer intrusion detection. Unlike misuse detection, which generates an alarm when a known attack signature is matched, anomaly detection identifies activities that deviate from the normal behavior of the monitored system (or users) and thus has the potential to detect novel attacks [1].

Over the past decade many anomaly detection techniques, including neural networks (NNs) [2][3], support vector machines (SVMs) [4] and data mining [5], have been proposed to capture system or user's normal usage pattern and classify *new* behavior as either normal or abnormal. These techniques can be further categorized as *generative* or *discriminative* approaches. A generative approach (e.g., [6]) builds a model solely based on normal training examples and evaluates each testing case to see how well it fits the model. A discriminative approach (e.g., [4]), on the other hand, attempts to learn the distinction between the normal and abnormal classes. Both normal and attack examples (attack examples are usually rare) are used in training for discriminative approaches.

Regardless of the approach used, most methods currently in use are based on the assumption that the training examples used by the intrusion detector are untainted and trustable, i.e., the labels of training samples are absolutely right. However, in practice, the data sets obtained from real-world systems (audit trails, application logs, network packet sniffing) hardly satisfies this assumption. First of all, clean data is not always easy to obtain. The training examples may be mislabelled because of the unclear boundary between normal and anomalous behavior. More importantly, there is usually an initial training period for an anomaly detector to learn the monitored system's behavior and samples collected during this training phase are presumed normal. As normal behavior changes over time, new examples are periodically incorporated into the training data and the anomaly detector undergoes frequent retraining. If an attack occurred during the training process, the undesired, intrusive behavior could get established as a part of the anomaly detector's model of normal behavior and thus undermine its ability to detect subsequent occurrences [1] [7] [8]. Machine learning techniques used for anomaly detection, such as neural networks and support vector machines, are sensitive to noise in the training samples. The presence of mislabelled data can result in highly nonlinear decision surface and over-fitting of the training set. This leads to poor generalization ability and classification accuracy.

In this paper, we present a new approach, based on robust support vector machines (RSVMs) [9], to anomaly detection over noisy data. RSVMs effectively address the over-fitting problem introduced by the noise in the training data set. With RSVMs, the incorporation of an averaging technique (in the form of class center) in the standard support vector machines makes the decision surface smoother and controls the amount of regularization automatically (see Appendix for details.). Moreover, the number of support vectors of RSVMs is significantly less compared to those of standard SVMs, which leads to a faster running time of RSVMs. We evaluate this method with the 1998 DARPA BSM data. We compare RSVMs with the standard SVMs and the k-Nearest Neighbor classifier (kNN). Our experiments show the superiority of RSVMs not only in terms of high intrusion detection accuracy and low false positives but also in terms of their their generalization ability in the presence of noise and running time.

The rest of this paper is organized as follows. In Section II we review some related work. Section III describes the method of RSVMs for anomaly detection over DARPAR data set from MIT Lincoln laboratory. Finally, we will summarize and discuss our work in Section IV. A brief mathematical description of SVMs and how they differ from RSVMs is presented in Appendix section.

II. Related Work

The idea of anomaly detection in computer security was proposed in Anderson's paper [10]. Since then, various anomaly

detection approaches have been implemented by establishing statistical models for user [11]-[14], program [15]-[18] or network behavior [4] [5]. The goal of using machine learning techniques for anomaly detection is to develop a generalization capability from limited training data and to be able to correctly classify future data as normal or abnormal. Clean training data is usually assumed.

More recently Eskin and others [8] [19] proposed unsupervised anomaly detection algorithms with unlabelled data, based on the assumption that number of normal instances is significantly larger than the number of anomalies and anomalies appear as outliers in the data.

Forrest et al. introduced the idea of building program profiles with short sequences of system calls issued by running programs for intrusion detection [15]. The underlying premise is that the sequences of system calls during an intrusion are noticeably different from normal sequences of system calls. Lee et al. [16] extended the work of Forrest's group and applied RIPPER, a rule learning program, to the program execution audit data. Warrender et al. [6] introduced a new data modelling method, based on Hidden Markov Model (HMM), and compared it with RIPPER and simple enumeration method. Ghosh and others [17] employed artificial neural network techniques to learn normal sequences of system calls for specific UNIX system programs using the 1998 DARPA BSM data. Liao et al. [18] drew an interesting analogy between a text document and the sequence of all system calls issued by a program. System call frequencies, instead of short sequences of system calls, were used to represent program behavior. Then the k-nearest neighbor classifier was employed to classify new program behavior as normal or intrusive.

This paper extends the work of Liao et al and compares the intrusion detection performance of RSVMs, the standard SVMs and the kNN classifier over the clean training data of program behavior profiles. More importantly, we deliberately form noisy training data and show the robustness of RSVMs.

III. RSVMs WITH DARPA DATA SET

A. An Introduction to RSVMs and SVMs

A brief mathematical description of SVMs and how they differ from RSVMs is presented in the appendix. More elaborate tutorial papers can be found in the literature [20] [21] [22] .

In simple terms a SVM is a perceptron-like neural network and is ideally suitable for binary pattern classification of patterns that are linearly separable. A perceptron-solution, however, is not unique because one can draw a number of possible hyperplanes between the two classes. The main idea of the SVM is to derive a unique separating hyperplane that maximizes the separating margin between the two classes. The feature vectors that lie on the boundary defining this separating margin, in the jargon of linear algebra, are called "support vectors". Classifiers that exploit this property are therefore called support vector machines.

Since the introduction of the original idea, several modifications and improvements are made: hard-margin SVMs for separable cases, soft-margin SVMs for non-separable cases and robust SVMs that exhibit good generalization properties while handling noisy, that is, mis-labelled data. The mathematical differences in these three formulations are pointed out in the appendix.

In order to test the properties of the RSVMs, it is necessary that training data sets are carefully prepared with "noise" explicitly incorporated into them. Using DARPA data as a starting point such noisy data sets are prepared as described next.

B. Pre-processing DARPA Data

The 1998 DARPA Intrusion Detection System Evaluation program provides a large corpus of computer attacks embedded in normal background traffic [23]. The TCPDUMP and BSM (Basic Security Module) audit data were collected on a simulation network that simulated the traffic of an Air Force Local Area Network. It consisted of 7 weeks of training data and 2 weeks of testing data. We used the BSM audit data collected from a victim Solaris machine. The BSM audit logs contain information on system calls produced by programs running on the Solaris machine. The data were labelled with session numbers. Each session corresponds to a TCP/IP connection between two computers. On each simulation day about 500 sessions were recorded by the BSM tool of the Solaris machine.

The pre-processing of the BSM audit data was described in [18]. For the sake of completeness, we outline the procedure here. The names of system calls were extracted for every session from the BSM audit logs. Each session usually consists of one or more processes. A complete ordered list of system calls was generated for each process. A buffer overflow attack session, named `Eject` [24], and the list of system calls of one of its processes are shown in Table I.

The numbers of occurrences of individual system calls during the execution of a process were counted. Liao et al. [18] used two text weighting techniques, namely frequency weighting and *tf ·idf* weighting, to transform the process into a vector. The dimension of a process vector was equal to the number of unique system calls. We adopted the frequency weighting method, which simply assigns the number of occurrences of a system call during the process execution to a vector entry.

C. Clean and Noisy Data

A careful study of the 1998 DARPA BSM data revealed that there are 5 simulation days that are free of attacks during the seven-week training period. We picked 4 days of data out of those 5 days as normal samples for training. There are 606 distinct processes drawn from over 2000 sessions during these 4 days. Our training normal data set contains 300 out of the 606 processes.[1] The other simulation day, the third day of the seventh training week, was chosen for normal testing examples as no attack was launched on this day. It contains 412 sessions and 5285 normal processes (We did not require the testing processes to be distinct in order to count false alarms for one day).

[1]The reason that we chose 300 processes is to mitigate the imbalance between normal and intrusive examples.

TABLE I

(A)AN INTRUSIVE SESSION (EJECT) SAMPLE AND THE CORRESPONDING PROCESSES. (B) THE LIST OF SYSTEM CALLS ASSOCIATED WITH THE PROCESS PWD

Session: **Eject**
telnetd
login
tcsh
quota
cat
mail
cat
gcc
cpp
cc1
as
ld
ejectexploit
pwd

(a)

Process name: **pwd**			
close	close	close	close
open	close	close	execve
open	mmap	open	mmap
mmap	munmap	mmap	close
open	mmap	mmap	munmap
mmap	mmap	close	open
mmap	mmap	munmap	mmap
close	open	mmap	close
open	mmap	mmap	munmap
mmap	close	close	munmap
pathdonf	stat	stat	open
close	open	open	ioctl
lstat	lstat	close	close
close	close	close	exit

(b)

We carefully selected 28 distinct intrusive processes for training (12 of them are the same as the ones used in [18]) from 55 intrusion sessions in the seven-week training data. These 28 processes cover most attack types of the DARPA training data, including the most clearly malicious processes, such as `ejectexploit`, `formatexploit`, `ffbexploit` and so on. In an intrusive session, only a small part of activities are intrusive. For example, the session `Eject` in Table I cosists of 14 processes. We only selected the process `ejectexploit` as an intrusive sample. We used 22 intrusive sessions from the two-week testing data as the intrusive samples for testing purpose.

To demonstrate the robust property of RSVM, we prepared two different training data sets, as illustrated in Table II. For the noisy data set, 16 out of the original 28 intrusive training processes were disguised as normal and incorporated into the 300 truely normal examples, while the testing subset remains the same.

TABLE II

CLEAN AND NOISY DATA SETS

	clean data	noisy data
training	300 normal processes	316 normal processes (16 mislabelled)
	28 intrusive processes	12 intrusive processes
testing	5285 normal processes, 22 intrusion sessions	

D. RSVMs with Clean Data

In intrusion detection, the Receiver Operating Characteristic (ROC) curve is usually used to measure the performance of an intrusion detection method. The ROC curve is a plot of intrusion detection accuracy against the false positive probability. In our experiments, individual processes are classified as normal or intrusive. When a process is categorized as intrusive, the session that the process is associated with is classified as an attack session. The intrusion detection accuracy is then calculated as the rate of detected attacks. The false positive probability, on the other hand, is defined as the rate of misclassified normal processes [18].

Figure 1 (a) shows the performance of the RSVMs, SVMs (RBF Kernel with width $\sigma = 1.0$) and kNN expressed in ROC curves with the clean training data set. RSVMs and SVMs were implemented with the RBF kernel function. The curves were obtained by varying the regularization parameters.

The kNN algorithm is described in [18]. The cosine similarity is used to measure the distance between two processes in the form of vectors. Each testing process is compared to the intrusive training processes first. Whenever there is a perfect match, i.e., the cosine similarity is equal to 1.0, the new process is labelled as intrusive behavior. Otherwise, the testing process is compared with each normal training process. If the similarity score of one normal training process is equal to 1, the new process would be classified as a normal process immediately. Otherwise, We calculate the average similarity value of its k nearest neighbors (with highest similarity scores) and set a threshold. Only when the average similarity value is above the threshold, is the new process considered normal. Here we set k's value to 5 and varied the threshold to get the ROC curve.

As depicted in Figure 1 (a), the attack detection rate of RSVMs was 74.7% with zero false positive rate. The detection rate reached 100% rapidly and the false positive rate remained as low as 3%. SVMs could detect nearly 50% attack sessions with 0% false positive rate and the detection rate reached 100% with a false positive rate of 14.2%. The kNN method gave relatively poor performance. It attained a low attack detection rate (13.6%) at zero false positives. The attack detection rate reached 100% with a false positive rate of 8.6%.

TABLE III

ATTACK DETECTION RATE OF RSVMS, SVMS AND kNN METHODS OVER THE CLEAN TRAINING DATA SET WHEN FALSE POSITIVE RATE IS LESS THAN 1%.

	Attack detection Rate	False Positive Rate
RSVMs	81.8%	$< 1\%$
SVMs	81.8%	$< 1\%$
kNN	63.6%	$< 1\%$

An intrusion detection system is typically aimed at a false positive rate of 1%, as too many false alarms would make

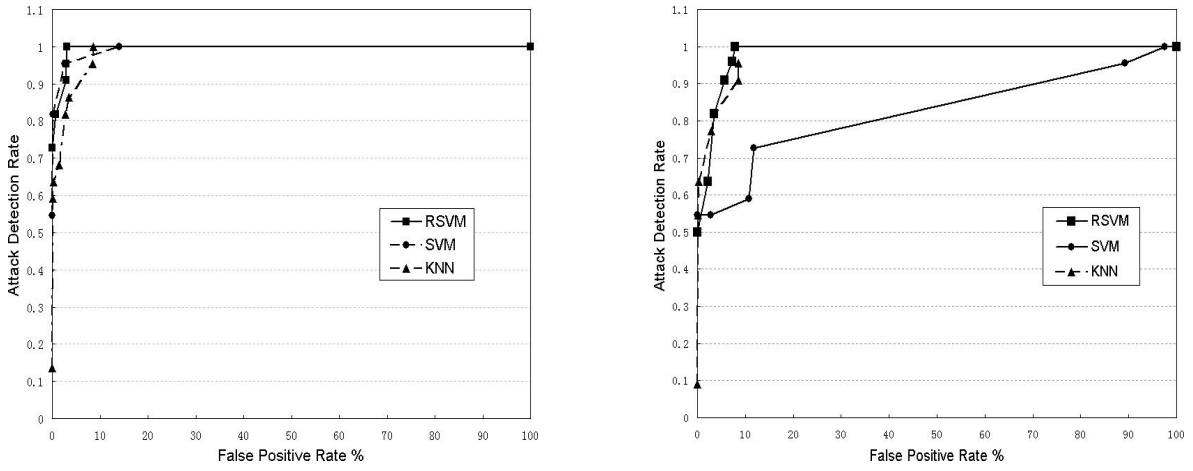

Fig. 1. Performance of RSVMs, SVMs and kNN expressed in ROC curves over (a) clean training data; (b) noisy training data.

the system useless. Table III shows the attack detection rates of RSVMs, SVMs and kNN methods over the clean training data set with the stipulation that the false positive rate does not exceed 1%. Both RSVMs and SVMs showed competitive results (attack detection rate $> 80\%$).

E. RSVMs with Noisy Data

Figure 1 (b) presents the ROC curves of RSVMs, SVMs (RBF Kernel with width $\sigma = 1.0$) and kNN methods over the noisy training data. The attack detection rate of RSVMs was 50% at zero false positives. It reached 100% with a false positive rate of 8%. Although SVM could detect 54% attacks with zero false positive rate, the attack detection rate could not reach 100% until the false positive rate approached 100% (This's useless in practice). Table IV shows the attack detection rates of RSVMs, SVMs and kNN methods over the noisy training data set with the stipulation that the false positive rate does not exceed 3% [2]. The RSVMs showed slight decline of performance in the presence of noise. Meanwhile, SVMs experienced severe performance deterioration due to the noisy training data. These results indicate that RSVMs effectively suppressed the effect introduced by the 16 mislabelled training examples, while the conventional SVMs gave poor generalization ability because of the noise.

TABLE IV
ATTACK DETECTION RATE OF RSVMS, SVMS AND kNN METHODS OVER THE NOISY TRAINING DATA SET WHEN FALSE POSITIVE RATE IS LESS THAN 3%.

	Attack detection Rate	False Positive Rate
RSVMs	81.8%	$< 3\%$
SVMs	54.2%	$< 3\%$
kNN	57.6%	$< 3\%$

The kNN method did not manifest any decline in the false positive rate. This is not surprising considering the fact that by taking the average of the k neighbors nearest to the testing process, it can smooth out the impact of isolated noisy training examples. However, one testing process happened to match one of the 16 mislabelled intrusive processes. It was classified as normal immediately without finding its k nearest neighbors. This testing process was the only process of the testing attack session. Therefore, this attack session could not be detected with our kNN classifier, and the attack detection rate never reached 100%.

[2]Generally, intrusion detection systems are aimed at a false positive rate of 1%. The results showed in Table IV indicate slight deterioration of performance in the presence of noisy training data.

TABLE V
COMPARISON OF THE NUMBER OF SUPPORT VECTORS AND THE EXPERIMENTAL RATIO OF RUNNING TIME OF RSVMS AND SVMS FOR CLEAN AND NOISY TRAINING DATA SETS.

	RSVMs	SVMs	RSVMs /SVMs
Clean Data	30	45	71 %
Noisy Data	15	40	42 %

Besides the classification accuracy, another problem that needs to be addressed is the running time of the intrusion detector. The computational complexity of RSVMs/SVMs is of linear proportion to the number of support vectors as shown in Equation (3) when classifying new examples. As shown in the Appendix, the number of support vectors of RSVMs can be much less than those of the standard SVMs. Therefore, RSVMs require less running time. Table V shows the number of support vectors of RSVMs/SVM with clean/noisy data and the ratio of experimental testing time for the 5285 normal testing processes. In the clean training data case, RSVMs' number of support vectors was one-third less (30 vs 45) and its testing time was 71% of that of SVM. In the noisy data case, the testing time of RSVMs was only 42% of that of SVM due to the greater difference of their decision surfaces. For the kNN classifier, the cost of classifying new examples can be much higher than that of RSVMs and SVMs. This is due to the fact that nearly all computation takes place at classification time.

IV. Conclusion

In this paper, we have proposed a new approach, based on Robust Support Vector Machines, to anomaly detection in computer security. Experiments with the 1998 DARPA BSM data set show that RSVMs can provide good generalization ability and effectively detect intrusions in the presence of noise. The running time of RSVMs can also be significantly reduced as they generate fewer support vectors than the conventional SVMs.

Future work involves quantitatively measuring the robustness of RSVMs over the noisy training data and addressing the fundamental issue of the unbalanced nature between normal and intrusive training examples for discriminative anomaly detection approaches.

V. Acknowledgements

Work reported here is supported in part by the Center for Digital Security of the University of California at Davis under the AFOSR grant F49620-01-1-0327, and by the Institute of Scientific Computing Research of Lawrence Livermore National Laboratory.

Appendix: SVM vs RSVM

The main idea of Support Vector Machines (SVMs) is to derive a hyperplane that maximizes the separating margin between two classes — the positive and the negative [22]. A tutorial introduction to SVM can be found in [20]. The promising property of SVM is that it is an approximate implementation of the Structure Risk Minimization principle based on statistical learning theory rather than the Empirical Risk Minimization method, in which the classification function is derived by minimizing the Mean Square Error over the training data set.

One of the main assumptions of SVM is that all samples in the training set are independently and identically distributed (i.i.d.). However, in practice, the training data are often contaminated with noise. The noisy data makes the validity of this i.i.d. assumption questionable. The standard SVM training algorithm will make the decision surface deviate from the optimal position in the *feature space*. When mapped back to the *input space*, it results in a highly nonlinear decision boundary. Therefore the standard SVM is sensitive to noise, leading to poor generalization ability.

Consider the training samples

$$(\mathbf{x}_1, y_1),\ \cdots,\ (\mathbf{x}_\ell, y_\ell), \quad y_i \in \{-1,\ +1\}, \quad i = 1, ..., \ell \quad (1)$$

where $\{(\mathbf{x}_i, y_i)\}\ i = 1, ..., \ell$ are feature vectors and $y_i \in \{-1,\ +1\}, \quad i = 1, ..., \ell$ are the corresponding labels. Positive class represents normal behavior and negative class represents anomalous behavior. Now the classification problem can be posed as a constrained optimization problem. The *primal* problems of SVM and Robust SVM are shown in Table VI, where $\mathbf{w}$ is the weight vector of the decision hyperplane. The other terms in the table are explained below.

Vapnik [22] proposed the initial idea of SVM for the *separable* case (hard margin SVM) in which the positive and negative samples can be definitely separated by a unique optimal hyperplane with the largest margin. However, this algorithm will find no feasible solution when applied to the *non-separable* case. Cortes and Vapnik [21] extended this idea to the *non-separable* case (soft margin SVM or the so called standard SVM) by introducing positive slack variables $\{\xi_i\}\ i = 1, ..., \ell$. "One must admit some training errors ξ_i to find the best tradeoff between training error and margin by choosing the appropriate constant C associated with slack value" [21] . This error-tolerant property of soft margin SVM makes it very useful in many applications due to its good generalization ability. However, when trained with noisy data, the decision hyperplane might deviate from optimal position (without maximized separating margin) because of the slack term (sum of misclassification errors $\sum \xi_i$) in the objective function of soft margin SVM. This leads to a complicated decision surface. which is known as the over-fitting problem. Song et al [9] proposed the Robust SVM which is aimed at address this over-fitting problem by only minimizing the margin of the weight $\mathbf{w}$ instead of minimizing the margin and the sum of misclassification errors. A new slack term $\lambda D^2(\mathbf{x}_i, \mathbf{x}^*_{y_i})$ is introduced in place of $\{\xi_i\}\ i = 1, ..., \ell$ in the soft margin SVM. Here $\lambda \geq 0$ is a pre-selected regularization parameter measuring the influence of averaged information (distance to the class center), and $D^2(\mathbf{x}_i, \mathbf{x}^*_{y_i})$ represents the normalized distance between data point $\mathbf{x}_i$ and the center of the respective classes, ($\mathbf{x}^*_{y_i}$, $y_i \in \{+1, -1\}$), in the *feature space*. That is,

$$\begin{aligned} D^2(\mathbf{x}_i, \mathbf{x}^*_{y_i}) &= \left|\phi(\mathbf{x}_i) - \phi(\mathbf{x}^*_{y_i})\right|^2 / D^2_{max} \\ &= [(\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_i) - 2\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}^*_{y_i}) \\ &\quad +\phi(\mathbf{x}^*_{y_i}) \cdot \phi(\mathbf{x}^*_{y_i})]/D^2_{max} \\ &= [k(\mathbf{x}_i, \mathbf{x}_i) - 2k(\mathbf{x}_i, \mathbf{x}^*_{y_i}) \\ &\quad +k(\mathbf{x}^*_{y_i}, \mathbf{x}^*_{y_i})]/D^2_{max} \end{aligned} \quad (2)$$

where $\{\phi(\mathbf{x}_i)\},\ i = 1, \cdots, \ell$ denotes a set of nonlinear transformations from the input space to the feature space [3]; $k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$ represents the inner-product kernel function; $D_{max} = max(D(\mathbf{x}_i, \mathbf{x}^*_{y_i}))$ is the maximum distance between the center and training data of the respective classes in the feature space; $k(\mathbf{x}_i, \mathbf{x}^*) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}^*)$ is the kernel function in the feature space. Figure 2 illustrates the meanings of these terms for the case of a linear separating hyperplane in the non-separable case. In this figure, the situations of ξ_i $i = 1, ..., \ell$, the distance $|\Phi(\mathbf{x}) - \Phi(\mathbf{x}^*)|$ of (2) from a point in solid dot class to its class center and threshold value b of decision function (3) are summarized schematically.

The formulas discussed above are in the *primal space*. The solution can be obtained by solving this optimization problem in the *dual space* — the space of Lagrange multipliers α_i, $i = 1, \cdots, \ell$. Table VII shows the dual problems of SVM and Robust SVM, which are Quadratic Programming (QP) optimization problems. The decision function now can be

[3] In class +1, $\phi(\mathbf{x}^*_{y_i}) = \phi(\mathbf{x}^*_{+1}) = \frac{1}{n^+} \sum_{y_j=+1} \phi(\mathbf{x}_j)$, n^+ is the number of data in class +1, in class -1, $\phi(\mathbf{x}^*_{y_i}) = \phi(\mathbf{x}^*_{-1}) = \frac{1}{n^-} \sum_{y_j=-1} \phi(\mathbf{x}_j)$, n^- is the number of data in class -1.

TABLE VI
PRIMAL PROBLEMS OF STANDARD SVM AND ROBUST SVM

	Objective Function	Constrains
Hard margin SVM	$\Phi(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w}$	$y_i f(\mathbf{x}_i) \geq 1$
Soft margin (standard) SVM	$\Phi(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{\ell}\xi_i$	$y_i f(\mathbf{x}_i) \geq 1 - \xi_i$
Robust SVM	$\Phi(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w}$	$y_i f(\mathbf{x}_i) \geq 1 - \lambda D^2(\mathbf{x}_i, \mathbf{x}^*_{y_i})$

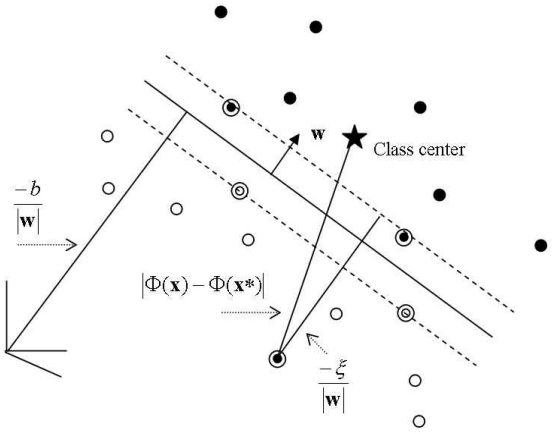

Fig. 2. Linear separating hyperplane for non-separable case. The Star point represents the center of solid points class. The points which are circled are support vectors.

given by the values of α_i, $i = 1, \cdots, \ell$ and computing the sign of

$$f(\mathbf{x}) = \sum_{\texttt{Support Vectors}} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \qquad (3)$$

where b is the threshold value of decision function. The sample data with the corresponding Lagrange multiplies $\alpha_i > 0$ are called *support vectors*.

From Table VII, we can see that the dual problem associated with hard margin SVM is identical to the Robust SVM except the additional term $\gamma_i = 1 - \lambda D^2(\mathbf{x}_i, \mathbf{x}^*_{y_i})$ in the maximization functional $W(\alpha)$. We can justify the slack variable $\lambda D^2(\mathbf{x}_i, \mathbf{x}^*_{y_i})$ by taking it into account as part of the margin. For each data point, the separation margin can be thought of as $1 - \lambda D^2(\mathbf{x}_i, \mathbf{x}^*_{y_i})$ which is more adaptive than the margin $1 - \xi_i$ in soft margin SVM algorithm. Suppose a data point is an outlier that is located on the wrong side and far away from the separable hyperplane. The distance between this point and the center of the class is longer than that of the other normal point in the same class. The augmented term $\lambda D^2(\mathbf{x}_i, \mathbf{x}^*_{y_i})$ is relatively large. Therefore, the inequality constraint shown in Table VI of Robust SVM is satisfied and the coefficient α_i associated with the data point moves toward zero (This result can be derived from the Karush-Kuhn-Tucker condition of the dual form. see reference [9] for details). This data point, therefore, may not become a support vector. Thus the number of support vectors in the RSVM algorithm will be reduced and the decision boundary will be smoother.

REFERENCES

[1] H. Debar, M. Dacier and A. Wespi. "Towards a taxonomy of intrusion detection systems", *Computer Networks*, 31(8), pp 805-822, April 1999.

[2] A. K. Ghosh and A. Schwartzbard. "A Study in Using Neural Networks for Anomaly and Misuse Detection", *Proceedings of the 8th USENIX Security Symposium*, pp 23-36, Washington, D.C. US, 1999.

[3] V. N. P. Dao and V. R. Vemuri. "A Performance Comparison of Different Back Propagation Neural Networks Methods in Computer Network Intrusion Detection", *Differential Equations and Dynamical Systems*, vol 10, No 1&2, pp 201-214, Jan&April, 2002.

[4] S. Mukkamala, G. I. Janoski, and A. H. Sung. "Intrusion Detection Using Support Vector Machines", *Proceedings of the High Performance Computing Symposium - HPC 2002*, pp 178-183, San Diego, April 2002.

[5] W. Lee, S. J. Stolfo and K. Mok. "Data mining in work flow environments: Experiences in intrusion detection", *Proceedings of the 1999 Conference on Knowledge Discovery and Data Mining (KDD-99)*, 1999.

[6] C. Warrender, S. Forrest and B. Pearlmutter. "Detecting Intrusions Using System Calls: Alternative Data Models." *In Proceedings of 1999 IEEE Symposium on Security and Privacy*, pp 133-145, Oakland, 1999.

[7] E. Lundin and E. Johnsson, "Anomaly-based intrusion detection: privacy concern and other problems", *Computer Networks*, vol. 34, pp 623-640, 2000.

[8] E. Eskin. "Anomaly Detection over Noisy Data using Learned Probability Distributions", *In Proceedings of the 2000 International Conference on Machine Learning (ICML-2000)*. Palo Alto, CA: July, 2000.

[9] Qing Song, Wenjie Hu and Wenfan Xie. "Robust Support Vector Machine for Bullet Hole Image Classification", *IEEE Transaction on Systems, Man and Cybernetics Part C. Vol 32. Issue 4*, pp 440-448. Nov. 2002.

[10] J. P. Anderson."Computer Security Threat Monitoring and Surveillance", *Technical Report*, James P. Anderson Co., Fort Washington, Pennsylvania, April 1980.

[11] H. S. Javitz and A. Valdes, *The NIDES Statistical Component: Description and Justification*, Technical Report, Computer Science Laboratory, SRI International, Menlo Park, CA, March 1994.

[12] T. Lane and C. E. Brodley. "An Application of Machine Learning to Anomaly Detection", *Proceedings of the 20th National Information Systems Security Conference*, pp 366-377, Baltimore, MD. Oct. 1997.

[13] D. Endler. "Applying Machine Learning to Solaris Audit Data", *Proceedings of the 1998 Annual Computer Security Application Conference*, pp 268-279. Los Alamitos, CA. Dec 1998.

[14] V. N. P. Dao, V. R. Vemuri and S. J. Templeton. "Profiling Users in the UNIX OS Environment", *International ICSC Conference on Intelligent Systems and Applications*, pp 11-15, University of Wollongong, Australia, Dec. 2000.

[15] S. Forrest, S. A. Hofmeyr, A. Somayaji and T. A. Longstaff. "A Sense of Self for Unix Process", *Proceedings of the 1996 IEEE Symposium on Securiry and Privacy.* pp 120-128. IEEE Computer Society Press, Los Alamitos, CA. 1996.

[16] W. Lee, S. J. Stolfo and P. K. Chan. "Learning Patterns from Unix Process Execution Traces for Intrusion Detection", *Proceedings of AAAI97 workshop on AI Methods in Fraud and Risk Management*, pp 50-57, Rhode Island, July, 1997.

[17] A. K. Ghosh, A. Schwartbard and A. M. Shatz. "Learning Program Behavior Profiles for Intrusion Detection", *Proceedings of 1st USENIX Workshop on Intrusion Detection and Network Monitoring*, pp 51-62, Santa Clara, CA, 1999.

[18] Y. Liao and V. R. Vemuri. "Use of K-Nearest Neighbor Classifier for Intrusion Detection", Computers & Security, 21(5), pp 439-448, October 2002.

[19] E. Eskin, A. Arnold, M, Prerau, L. Portnoy and S. Stolfo. "A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data", In D. Barbara and S. Jajodia (editors), *Applications of Data Mining in Computer Security*, Kluwer, 2002.

TABLE VII

DUAL PROBLEMS OF SVM AND ROBUST SVM (WHERE $\gamma_i = 1 - \lambda D^2(\mathbf{x}_i, \mathbf{x}^*_{y_i})$).

	Objective Function	Constraints
Hard margin SVM	$W(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$	$\sum_{i=1}^{\ell} y_i \alpha_i = 0$ $0 \leq \alpha_i.$
Soft margin (standard) SVM	$W(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$	$\sum_{i=1}^{\ell} y_i \alpha_i = 0$ $0 \leq \alpha_i \leq C.$
Robust SVM	$W(\alpha) = \sum_{i=1}^{\ell} \alpha_i \gamma_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$	$\sum_{i=1}^{\ell} y_i \alpha_i = 0$ $0 \leq \alpha_i.$

[20] C. J. C. Burges. "A Tutorial on Support Vector Machines for Pattern Recognition", *Knowledge Discovery and Data Mining*, 2(2), 1998.

[21] C. Cortes and V. N. Vapnik. "Support Vector Network", *Machine Learning*, Vol. 20. pp273-297, 1995.

[22] V. N. Vapnik. *Statistical Learning Theory.* John Wiley&Sons, Inc. New York, 1998.

[23] R. Lippmann, D. Fried, I. Graf, J. Haines, K. Kendall, D. McClung, D. Webber, S. Webster, D. Wyschograd, R. Cunninghan and M. Zissan. "Evaluating Intrusion Detection Systems: the '1998 DARPA off-line Intrusion Detection Evaluation", *Proceedings of the DARPA Information Survivability Conference and Exposition, IEEE Computer Society Press*, Los Alamitos, CA, 12-26, 2000.

[24] K. Kendall. "A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems", Master's Thesis, Massachusetts Institute of Technology, 1998.
`http://www.ll.mit.edu/IST/ideval/pubs/1998/kkendall_thesis.pdf`