An Adaptive Intrusion Detection Algorithm Based on Clustering and Kernel-Method

Hansung Lee, Yongwha Chung, and Daihee Park*

Korea Univ. Dept. of Computer & Information Science,
{mohan, ychungy, dhpark}@korea.ac.kr

Abstract. An adaptive intrusion detection algorithm which combines the Adaptive Resonance Theory(ART) with the Concept Vector and the Mecer-Kernel is presented. Compared to the supervised- and the clustering-based Intrusion Detection Systems(IDSs), our algorithm can detect unknown types of intrusions in on-line by generating clusters incrementally.

Keywords: intrusion detection, ART, mercer kernel, concept vector.

1 Introduction

In the traditional *signature-based* IDSs, the rule-base has to be manually revised whenever each new type of attack is discovered. To solve this *manual revision problem*, some of the *machine learning* algorithms have been applied to the IDS[1][2]. However, most of these machine learning approaches are based on *supervised* learning, and have following problems: 1) a large volume of training data should be collected and classified manually; 2) the performance of the IDS depends on the quality of the training data; 3) a training phase with the huge data is computationally expensive and can not be performed in an incremental manner; 4) it is difficult to detect new intrusions which are not trained.

Recently, the *clustering* algorithms based on *unsupervised* learning have been proposed for IDS to overcome these problems[3][4][5]. However, the number of new intrusion types is increased rapidly and the volume of the information is too large. Thus, the general-purpose clustering algorithms used in artificial intelligence need to be modified to satisfy the following IDS requirements: 1) each event data should be processed as soon as it is received and clusters are generated adaptively without fixing the number of clusters; 2) clustering the huge volume of event data needs to be completed in few seconds; 3) the result of clustering needs to be insensitive to the order of input data since the sequence of event data is arbitrary in general.

In this paper, we propose a clustering-based intrusion detection algorithm which can satisfy all the requirements. First, we choose an on-line and incremental clustering algorithm, called *Adaptive Resonance Theory*(ART). In addition to that, we employ both *Concept Vector*[6] and *Mercer-Kernel*[7] to classify a high

* Corresponding author.

W.K. Ng, M. Kitsuregawa, and J. Li (Eds.): PAKDD 2006, LNAI 3918, pp. 603–610, 2006.

dimensional sparse pattern effectively and improve the separability, respectively. These two techniques can improve the detection rates of new intrusions because most of the information source for intrusion detection is high dimensional and very similar to each other. Based on the experimental results, our algorithm can provide superior performance by generating clusters incrementally and subdividing the patterns in detail.

The organization of this paper is as follows. Section 2 explains the data representation and the similarity measure, and Section 3 describes the proposed intrusion detection algorithm. The experimental results are given in Section 4, and conclusions are made in Section 5.

2 Data Representation and Similarity Measure

In this section, we define input dataset and similarity measure in order to evaluate the real world problems; the input patterns are represented by a mixture of variable types. For a given set of n input patterns $\mathbf{x} = \{\underline{x}_i\}_{i=1}^n$, we assume that the input pattern $\underline{x}_i$ consists of k numeric attributes and m symbolic attributes. Let R^k and R^m denote the k-dimensional numeric space and m-dimensional symbolic space, respectively. Then, $\mathbf{x}$ can be represented as follows:

$$\mathbf{x} = \{\underline{x}_i\}_{i=1}^n; \quad \underline{x}_i = \underline{x}_i^R + \underline{x}_i^S; \quad \underline{x}_i^R \in R^k, \, \underline{x}_i^S \in S^m \tag{1}$$

To avoid bias toward some features over other features, we perform L2 normalization on numeric attributes to have unit Euclidean norm.

$$\underline{x}_i^R = \frac{\underline{x}_i^R}{\|\underline{x}_i^R\|}; \quad \|\underline{x}_i\| = \sqrt{\sum_{j=1}^{k} x_{ij}^2} \tag{2}$$

Also, a similarity measure which computes the similarity between objects of mixed variable types is defined as follows: Let m and $\lambda \in [0, 1]$ denote the dimension of the symbolic space and an adjustable parameter in order to weight the attribute types, respectively. Then,

$$S(\underline{x}_i, \underline{x}_j) = \lambda \cdot < \underline{x}_i^R, \underline{x}_j^R > + (1 - \lambda) \cdot \frac{\sum_{l=1}^{m} \delta(x_{il}^S, x_{jl}^S)}{m} \tag{3}$$

where the delta function $\delta(\cdot)$ is defined as follows:

$$\delta(x_{il}^S, x_{jl}^S) = \begin{pmatrix} 1, & if \quad x_{il}^S = x_{jl}^S \\ 0, & otherwise \end{pmatrix} \tag{4}$$

Since numeric attributes are normalized to be unit vectors, the cosine measure is obtained by the inner product of two vectors.

$$< \underline{x}_i^R, \underline{x}_j^R > = \|\underline{x}_i^R\| \cdot \|\underline{x}_j^R\| \cdot COS(\theta(\underline{x}_i^R, \underline{x}_j^R)) = COS(\theta(\underline{x}_i^R, \underline{x}_j^R)) \tag{5}$$

3 Adaptive Intrusion Detection Algorithm

An intrusion detection algorithm proposed in this paper is an "adaptive" algorithm which combines the on-line and incremental clustering algorithm ART with Concept Vector and Mercer-Kernel. By employing the Concept Vector, a weight vector of each cluster is normalized to the mean vector of each clusters. Thus, we need not consider the learning rate parameter in updating the weight vectors and can improve the speed of the execution. Also, we can improve the separability by mapping the input pattern to a feature space with Mercer-Kernel. Details of the proposed algorithm, called *Kernel-ART*, can be described as follows:

Initialization: The number of clusters is set to one initially, and the first input pattern is assigned to its initial weight vector as follows:

$$\underline{w}_1 = \underline{x}_1 = \underline{w}_1^R + \underline{w}_1^S = \underline{x}_1^R + \underline{x}_1^S \tag{6}$$

Then, the matching value, computed by the activation function between the initial weight vector and the first input pattern, is set to one. This ensures that the first input pattern is assigned to the first cluster for any vigilance parameter $\rho \in [0, 1]$.

Activation Function: The basic idea of the Mercer-Kernel is to perform a nonlinear data transformation into some high dimensional dot-product space, called *feature space*, to increase the probability of the linear separability of the patterns within the transformed space[7]. By replacing the inner product in the similarity measure of equation (3) with the RBF kernel function $K(\underline{x}_i, \underline{x}_j) = exp\{-\frac{1}{c}||\underline{x}_i - \underline{x}_j||^2\}$, we can obtain a similarity measure function in the feature space. Then, the activation function is defined by the similarity measure in the feature space as follows:

$$AF(\underline{x}_i, \widehat{\underline{w}}_j) = \lambda \cdot exp\left\{-\frac{1}{c}\|\underline{x}_i^R - \widehat{\underline{w}}_j^R\|^2\right\} + (1-\lambda) \cdot \frac{\sum_{l=1}^{m} \delta(x_{il}^S, w_{jl}^S)}{m} \tag{7}$$

where $\hat{\underline{w}}_j^R = \frac{\underline{w}_j^R}{\|\underline{w}_j^R\|}$ is the Concept Vector of a cluster j. The Concept Vector is the mean vector of the cluster normalized to the unit Euclidean norm. Since the Concept Vectors(i.e., clusters) are localized in the high dimensional sparse space, the clusters can represent the class structure of the dataset. That is, the clusters can represent each types of attacks individually.

Matching Function: If the activation function $AF(\cdot)$ and the matching function $MF(\cdot)$ are chosen as

$$MF(\underline{x}_i, \widehat{\underline{w}}_1) > MF(\underline{x}_i, \widehat{\underline{w}}_2) \Leftrightarrow AF(\underline{x}_i, \widehat{\underline{w}}_1) > AF(\underline{x}_i, \widehat{\underline{w}}_2), \tag{8}$$

then the mismatch reset condition and the template matching process of the original ART can be eliminated for the resonance domain[8]. The most simple

way to define the activation and the matching functions under the condition of equation (8) is to set the activation function to equal to the matching function. By this setting, we can make the algorithm simple and improve the speed of execution.

Resonance Condition: According to the simple setting of the matching function, the resonance unit is selected as follows:

$$AF(\underline{x}_i, \widehat{\underline{w}}_{j^*}) \geq \rho; \quad j^* = arg\, max_{j=1,\ldots,c}\{AF(\underline{x}_i, \widehat{\underline{w}}_j)\} \tag{9}$$

When the best-matching template does not satisfy the vigilance criterion, a new cluster unit can be created and the input pattern is assigned to it. This condition can speed-up the execution time of the algorithm further.

Update Weight Vector: When a cluster j^* is selected by equation (9), the input pattern is assigned to the cluster j^* and the weight vector is updated as follows:

$$\begin{aligned} \underline{w}_{j^*}^{R(t)} &= \underline{w}_{j^*}^{R(t-1)} + \underline{x}_i^R \\ \underline{w}_{j^*}^{S(t)} &= MostFrequentSymbol \end{aligned} \tag{10}$$

Normalize input pattern with L2 norm and Initialize Weights:
$\underline{w}_1 = \underline{x}_1 = \underline{w}_1^R + \underline{w}_1^S = \underline{x}_1^R + \underline{x}_1^S$

While Stopping Condition is false
For each input data
Set activation of all F_2 to zero
Compute Activation Function:
$AF(\underline{x}_i, \widehat{\underline{w}}_j) = \lambda \cdot exp\left\{-\frac{1}{c}\|\underline{x}_i^R - \widehat{\underline{w}}_j^R\|^2\right\} + (1-\lambda) \cdot \frac{\sum_{l=1}^{m} \delta(x_{il}^S, w_{jl}^S)}{m}$

Find j^* with max activation
$j^* = arg\, max_{j=1,\ldots,c}\{AF(\underline{x}_i, \widehat{\underline{w}}_j)\}$

Test for reset:
If $AF(\underline{x}_i, \widehat{\underline{w}}_{j^*}) \geq \rho$ then
$\underline{w}_{j^*}^{R(t)} = \underline{w}_{j^*}^{R(t-1)} + \underline{x}_i^R$
$\underline{w}_{j^*}^{S(t)} = \{MostFrequentSymbol\}$
else
allocation: c = c + 1
$\underline{w}_c^{R(t)} = \underline{w}_c^{R(t)} + \underline{x}_i^R$
$\underline{w}_c^{S(t)} = \underline{w}_c^{S(t)}$

Fig. 1. Outline of the Kernel-ART algorithm

The weight vector of the cluster j^* is defined by sum of input patterns that are assigned to the cluster j^*. Thus, we need not consider the learning rate parameter in updating the weight vectors, and our algorithm is less sensitive to the order of input patterns than that of previous clustering such as Fuzzy ART. This is because, in *Kernel-ART*, the weight vectors memorize the normalized mean vector of the input patterns assigned to each clusters. This intrusion detection algorithm, called *Kernel-ART*, is summarized in Fig. 1.

4 Experimental Results

To evaluate the effectiveness of *Kernel-ART*, KDD CUP 99 data[9] were used for the experiments. In order to make accurate analysis on the experiment result, we used only the Corrected-labeled dataset among KDD CUP 99 data. It was collected through the simulation on the U.S. military network by 1998 DARPA Intrusion Detection Evaluation Program, aiming at obtaining the benchmark dataset in the field of intrusion detection. The size of data is 311,029 and it consists of 9 symbolic attributes and 32 numeric attributes. The data is mainly divided into four types of attack: DOS, R2L, U2R and PROBING. In *Kernel-ART*, ρ is the vigilance parameter which affects the support of clusters. $\lambda \in [0, 1]$ and c denote the weight of the similarity measure function and the RBF kernel-width parameter of *Kernel-ART*, respectively. We set ρ to 0.93, λ to 0.5 and c to 1.

4.1 Comparison with Other Intrusion Detection Algorithms

Because many research results of intrusion detection have been reported recently, we compare our performance with those supervised and unsupervised(clustering) learning algorithms. Table 1 shows the classification capability of each research for normal data and four types of attack. The *Kernel-ART* proposed in this paper can provide good classification capability as a whole, as shown in Table 1. Most of previous methods except IDBGC[5] show considerable inferior performance only at the classification capability as to R2L and U2R. Note that R2L and U2R are host-based attacks which exploit vulnerabilities of the operating systems, not of the network protocol. Therefore, these are very similar to the "normal" data

Table 1. Comparison with Other Intrusion Detection Algorithms

		Supervised Learning			Unsupervised Learning	
		Bernhard [10]	KayAcik [11]	Ambwani [12]	IDBGC[5] Sampled	**Proposed Kernel-ART**
Normal		99.5%	95.4%	99.6%	-	**97.1%**
Attack	DOS	97.1%	95.1%	96.8%	56.0%	**99.9%**
	U2R	13.2%	10.0%	4.2%	78.0%	**61.4%**
	R2L	8.4%	9.9%	5.3%	66.0%	**33.1%**
	PROBING	83.3%	64.3%	75.0%	44.0%	**95.5%**

in the KDD CUP 99 data collected from network packets. However, our method can provide superior performance in separating these two patterns. It can be said, therefore, that the strategy of Kernel-Method employed in this paper, that is to improve the separability(see Eq. 7), is turned out to be very efficient. Note that results reported in [5] were average detection ratios with small number of "sampled" instance from KDD CUP 99 data, which are sampled with similar number of each attack type. The comparisons indicates that our method is not only comparable to [5] in general but also outperformed in DOS and PROBING, in particular.

4.2 Clustering Results of Each Subsidiary Types of Attack

To show the efficiency of *Kernel-ART*, we summarized the clustering results of each subsidiary types of attack in Table 2. In general, reasonable detection ratio with large number of instance. Depending on type of attack, the size of data available for the testing is quite different. The size of attack instance that pertained to U2R and R2L is much smaller than that of other types of attack. Therefore, some of the attacks in those two classes show low detection ratio. In case of PROBING, Saint is a network probing tool modeled after Satan. So, saint and Satan are clustered together. According to the experimental results of Table 2, detailed separation capability of *Kernel-ART* is relatively good. As in section 4.1, this result matches well with the strategy of Kernel-Method employed in this paper.

Table 2. Experimental Results of each Subsidiary Types of Attack

Type	Attacks	No. of instance	Detection Ratio	Attacks	No. of instance	Detection Ratio
DOS	land	9	100.0%	mailbomb	5000	100.0%
	processtable	759	100.0%	smurf	164091	100.0%
	neptune	58001	99.8%	apache2	794	99.6%
	back	1098	99.5%	pod	87	88.5%
	teardrop	12	0.0%	udpstorm	2	0.0%
U2R	multihop	18	38.9%	buffer overflow	22	27.3%
	ps	16	25.0%	perl	2	0.0%
	rootkit	13	0.0%	loadmodule	2	0.0%
	sqlattack	2	0.0%	xterm	13	0.0%
R2L	imap	1	100.0%	guess passwd	4367	99.6%
	httptunnel	158	65.8%	warezmaster	1602	54.8%
	xsnoop	4	50.0%	named	17	47.1%
	ftp write	3	33.3%	snmpgetattack	7741	0.5%
	snmpguess	2460	0.1%	phf	2	0.0%
	worm	2	0.0%	xlock	9	0.0%
	sendmail	17	0.0%			
PROBING	nmap	84	100.0%	mscan	1053	96.6%
	satan	1633	95.8%	portsweep	354	93.5%
	ipsweep	306	83.0%	saint	736	14.3%

Table 3. Input Parameters of Experiments: α and β are learning rates of Fuzzy Art, and ρ is the vigilance parameter. λ and c denote the weight of the similarity measure function and the RBF kernel-width parameter of Kernel-ART, respectively.

K-means	# of cluster = 39, repeat 30 experiments, using min-max normalization
Fuzzy ART	$\alpha = 0.00001$, $\beta = 1.0$, varying ρ from 0.35 to 0.95
Kernel-ART	$\lambda = 0.5$, c from 0.01 to 0.1, varying ρ from 0.35 to 0.95

Table 4. Comparison with Other Clustering Algorithms

		K-means	Fuzzy ART	**Proposed Kernel-ART**
Normal		75.6%	82.4%	**96.6%**
Attack	DOS	64.2%	93.8%	**93.2%**
	U2R	81.8%	84.1%	**87.5%**
	R2L	33.0%	62.3%	**73.9%**
	PROBING	96.6%	99.4%	**100.0%**

4.3 Comparison with Other Clustering Algorithms

To evaluate the clustering characteristics of *Kernel-ART*, we compared our method with typical clustering algorithms such as K-means and Fuzzy ART. Among the labeled 311,029 data instances, we sampled 880 data instances such as 176 normal instances, 176 DOS attacks, 176 R2L attacks, 176 U2R attacks, and 176 PROBING instances. The conditions of this experiment are summarized in Table 3, and the results of the experiment are shown in Table 4. These results show that *Kernel-ART* can provide better performance in classifying both "normal" and "attack" than the typical clustering methods.

5 Conclusions

In this paper, we have presented a robust and efficient intrusion detection algorithm which can detect various types of unknown intrusions in on-line by generating clusters incrementally. The Concept Vector and the Mercer-Kernel can classify a high dimensional sparse pattern effectively and improve the separability. Based on the experimental results, the proposed *Kernel-ART* can classify individual attacks more accurately than the previous methods. This classifying information can be exploited further for developing different responses to different attacks and several intrusion prevention strategies. Because our algorithm has no training phase and does not require periodical renewals of discovered attacks, the cost of system maintenance can also be reduced significantly. We believe our algorithm is very practical and can be employed in future IDSs because of its computational efficiency and ability in detecting new intrusions.

References

1. Lee, W., Stolfo, S., and Mok, K.: 'A Data Mining Framework for Building Intrusion Detection Models', Proceedings of the 1999 IEEE Symposium on Security and Privacy, pp. 120-132, 1999.
2. Hu, W., Liao, Y., and Vemuri, V. : 'Robust Support Vector Machines for Anomaly Detection in Computer Security', Proceedings of the International Conference on Machine Learning and Applications, pp.168-174, 2003.
3. Portnoy, L., Eskin, E., and Stolfo, S.: 'Intrusion Detection with Unlabeled Data using Clustering', Proceedings of the ACM Workshop on Data Mining Applied to Security, 2001.
4. Ye, N. and Li, X.: 'A Scalable Clustering Technique for Intrusion Signature Recognition', Proceedings of the IEEE Man, Systems and Cybernetics Information Assurance Workshop, 2001.
5. Liu, Y., Chen, K., Liao, X., and Zhang, W.,: 'A Genetic Clustering Method for Intrusion Detection', Pattern Recognition, Vol. 37, Issue 5, pp. 927-942. 2004.
6. Dhillon, I. and Modha, D.: 'Concept Decomposition for Large Sparse Text Data using Clustering', Technical Report RJ 10147(95022), IBM Almaden Research Center, 1999.
7. Girolami, M.: 'Mercer Kernel-based Clustering in Feature Space', IEEE Transaction on Neural Networks, Vol. 13, Issue 3, pp. 780-784, 2002.
8. Baraldi, A. and Chang, E.: 'Simplified ART: A New Class of ART Algorithms', International Computer Science Institute, TR 98-004, 1998.
9. KDD Cup 1999 Data, Available in http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.
10. Results of the KDD 1999 Classifier Learning Contest, Available in http://www-cse.ucsd.edu/users/elkan/clresults.html.
11. Kayacik, H., Zincir-Heywood, A., and Heywood, M.: 'On the capability of an SOM based intrusion detection system', Proceedings of the International Joint Conference on Neural Networks, Vol. 3, pp. 1808-1813, 2003.
12. Ambwani, T.: 'Multi class support vector machine implementation to intrusion detection', Proceedings of the International Joint Conference on Neural Networks, Vol. 3, pp. 2300-2305, 2003.