

Network based anomaly intrusion detection system using SVM

J. Arokia Renjit¹ and K.L. Shunmuganathan²

¹ Department of CSE, Jeppiaar Engineering College, TamilNadu-600 119, India

² Department of CSE, RMK Engineering College, TamilNadu-601 206, India
arokiarenjith@gmail.com; Kls_nathan@gmail.com

Abstract

The security and integrity of a computer system is compromised when an intrusion occurs. It becomes impossible for legitimate users to access different network services when network-based attacks purposely occupy or sabotage network resources and services. Our proposed method is a scalable detection method for network based anomalies. We use Support Vector Machines (SVM) for classification. This paper presents a method for enhancing the training time of SVM, particularly when dealing with large data sets, using hierarchical clustering technique. We use the Dynamically Growing Self-Organizing Tree (DGSOT) algorithm for clustering because it has proved to overcome the problems of traditional hierarchical clustering algorithms (e.g., hierarchical agglomerative clustering). Clustering analysis helps to find the boundary points, which are the most qualified data points to train SVM, between any two classes. We present a new approach of combination of SVM and DGSOT, which begins with an initial training set and expands it gradually using the clustering structure produced by the DGSOT algorithm. We show that our proposed variations contribute significantly in improving the training process of SVM with high percentage of detection accuracy.

Keywords: SVM, classification, Intrusion detection, Intrusion detection System, Network Security

Introduction

IDS attempts to detect computer attacks by probing various data records obtained through processes on the same network. These attacks are classified into two categories, host-based attacks (Anderson *et al.*, 1995) and network-based attacks (Marchette *et al.*, 1999). Intrusion detection systems are classified into two groups, namely anomaly detection systems and misuse detection systems. Anomaly detection is the process of identifying malicious traffic based on deviations from recognized normal network traffic patterns (Mccanne *et al.*, 1989). Misuse detection is the capability to identify intrusions based on a known pattern for the malicious behavior (Ilgun *et al.*, 1995). These known patterns are referred to as signatures. Anomaly detection is capable of detecting new attacks. The SVM is one of the most successful and traditional classification algorithms in the data mining area, but its long training time limits its use. This paper proposes a new approach for enhancing the training process of SVM, particularly when dealing with large training data sets. It is based on the combination of SVM and clustering technique. The idea is as follows: SVM computes the maximal margin separating data points; hence, only those patterns closest to the margin will affect the computations of that margin, while other points are discarded without affecting the final result. Those points lying close to the margin are called support vectors. The proposed approach tries to approximate these points by applying clustering analysis. Also, in our approach, the growth of the hierarchical tree is controlled by allowing tree nodes (support vector nodes) closer to the marginal area to grow, while halting distant ones. Therefore, the computations of SVM and further clustering analysis will be reduced considerably. Also, to avoid the cost of computations involved in clustering analysis, SVM training is done on the nodes of the tree after each phase / iteration, in which few nodes are added to the tree. Each

iteration involves growing the hierarchical tree by adding new children to the tree. This could cause a degradation of the accuracy of the resulting classifier. However, we use the support vector set as a priori knowledge to instruct the clustering algorithm to grow support vector nodes and to stop growing non-support vector nodes. By applying this procedure, the detection accuracy of the classifier improves and the size of the training set is kept to a minimum. The main contributions of this work, is to reduce the training time of SVM using clustering analysis. Here, a combination of clustering analysis with SVM training phases is presented.

Related work

Anomaly detection is the process of identifying malicious traffic based on deviations from established normal network traffic patterns. Lee and Stolfo (2000) propose a data mining framework for intrusion detection which is misuse detection. Anomaly detection is capable of detecting new attacks (Upadhyaya *et al.*, 2001). However, new legitimate user behavior can also be falsely identified as an attack, resulting in a false positive prediction. In recent years, there have been several learning-based research efforts or data mining-based research efforts in intrusion detection (Stolfo *et al.*, 2001). Instead of network level data, researchers may also concentrate on user command-level data (Bivens *et al.*, 2002). An SOM is used to create clusters and then graphically display the network data for the user to determine which clusters contained attacks (Girardin & Brodbeck, 1998). SVM is also used for classification in an intrusion detection system. Wang and Stolfo (2003) use "one class SVM" based on single set of examples belonging to a particular class and no negative examples rather than using positive and negative examples. None of these approaches addresses the problem of reduction of the training time of SVM, which prohibits real-time usage of these approaches. Regarding to the training

time of SVM, random sampling has been used to enhance the training of SVM (Tufis *et al.*, 2000). Sub-sampling speeds up a classifier by randomly removing training points. Balacazar *et al.* (2001) use random sampling successfully to train SVM in many applications in the data mining field. However, Yu *et al.* (2003) shows that random sampling could harm the training process of SVM, especially when the probability distribution of training and testing data are different. Yu *et al.* (2003) use the idea of clustering, using BIRCH (Zhang *et al.*, 1996) to fit a huge data set in the computer memory and train SVM on the tree's nodes. The training process of SVM begins at the end of building the hierarchical tree resulting in expensive computations, especially when the data cannot fit in the computer memory or the data is not distributed uniformly. The main objective of the clustering algorithm is to reduce the expensive disk access cost; on the other hand, our main focus is to approximate support vectors iteratively in advance. Furthermore, use of clustering analysis goes in parallel with training SVM, i.e., we need not wait until the end of building the hierarchical tree in order to start the training process.

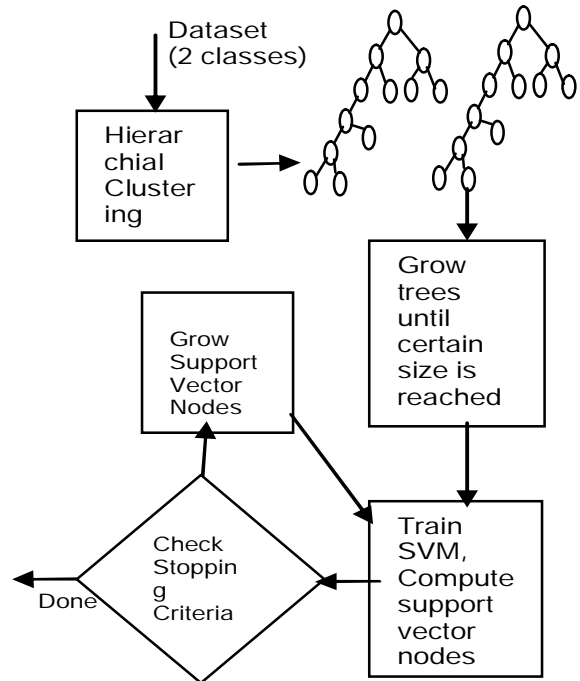
SVM with clustering

SVM are defined as learning systems that uses a hypothesis space of linear functions in a high dimensional feature space, trained with a learning algorithm from optimization theory. SVM are based on the idea of a hyper-plane classification, or linearly separability. Suppose we have N training data points $\{(x_1, y_1), (x_2, y_2), (x_3, y_3), \dots, (x_N, y_N)\}$, where $x_i \in R^d$ and $y_i \in \{+1, -1\}$. Consider a hyper-plane defined by (w, b) , where w is a weight vector and b is a bias. A new object x can be classified with

$$f(x) = \text{sign}(w \cdot x + b) = \text{sign}\left(\sum_i^N \alpha_i y_i (x_i \cdot x) + b\right) \quad (1)$$

The training vectors x_i occur only in the form of a dot product which is a Lagrangian multiplier α_i for each training point. The Lagrangian multiplier values α_i reflect the significance of each data point. When the maximal margin hyper-plane is found, only points that lie closest to the hyper-plane will have the Lagrangian multiplier value $\alpha_i > 0$ and these points are called support vectors. All other points will have $\alpha_i = 0$. Our approach for enhancing the training process of SVM is based on the combination of clustering and SVM to identify the relevant support vectors. In the first step we build a hierarchical clustering tree for each class in the data set using the DGSOT (Dynamically Growing Self Organizing Tree) algorithm. The DGSOT algorithm, top-down clustering, builds the hierarchical tree iteratively. After each iteration, new nodes are added to the tree based on a learning process. After each iteration we train SVM on the nodes of both trees. We use the support vectors of the classifier as prior knowledge for the succeeding iteration in order to control the tree growth. Particularly, support vectors are allowed to grow, while non-support vectors are stopped. This has the impact of adding nodes in the boundary areas

Fig. 1. Flow diagram of clustering trees



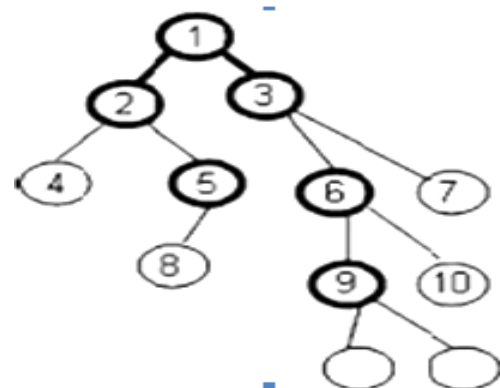
between the two classes, while eliminating distant nodes from the boundaries. Fig. 1 outlines the steps of this approach. First, we generate a hierarchical tree assuming binary classification, for each class in the data set. If a tree exhibits earlier convergence (i.e., fewer number of nodes), one option is to train SVM with these existing nodes.

First if the result is unsatisfactory then the profile and radius thresholds are adjusted. Reducing the threshold values may increase number of clusters and nodes. Second SVM training on the nodes of the trees are done, and the support vectors are computed. Third, on the basis of stopping criteria, we either stop the algorithm or continue growing the hierarchical trees

Stopping criteria

The algorithm can be stopped if the generalization accuracy over a validation set exceeds a specific value (say 98%). For the accuracy estimation purpose, support vectors will be tested with the existing training set based

Fig.2. Growing support vectors



on proximity. Fig.2 shows the growth of one of the hierarchical trees during this approach. In Fig.3, the bold nodes represent the support vector nodes. Notice that nodes 1, 2, 3, 5, 6, and 9 are allowed to grow and expand because they are support vector nodes. Meanwhile, we stop nodes 4, 8, 7, and 10 from growing because they are not support vector nodes.

Fig.3. Updating training set

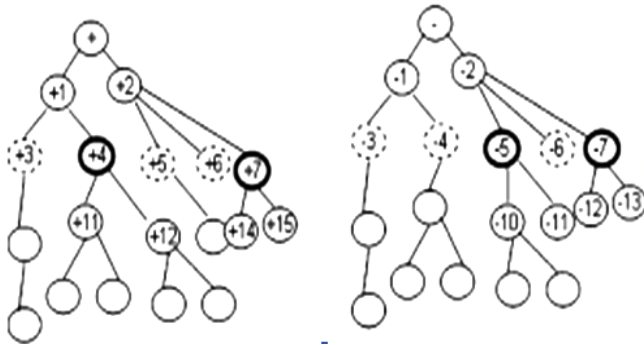
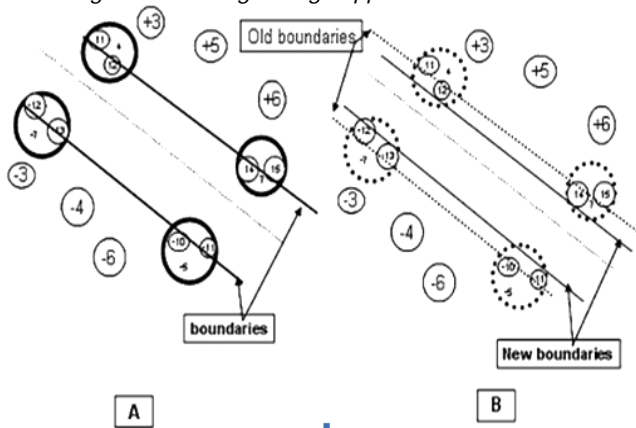


Fig.3 shows an illustrative example of growing the tree up to a certain level and in which we have the training set (+3, +4,+5,+6 + 7,-3,-4,-5,-6,-7). In the above example the dashed nodes represent the clusters' references which are not support vectors. The bold nodes represent the support vector references and hence, we add the children of the support vector nodes +4 and +7 to the training set. The same applies to the support vector nodes -5 and -7. The new data set now is (+3,+11,+12,+5,+6,4,+15,-3,-4,-10,-11,-6,-12,-13). Notice that expanded nodes, such as +4 and +7, are excluded because their children nodes are added to the training set. Part A of Fig. 4 shows the structure of those clusters around the boundaries, and Part B shows the effect of growing the trees on the classifiers.

Fig.4. Result of growing support vector nodes



Hierarchical tree construction

The proposed system uses DGSOT algorithm in which the learning process consists of a series of procedures to distribute all the data to leaves and update the reference

vectors. Each procedure is called a cycle. Each cycle contains series of epochs. Each epoch consists of a presentation of input data with each presentation having two steps. First step is to find the best match node and to update the reference vector. The input data is only compared to the leaf nodes bounded by a sub-tree based on KLD mechanism in order to find the best match node, which is called as the winner.

The leaf node, *c*, which has the minimum distance to the input data *x*, is the best match node (i.e., winner).

$$c : \|x - n_c\| = \min \{ \|x - n_i\| \} \tag{2}$$

After a winner *c* is found, the reference vectors of the winner and its neighborhood will be updated using the following function:

$$\Delta n_i = \phi(t) \times (x - n_i) \tag{3}$$

where $\phi(t)$ is the neighborhood function:

$$\phi(t) = \alpha \times \eta(t), \quad 0 < \eta(t) \leq 1 \tag{4}$$

$\eta(t)$ is the learning rate function, α is the learning constant, and *t* is the time parameter. The convergence of the algorithm depends on a proper choice of α and $\eta(t)$. For a sibling node *j*, the Gaussian neighborhood function $\eta(t)$ will be calculated in the following way:

$$\eta(t) = \exp\left(-\frac{\|h(i, j)\|^2}{2\sigma^2(t)}\right) \tag{5}$$

Where σ gives the width of the Gaussian kernel function, and $h(i, j)$ is the number of hops between the winner *i* and node *j*. The number of hops represents the length of the shortest acyclic path that connects two sibling nodes. The closest sibling of a winner will have a higher value $\eta(t)$ when compared to the most distant sibling node.

Evaluation

The proposed system focuses on network-based anomaly intrusion detection. The proposed algorithm has been applied to a set of standard benchmark data, namely the 1998 DARPA data (Lippmann *et al.*, 1998) that originated from the MIT Lincoln Lab. Network traffic can be obtained using packet-capturing utilities like libpcap (McCanne *et al.*, 1989) or operating system utilities at the system call level like BSM and are then stored as a collection of records in the audit file. Each data point represents either an attack or a normal connection. There are four categories of attacks namely denial-of service (Dos), surveillance (Probe), remote-to-local (R2L) and user-to-root (U2R). Therefore, overall training and testing data will be categorized into any one of these five classes, i.e., a normal and four attack classes. As a training set, we have total 10,23,167 data points; distribution of data among these normal, Dos, U2R, R2L, and Probe classes are as follows: 532415, 482018, 22, 594, and 33152, respectively. Note that the U2R and R2L classes do not have sufficient training examples as compared to others. Distribution of test data is as follows: normal, Dos, U2R, R2L, and Probe classes,

each with 57214, 11527, 17, 1318, and 2229 data points, respectively. Each data point is described by 41 features. Each item of record are represented by a vector. We used a bit vector to represent each nominal value. We use the LIBSVM for SVSM implementation and use the ν -SVM with RBF kernel. We set ν very low (0.001). With regard to setting parameters for the DGSOT, ϵ is set to 0.08, with α sets to 0.01 K, TR and TP are set to 3, 100, and 300, respectively. The parameter settings are purely based on observation.

Results

We have selected randomly 14% from the overall training data set. The results by applying Random selection has been used in many applications to reduce the size of the data set. Random selection technique has been used to reduce the training set of SVM. In this table, diagonal values represent the number that is correctly 55216, 547, 64, 237, and 3342 data points are classified as Normal, Dos, U2R, R2L, and Probe, respectively (Table 1).

Table 1. Classification of data sets

	Normal	DOS	U2R	R2L	Probe	Accuracy (%)	FP (%)	FN (%)
Normal	55216	547	64	237	3342	95	3	5
DOS	316	21145	0	48	211	97	2	3
U2R	4	0	4	9	0	23	100	76
R2L	793	0	408	1024	103	43	24	56
Probe	112	1	0	0	1156	91	60	9

Discussion

Some intrusion experts believe that most novel attacks are variants of known attacks and the signature of known attacks can be sufficient to catch novel variants. Random selection has proved to be a successful technique to save training time. This exactly happens with SVM and DGSOT combination. The accuracy rate of our SVM and DGSOT combination is the best and it is better as compared to pure SVM. Therefore, from the above discussion the superiority of SVM and DGSOT algorithm has been proved over the other methods.

Conclusion

In this paper, we have proposed reduction techniques using clustering analysis to approximate support vectors in order to speed up the training process of SVM. Clustering Trees based on SVM has been implemented and tested, to reduce the training set and approximate support vectors. The proposed algorithm proved to work well and to outperform all other techniques in terms of accuracy, false positive rate, and false negative rate.

References

1. Anderson D, Frivold T and Valdes A (1995) Next-generation intrusion detection expert system (NIDES) a summary. *Technical Report SRI-CSL-95-07. Computer Sci.Laboratory, SRI Intl. Menlo Park.*
2. Bivens A, Palagiri C, Smith R, Szymanski B and EmbrechtsM (2002) Intelligent engineering systems through artificial neural networks. *Proc. ANNIE-2002*, vol. 12, pp. 579-584.
3. Balcazar JL, Dai Y and Watanabe O (2001) A random sampling technique for training support vector machines for primal-form maximal-margin classifiers, algorithmic learning theory. *Proc. 12th Intl. Conf., ALT*, pp: 119.
4. Girardin L and Brodbeck D (1998) A visual approach or monitoring logs. *Proc. 12th System Administration Conf. (LISA 98)*. pp: 299-308.
5. Ilgun K, Kemmerer RA and Porras PA (1995) State transition analysis: A rule-based intrusion detection approach. *IEEE Trans. Software Eng.* 21(3), 181-199.
6. Lee W and Stolfo SJ (2000) A framework for constructing features and models for intrusion detection systems. *ACM Trans. Inform. Syst. Security.* 3(4), 227-261.
7. Marchette D (1999) A statistical method for profiling network traffic. In: *Proc. of the First USENIX Workshop on Intrusion Detection and Network Monitoring*. pp:119-128.
8. McCanne S, Leres C and Jacobson V (1989) Libpcap. available via anonymous ftp at <ftp://ftp.ee.lbl.gov/>
9. Lippmann R, Graf I, Wyschogrod D, Webster SE, Weber DJ and Gorton S (1998) The 1998 DARPA/AFRL off-line intrusion detection evaluation. In: *Proc. of the First Intl. Workshop on Recent Advances in Intrusion Detection (RAID)*.
10. Stolfo SJ, Lee W, Chan PK, Fan W and Eskin E (2001) Data miningbased intrusion detectors: an overview of the Columbia IDS project. *ACM SIGMOD Record.*30(4),5-14.
11. Tufis D, Popescu C and Rosu R (2000) Automatic classification of documents by random sampling. *Proc. Romanian Acad. Ser.*1(2), 117-127.
12. Upadhyaya S, Chinchani R and Kwiat K (2001) An analytical framework for reasoning about intrusions. In: *Proc. IEEE Symposium on Reliable Distributed Systems*. pp:99-108.
13. Wang K and Stolfo SJ (2003) One class training for masquerade detection. In: *Proc. 3rd IEEE Conf. Data Mining Workshop on Data Mining for Computer Security*.
14. Yu H, Yang J and Han J (2003) Classifying large data sets using SVM with hierarchical clusters. In: *Proc. SIGKDD 2003*. pp: 306-315.
15. Zhang T, Ramakrishnan R and Livny M (1996) BIRCH: an efficient data clustering method for very large databases. *Proc. SIGMOD Conf.* pp:103-114.