A Grid-Based Clustering for Low-Overhead Anomaly Intrusion Detection

Yang Zhong
Graduate School of Information Science
Nagoya University
Furo-cho, Chikusa-ku
Nagoya, 464-8601 Japan
zhongyang@net.itc.nagoya-u.ac.jp

Hirohumi Yamaki
Information Technology Center
Nagoya University
Furo-cho, Chikusa-ku
Nagoya, 464-8601 Japan
yamaki@itc.nagoya-u.ac.jp

Hiroki Takakura
Information Technology Center
Nagoya University
Furo-cho, Chikusa-ku
Nagoya, 464-8601 Japan
takakura@itc.nagoya-u.ac.jp

***Abstract*—To defend a network system from security risks, intrusion detection systems (IDSs) have been playing an important role in recent years. There are two types of detection algorithms of IDSs: misuse detection and anomaly detection. Because misuse detection is based on a signature which is created from the features of attack traffic by security experts, it can achieve accurate and stable detection. However, its weakness is the difficulty of detecting new attacks (i.e., 0-day attack), and the cost of maintaining the latest signature version. Thinking of the increase of the skillful intrusion, e.g., intrusion showing similar access behavior to normal, misuse detection cannot handle these critical attacks, which results in a large number of false alarms. To cope with these problems, we present a clustering algorithm based on an unsupervised anomaly detection. We evaluated our system using Kyoto2006+ data set and KDD Cup 1999 data set. Evaluation results show that our approach achieved a higher detection rate in the region of very low false positive rate and real-time preprocessing capability.**

I. Introduction

To protect our privacy from network threats, there are various defensive measures like firewalls, antivirus softwares and intrusion detection systems (IDSs). Two different approaches are taken to build IDSs: *misuse detection* and *anomaly detection*. Misuse detection uses a signature which describes the features of attack traffic for deciding whether traffic is attack or not. This indicates that, if unknown attacks or attacks which do not match any signature were carried out, these cannot be detected. In addition, due to the recent attacks, security operators are required to tune set of signatures frequently. Although this work is mandatory to maintain high detection performance, it is very time consuming.

Because of these weak points of misuse detection, much research on anomaly detection using unsupervised learning techniques has been carried out over the past few years. Many approaches use clustering or one-class support vector machine (SVM) [9]. Unsupervised anomaly detection uses unlabeled data for training, and finds a surface that can separate attack and normal. The benefit of this approach is that it can detect new attacks which training data do not contain, and is not affected by wrongly labeled data. However, anomaly detection still has two problems: a low detection rate and a high false positive rate.

In this paper, we present a clustering algorithm for unsupervised anomaly detection. Our algorithm is based on K-means clustering, which is a typical clustering algorithm. The difference between our method and K-means is the generation strategy of the initial cluster centers which is sensitive to effect clustering results.

We evaluated the proposed method using Kyoto 2006+ data set, which consists of network connections collected from real network environment, and KDD Cup 1999 data set which is widely used in the security field. Our experimental result shows that under low false positive rate, e.g., 1%, the proposed method achieves better detection rate than existing methods.

The rest of the paper is organized as follows: In Sect.II, we present the K-means clustering algorithm. In Sect.III, we present our clustering algorithm in detail. In Sect.IV, we describe details of our experiment results and analysis. Finally, in Sect.V, we describe our conclusion from the experiment and suggestions for future study.

II. K-means Clustering Algorithm

K-means [4] is one of the non-hierarchy clustering algorithm which partitions a set of data to k clusters. The time complexity of this method is $\mathrm{O}(kn)$. n denotes the size of data (i.e., number of instances). if $k \ll n$, this clustering method is linear time complexity and faster than other hierarchy algorithms. Due to a mass of traffic data for clustering, K-means based algorithms have been widely deployed in IDS research field. This clustering algorithm takes the following steps:

- Initialization : Randomly choose k instances from dataset and regard them as initial cluster centers.
- Assignment : Assign each instance to the closest center.
- Updating : Replace every center with the mean of its members.
- Iteration : Repeat Assignment and Updating until terminate condition is satisfied.

It is known that, the K-means algorithm is sensitive to the initial centers [5]. In addition, we cannot predict the most appropriate k for clustering beforehand. These initialization problems lead to the result of clustering falling in the local optimum, not in the global optimum. We propose a method to overcome the initialization problems of K-means for intrusion detection.

III. Proposed Algorithm

As with other algorithms, our method mainly consists of two phases: training and testing. The training phase contains the following process:

1) Initialization (Sect.III-A)
2) Assignment and Updating (Sect.III-B)
3) Iteration (Sect.III-C)
4) Labeling (Sect.III-D)

In initialization, we use an algorithm using cells to generate initial cluster centers. A cell is a hypercube subspace of a whole feature space. After that, we use K-means algorithm for clustering. Finally, we label each cluster to detect attacks. In the testing phase, we assign instances to labeled cluster centers in the same way as training. The testing instances are classified by the assigned cluster's label which indicates attack or normal.

In this paper, we define the following symbols:

- k', k : the number of cells and clusters.
- $\boldsymbol{x} = (x_1, x_2, ..., x_d)$: a vector (i.e., instance) in real d-dimension space.
- $\|x\|$: Euclidean distance of $\boldsymbol{x}$.
- $X = \{\boldsymbol{x}_i | 1 \leq i \leq n\}$: the universal set of instances.
- $G = \{g_l | 1 \leq l \leq k'\}$: the universal set of whole cells.
- g : especially initial cell.
- $n(g_l)$, $d(g_l)$: number of instances in cell and density of cell g_l.
- $e(g_{l(j)})$: edge length of j-th dimension of cell g_l.
- $C = \{c_l | 1 \leq l \leq k\}$: the universal set of clusters.
- $\boldsymbol{c(c_l)}$, $n(c_l)$: center (i.e., real d-dimension vector) and number of instances of cluster c_l.

A. Generating Initial Cluster Centers-Based Cells

A cell is a hypercube subspace of a whole feature space, and splitting divides a cell evenly into two cells. After splitting, we maintain the relationship between the original cell and the two new ones by a tree structure (i.e., child cells are linked to their parent). There are 6 steps in our initialization algorithm.

Step1 In the beginning, we create a cell which contains all of the instances (i.e., initial cell). Each edge length of an initial cell is calculated by the difference between the maximum and minimum value of each dimension of whole instances. Here we define the density of the initial cell $d(g)$. Density function $d()$ is defined as follows:

$$d(g_l) = \frac{n(g_l)}{\prod_{j=1}^{d} e(g_{l(j)})} \tag{1}$$

The product of all edge length of a cell represents the volume of the cell.

Step2 Orthogonally to each dimension, we split a cell evenly as shown in Figs.1 and 2. This paper calls this step *pseudo-splitting*. We have d ways of pseudo-splitting, so that $2 \times d$ cells are created from one original cell. From Figs.1 and 2, we can see 2×2 cells are created from one original cell by pseudo-splitting.

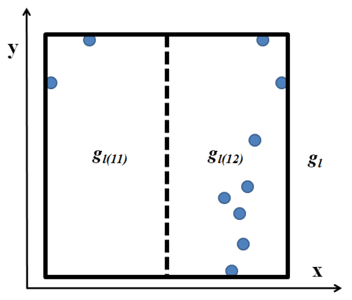

Fig. 1. Split orthogonally to x axis

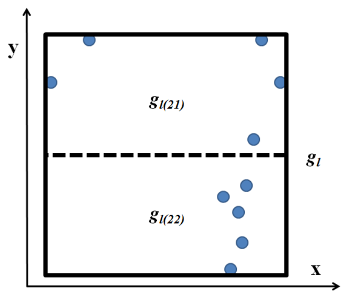

Fig. 2. Split orthogonally to y axis

Step3 In this step, we select only one split from the results of Step 2. $g_{l(j1)}$, $g_{l(j2)}$: When g_l is divided orthogonally to j-th axis as shown in Figs.1 and 2, equi-sized sub cells $g_{l(j1)}$ and $g_{l(j2)}$ are obtained. If the distribution of instances in a cell is not even, this cell should be divided. For effective splitting, instances which have similar distribution should be gathered into a cell. In order to find out the most suitable splitting among all splitting of pseudo-splitting, we define the following function.

$$\sigma(g_{l(j)}) = \frac{1}{2} \cdot \sum_{m=1}^{2} |n(g_{l(jm)}) - \frac{n(g_l)}{2}| \tag{2}$$

Since $\frac{n(g_l)}{2}$ means the half number of original cell g_l, a large value of $\sigma(n(g_{l(j)}))$ represents that there exists uneven distribution in g_l and that pseudo-splitting to j-th axis can divide the cell effectively. For example, in Figs.1 and 2, there are 10 instances in the original cell g_l. Therefore, $\sigma(n(g_{l(1)})) = \frac{|2-5|+|8-5|}{2} = 3$ (Fig.1) and $\sigma(n(g_{l(2)})) = \frac{|5-5|+|5-5|}{2} = 0$ (Fig.2). However, if most instances concentrate around a point in a cell, the splitting operation cannot change $\sigma(n(g_{l(j)}))$ drastically. This result indicates that the same split is selected recursively, and one edge of the cell becomes too short. When we use clustering based distance, a cluster is regarded as a hypersphere. Therefore, the shape of the cell should be near a hypercube which is similar to the hypersphere, we introduce a parameter ω_{lj} which denotes the degree compatibility of split and by which both $\sigma(n(g_{l(j)}))$ and edge length of a cell are taken into account. $\omega_{l(j)}$ is defined as follows:

$$\omega_{l(j)} = \sigma(n(g_{l(j)})) \cdot e(g_{l(j)}) \tag{3}$$

$\omega_{l(j)}$ means that if $\sigma(n(g_{l(j)}))$ is larger or an edge of a cell is longer, this dimension is easier to split. The selected split dimension is j which equals to $max(\omega_{l(j)})$. For example, if we consider each length of edge as 1 in Figs.1 and 2, $\omega_{l(1)} = 2 \cdot 1$ and

$\omega_{l(2)} = 0 \cdot 1$. We decide that the split in Fig.1 is more appropriate than Fig.2. After selecting the split, we only create 2 cells as child cells from the parent cell g_l.

Step4 The splitting operation continues until one of the following conditions is satisfied:

$$\frac{n(g_l)}{n} \leq \alpha \tag{4}$$

$$\frac{d(g_l)}{d(g)} \geq \beta \tag{5}$$

The first condition means that if the number of instances in a cell is too small, this cell is regarded as sparse space and further splitting is not required. The second means that if the density of cell is too high, this cell is regarded as dense space which also does not require to split. Fig.3 illustrates how the splitting proceeds, and Fig.4 shows an example of the result of splitting applied to actual traffic data.

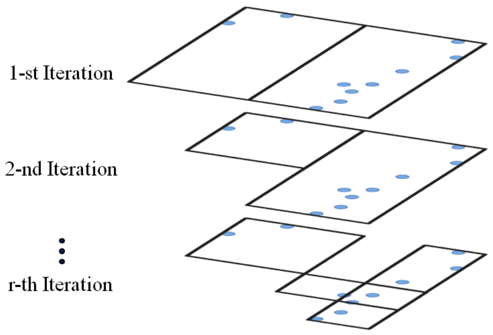

Fig. 3. Iteration of splitting

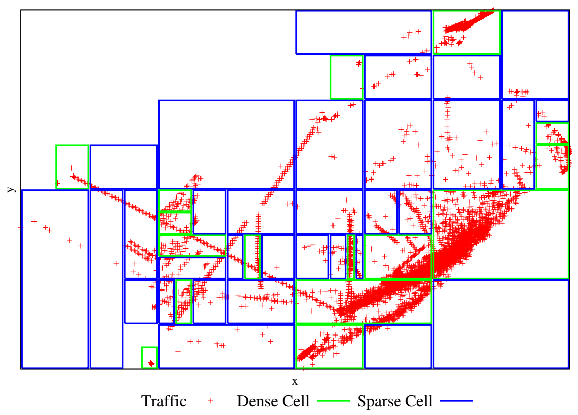

Fig. 4. Example of splitting to actual traffic data

Step5 After the splitting iteration (i.e., Step 4), cells are connected according to the following rules. If both g_p and g_q have higher density than $d(g)$, have same parent, and they satisfy the following condition, they are connected into one cell.

$$|d(g_p) - d(g_q)| < \sqrt{\frac{1}{k'} \cdot \sum_{l=1}^{k'} (d(g_l) - d(g))^2} \tag{6}$$

This condition is that if the difference of density between two neighboring cells is less than the deviation of all cells, we consider the two cells' density are similar, and these can be merged as one. To find out which space is dense or sparse clearly, we need to split cells finely. However, there is a problem that creating too many small cells by fine splitting is not good for clustering (e.g., each cell has only one instance). Due to this problem, we use merging process to prevent over-splitting such as the weakness of top-down clustering [2]. For example, considering the two cells that both density exceed $d(g)$, have same parent, and have different distribution that appears in Fig.5. The left side cell should be divided to two cells to eliminate the sparse domain from the cell, while right side cell should not. Because of the same density, two cells will be split shown as Fig.6 when β set larger. In Fig.6, we count each cell from left to right as 1,2,3 and 4, and assume each volume of cell is 1 for explain. Then, $d(g) = \frac{12}{4} = 3$, $|d(g_1) - d(g_2)| = 4$, $|d(g_3) - d(g_4)| = 0$, and the right side of (6) is

$$\sqrt{\frac{1}{4} \cdot \{(1-3)^2 + (5-3)^2 + (3-3)^2 + (3-3)^2\}} = \sqrt{2}$$

Therefore, we merge only right side cells (g_3 and g_4), and it becomes the state shown as Fig.7.

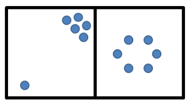

Fig. 5. Two cells with same density

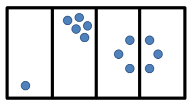

Fig. 6. Not applied merging after splitting

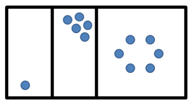

Fig. 7. Applied merging after splitting

Step6 Lastly, we calculate the mean vector of instances in each cell as initial centers.

$$\boldsymbol{c}(c_l) = \frac{1}{n(g_l)} \cdot \sum_{\boldsymbol{x} \in g_l} \boldsymbol{x} \tag{7}$$

B. Assignment and Updating

In this process, we assign instances to the closest cluster center. After the assignment, we update each cluster center with the mean vector of its new members.

$$\boldsymbol{c}(c_l) \leftarrow \frac{1}{n(c_l)} \cdot \sum_{\boldsymbol{x} \in c_l} \boldsymbol{x} \tag{8}$$

C. Convergence Criterion

In this process, we use the change of the number of instances in each cluster as convergence criterion. We define a new symbol $n(c_l)^*_r$ to denote the sorted $n(c_l)$ at r-th time assignment (i.e., $n(c_1)^*_r \leq n(c_2)^*_r \leq ... \leq n(c_k)^*_r$). The convergence criterion is denoted as follows:

$$\frac{1}{n} \cdot \sum_{l=1}^{k} |n(c_l)^*_r - n(c_l)^*_{(r-1)}| < t \tag{9}$$

$|n(c_l)^*_r - n(c_l)^*_{(r-1)}|$ means the difference between the number of instances in sorted cells at r-th time assignment and at $(r-1)$-th time assignment. Parameter t denotes the percentage of unstable instances.

D. Labeling

After the clustering process is finished, we label the final generated k cluster centers as either normal or attack. First, we define clusters which contain large numbers of instances as normal clusters. This condition is denoted as follows:

$$\frac{n(c_l)}{n} > \epsilon \tag{10}$$

To determine attack clusters, we propose two models of attack pattern':

- Low density attack : Little amount of traffic to hide its behavior, such as an intrusion, targeted attack, advanced persistent threat attack.
- High density attack : Using large traffic to exhaust computer resources, such as DoS, scan, etc.

Second, if the number of instances in a cluster is too small, such cluster is regarded as low density attack. Third, if the most instances exist within the small portion of a cell, e.g., one point, we regarded the cluster as high density attack. Considering these features of attacks, the two attack conditions are defined as follows:

$$\frac{n(c_l)}{n} < \gamma \tag{11}$$

$$\frac{n(c_l)}{\sum_{\boldsymbol{x} \in c_l} \|\boldsymbol{x} - \boldsymbol{c(c_l)}\|} > \delta \tag{12}$$

Finally, we regard the clusters which do not satisfy all of the above 3 conditions as normal clusters. Note that the labeling order (normal $\rightarrow$ low density attack $\rightarrow$ high density attack $\rightarrow$ remains) has meanings. If a cluster is labeled once, it does not labeled again.

E. Detection of attack

In the detection process, we assign test instances to the closest cluster center. In other words, if an instance is assigned to an attack cluster, this instance would be an attack. Otherwise it is normal. We use two parameters to measure the system performance: the detection rate and the false positive rate.

- Detection Rate :

$$\frac{\#\ of\ attacks\ that\ are\ labeled\ as\ an\ attack}{\#\ of\ total\ attacks}$$

- False Positive Rate :

$$\frac{\#\ of\ normal\ instances\ that\ are\ labeled\ as\ an\ attack}{\#\ of\ total\ normal\ instances}$$

High detection rate and low false positive rate are suitable for efficient detection performance.

IV. Evaluation

A. Data used in the Experiment

Kyoto 2006+ data set[1] [12] contains the network connections of Kyoto University honeypots [11] from November, 2006. Honeypots are deployed inside and outside Kyoto University: 1 class A and 4 class B networks. There are different types of real and virtual machines in honeypots such as Windows (e.g., XP with full patch, XP with no patch or Vista), Linux/Unix (e.g., Solaris 8 or Max OS X), network printer, home appliance (e.g., TV set or HDD recorder) and darknet and so on. In general, all traffic to/from honeypots can be regarded as attacks. To collect normal traffic for data sets, Kyoto2006+ deploys a server in the same network which has a DNS service and a mail service. Although several attacks are observed at the server, their ratio is negligible compared with the amount of DNS and mail traffic. There are 14 conventional features (e.g., duration, service type, source and destinations bytes, etc), and 10 additional features (e.g., IDS alert, Antivirus alert, Shellcode ID, etc). The 14 conventional features are extracted from 41 features of KDD Cup 1999 [1] data set as significant features.

Training Data : We use traffic data of November 2nd, 2007, as the training data. There are 59,800 instances in this data and the ratio of attack data was 44.95%.

Testing Data : We use traffic data of 2008 as testing data. For example, there are 108,121 instances in data of January 30th, 2008, and the ratio of attack data was 30.5%.

B. Preprocess

We use all service type (e.g., smtp, ssh, ftp, etc.) traffic and 12 continuous features (exclude service type and flag). In a general environment, the ratio of attack would be very small compared to normal. Thus, we randomly filtered out attack traffic to adjust the ratio of attack to 1% in the training data (In the experiment, 33,249 instances remained). Note that the attack ratio (i.e., 1%) is quite reasonable, and the majority of the existing research, they used the same attack ratio. In normalization, we rescale each feature space to $[0, 1]$ space. After normalization, we apply Principal Component Analysis (PCA) [15] for dimensionality reduction. There is important reason for applying PCA apart from avoiding the curse of dimension and reducing time consumption. We consider that using PCA could maintain the robustness of the data set, because eigen vector of each dimension which has outliers would be small. Therefore the effect of these outliers would be reduced. The outliers also affect system performance adversely because when we use grids to partitions instances using groups, if almost all instances gather near the same point,

[1]http://www.takakura.com/Kyoto_data/

the grids could not find the dense space and sparse space correctly, and only create one cell which contained almost all instances and one cell which contained outliers. Therefore, we applied PCA for robustness of the data set, and maintaining correct grids creation. When we reduce the dimension, we kept the contribution ratio of rescaled training data to 99% (remained 7 dimensions in the experiment). Testing data were also rescaled, and dimensionality reduction by the eigenvector used in training was applied to them.

C. Result

TABLE I
DEFAULT PARAMETER SET

Parameter	Value
α	0.0003
β	50.0
γ	0.0005
δ	0.5
ϵ	0.03
t	0.05

In the experiment, we set the parameters as shown in Table I in training. Fig.8 shows the performance of our system according to date. The horizontal axis shows the days from January, 1, 2008. The number of days is less than 365 because there are a few days where the traffic data were not able to be gathered because of maintenance, and such days have been excluded in Fig.8. According to Fig.8, the false positive rate is 5% or less at most dates, and the detection rate is about 70%. However, there is some date when the false positive rate greatly exceeds 10%. (e.g., 25th to 29th day, 130th to 140th day, and so on.) Some of the rapid increases of false positive rate are caused by errors in labels in training data. From the characteristic of the honey pot, all the traffics that reach this network can be considered to be attacks. To generate the bench mark data set (i.e., a data set including attack data and normal data), normal traffic is generated from the server that serves DNS and mail in Kyoto University honey pot environment [12]. From a long term observation, we know that even when an attack occurs in the entire network, the amount of attack to this server is little, excepts for several days when DoS attacks to the server were seen and caused the high false positive rate. If we considers DoS attacks to this server using normal packets to be attacks, the false positive rate is lower than that given in Fig.8. (e.g., for 135th day, 90.2% $\rightarrow$ 4.6%.) Considering the fact that the false alarm rate of current IDSs is around 10% and the performance is not always stable, we can consider that our approach gives a competitive performance.

D. Analysis

There are 6 parameters in the proposed system. These are

- α, β : Stop condition of cell split.
- t : Convergence criterion of clustering.
- γ, δ, ϵ : Attack and normal condition in labeling.

However, considering their usage, there are some parameters that are insensitive to performance (e.g., t, ϵ and δ). In analysis of parameters, we use the default value in Table I unless the parameter was focused. Tables II, III and IV shows the performance of the proposed system according to these parameters. "FPR" and "DR" fields in the table represent the false positive rate and detection rate. In each table, we changed only one parameter and others were set to default values. Table II shows that system performance is insensitive

TABLE II
PERFORMANCE VARIATION ACCORDING TO $t(0 \leq t \leq 1)$

t	FPR(%)	DR(%)
0.01	2.72	69.94
0.05	2.73	69.42
0.1	2.73	69.42
0.3	2.88	73.40

TABLE III
PERFORMANCE VARIATION ACCORDING TO $\delta(0 \leq \delta \leq 1)$

δ	FPR(%)	DR(%)
0.1	10.34	89.13
0.2	2.84	73.44
0.3	2.78	71.80
0.5	2.73	69.42
0.8	2.61	53.62

TABLE IV
PERFORMANCE VARIATION ACCORDING TO $\epsilon(0 \leq \epsilon \leq 1)$

ϵ	FPR(%)	DR(%)
0.01	2.25	38.77
0.03	2.73	69.42
0.05	2.73	69.42
0.1	2.73	69.42

to t. In the experiment, there is iteration when t exceeds 0.3. Furthermore, a larger t (i.e., less iteration) makes a better performance, meaning the suitability of our initialization (Sect.III-A). From Table III, if we set δ within the proper region, i.e., $0.2 \leq \delta \leq 0.5$. It means our algorithm is not sensitive to the parameter. From Table IV, we find that ϵ has a similar feature with δ (i.e., the suitable region is $\epsilon > 0.03$).

TABLE V
PERFORMANCE VARIATION ACCORDING TO $\alpha(0 \leq \alpha \leq 1)$

α	FPR(%)	DR(%)
0.0001	2.95	54.27
0.0003	2.73	69.42
0.0005	2.46	59.33
0.001	3.25	58.51

TABLE VI
PERFORMANCE VARIATION ACCORDING TO $\beta(1 \leq \beta)$

β	FPR(%)	DR(%)
30	4.07	62.47
50	2.73	69.42
100	4.79	70.74
500	5.14	62.29

TABLE VII
PERFORMANCE VARIATION ACCORDING TO $\gamma(0 \leq \gamma \leq 1)$

γ	FPR(%)	DR(%)
0.0003	2.65	63.67
0.0005	2.73	69.42
0.0010	6.40	80.19
0.0015	6.63	84.63

Compared to insensitive parameters, α, β and γ are sensitive to performance. From Tables V and VI, we can find that the performance of the proposed system has a peak point concerning α and β. This is because system performance will be down if over-splitting and lack of splitting occurs. From Table VII, we can see that the both of false positive rate and

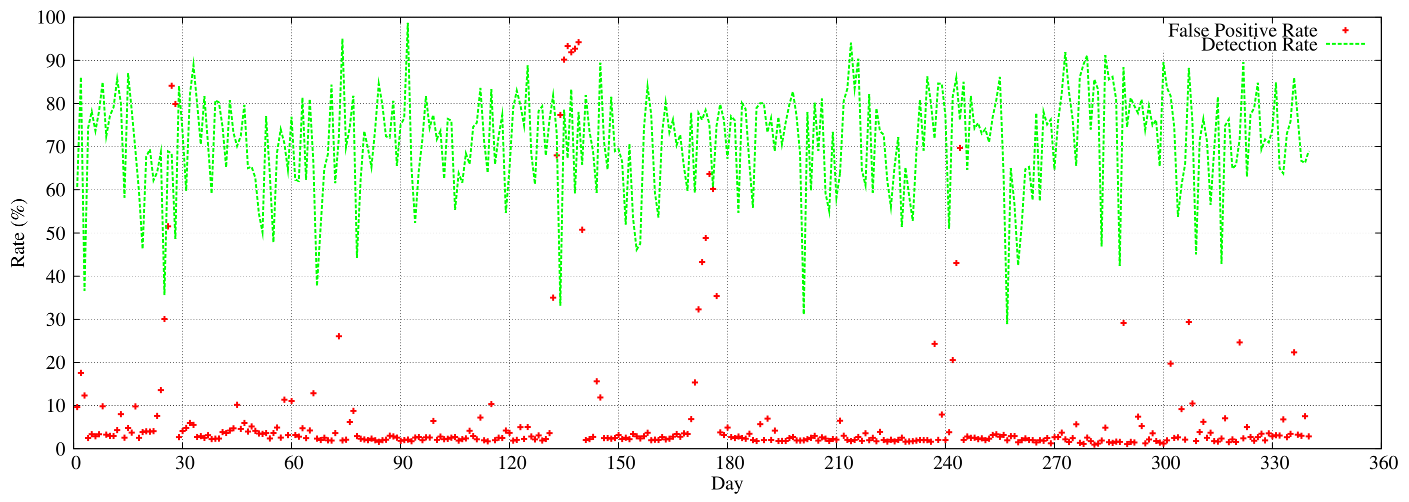

Fig. 8. Performance variation according to day

detection rate are increased by γ increase. The increase of γ means the space (i.e., the clusters which have little instances), which we regard as attack spreads widely. The detection rate grows in proportion to γ meaning that we can say attacks exist in sparse space, as we defined. From the parameter analysis, we consider there are 3 (not 6) parameters that need to be tuned when we deploy the proposed system.

E. Comparison with other methods

1) Performance: Figs.9 and 10 show the performance of the proposed system and others with the Kyoto 2006+ data set and KDD Cup 1999 data set[2] [1] over ROC curve [7]. KDD Cup 1999 data set contain a wide variety of intrusions simulated in a military network environment. It originated from 1998 DARPA Intrusion Detection Evaluation Program managed by MIT Lincoln Labs. The training data set consist of 4.9 million data instances. Each instance consist of 41 features of various types, and a class label that indicate either normal or one of attack types. 41 features consist of 34 continuous features and 7 discrete features. The test data contain 490,000 instances. There are 17 types of new attacks that are not present in training data and 20 types of known attacks that are present in training data. As the evaluation criterion, we used the portion of the ROC curve between the false positive rates of 0% to 7%. In our experiment for comparison, we use the same data set in [10] and [14] (i.e., 423,873 and 65,108 samples for training and testing in KDD Cup 1999 data set. November 2nd, 2007, and December 1st, 2007, for training and testing in Kyoto 2006+ data set). However the difference between our method and others is the method of normalization (i.e., other methods use [13] normalization scheme). We use continuous features excluding "hot" and "land" for KDD Cup 1999 data set because the distance between each discrete value has not relation, and there is no meaning of using grids to cut feature space, and "hot" and "land" features are not acceptable because these features indicate the answer (i.e., attack or not) by themselves. We chose some competitors, such as Song based K-means [10], Song based One-Class SVM [14], Y-means [3], K-means [8], Li [6]. From Fig.9, we can see our

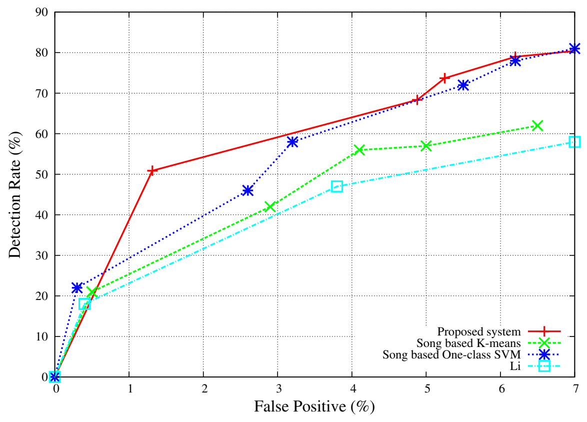

Fig. 9. ROC curve over honeypot data set

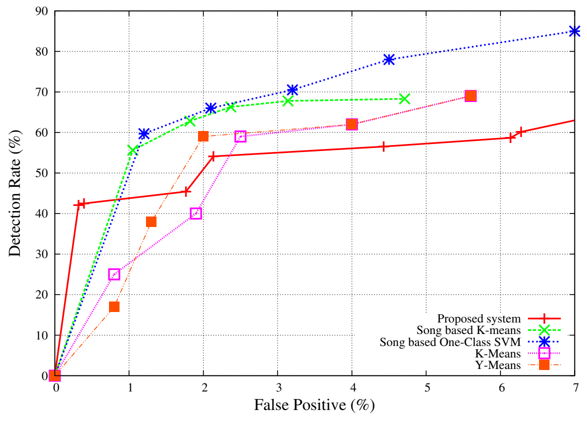

Fig. 10. ROC curve over KDD Cup 1999 data set

system is superior to others in false positive rate 1% to 6%.

[2]http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

From Fig.10, our system has a higher detection rate compared to others in false positive rates under 1%.

We also analyzed which process (i.e., PCA or grid partitioning) has the most impact to the performance. We use a random select method to choose initial cluster center which used by original K-means for comparison. Fig.11 shows the impact of grid partitioning is bigger than the impact of PCA in performance comparison. We can also figure out that the difference between "with PCA" and "without PCA" is little when we use random select. These results indicate the effectiveness of our grid partitioning, and the goodness of the compatibility of our grid partitioning and PCA.

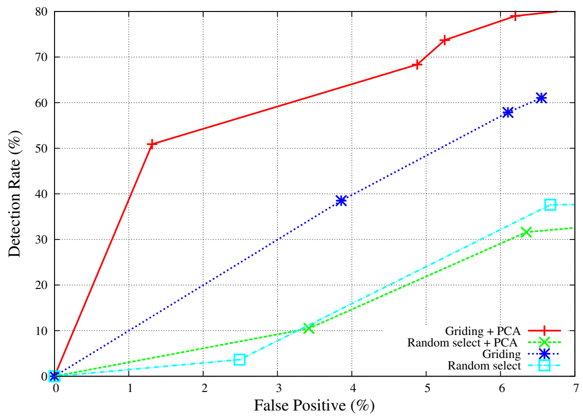

Fig. 11. Performance comparison according to PCA and grid partitioning (ROC curve over honeypot data set (December, 1st, 2007))

2) Time complexity: Time complexity of the proposed method is one of the most significant indicators to determine whether algorithms can be deployed in a real environment. In our system, time complexity of the additional processes, Initialization, Clustering, Labeling, Detecting are $\mathrm{O}(n \log k), \mathrm{O}(kn), \mathrm{O}(k), \mathrm{O}(kn)$. Since, in our system, k is from 50 to 200 and is smaller than n, we can say our algorithm has linear time complexity in the size of data. First of all, We show the time consumption for each training process of proposed method in Fig.12. It shows the process where the time consumption is the largest is Initialization, and both PCA and Clustering process are similar. In proposed system, we consider PCA consumes more time than the general preprocess for training. However, the cost of the calculation of PCA increases mainly in learning phase which is done off-line, and does not deteriorate the performance of testing too much because only the calculation of the products of eigen vectors and instances are necessary in the phase.

Figs.13 and 14 shows the training time and testing time according to number of instances without consider preprocessing time. This measurement of the proposed system was performed on a machine with Intel Xeon 3.2 GHz CPU and 2.0GB RAM and others on [14] environment (Intel Xeon 3.8 GHz and 3.5GB RAM). From Fig.14, we can see that the proposed system has a little good quality to Song based K-means which has real-time preprocessing capability. The reason why proposed method shows a clear reduction in testing

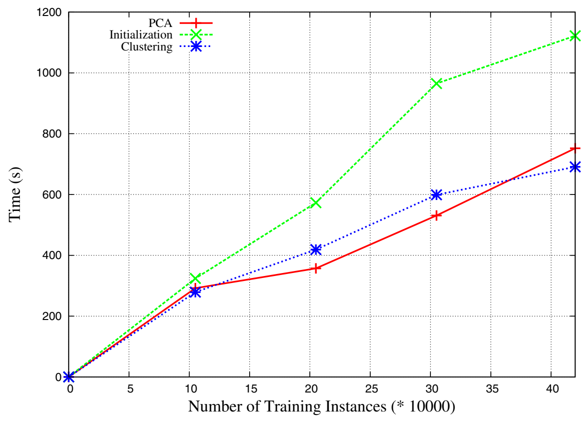

Fig. 12. Time Consumption for each training process

time when the number of training instance increase is because of reduction of the number of initial clusters. When the number of instances increases, the separation between dense and sparse spaces becomes clearer because of the tendency for the normal instances to gather around certain points. Cells are quickly judged whether it is dense or sparse by dense and sparse conditions (Sect.III **Step 4**) with same parameters. This causes the reduction of the number of initial clusters and the length of testing time.

V. Conclusion and Future works

In this paper, we have proposed a K-means based clustering algorithm for intrusion detection. To overcome the problem: unknown number of suitable initial cluster centers, we use an algorithm based cell partitioning that automatically generates cluster centers. We have evaluated our proposed method on two data sets with existing methods. The experimental results show our method achieves a higher detection rate with lower false positive rate. We also observe that only 3 parameters in our method are sensitive to performance. For future work, we need to improve the learning algorithm that can keep a steady detection performance, and need to make a detail analysis to data to show the detection capability of unknown attacks.

Acknowledgment

The authors are grateful to Dr. Jungsuk Song at the National Institute of Information and Communications Technology who provided their clustering program and data sets. This work was partially supported by Strategic Information and Communications R&D Promotion Programme (091603006).

References

[1] "the third international knowledge discovery and data mining tools competition dataset KDD99-Cup."

[2] H. Chipman and R. Tibshirani, "Hybrid hierarchical clustering with applications to microarray data," *Oxford Journals*, vol. 7, pp. 286–301, 2005.

[3] Y. Guan, A. Ghorbani, and N. Belacel, "Y-means: A clustering method for intrusion detection," in *Canadian Conference on Electrical and Computer Engineering*, 2010.

[4] J. A. Hartigan, *Clustering Algorithms*, 99th ed. New York, NY, USA: John Wiley & Sons, Inc., 1975.

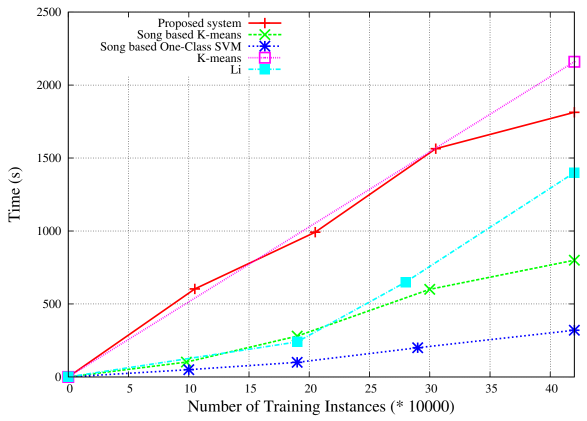

Fig. 13. Comparison training time

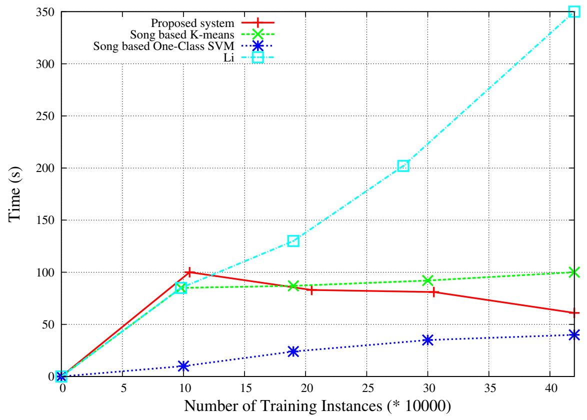

Fig. 14. Comparison testing time

[5] A. Jain, M. Murty, and P. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

[6] K. Li, H. Huang, S. Tian, and W. Xu, "Improving one-class svm for anomaly detection," in *Machine Learning and Cybernetics, 2003 International Conference on*, vol. 5. IEEE, 2004, pp. 3077–3081.

[7] R. Lippmann, D. Fried, I. Graf, J. Haines, K. Kendall, D. McClung, D. Weber, S. Webster, D. Wyschogrod, and R. Cunningham, "Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation," in *DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings*, vol. 2. IEEE, 2002, pp. 12–26.

[8] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 281-297. California, USA, 1967, p. 14.

[9] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. Smola, and R. Williamson, "Estimating the support of a high-dimensional distribution," *Neural computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[10] J. Song, K. Ohira, H. Takakura, Y. Okabe, and Y. Kwon, "A clustering method for improving performance of anomaly-based intrusion detection system," *IEICE Transactions on Information and Systems*, vol. 91, no. 5, p. 1282, 2008.

[11] J. Song, H. Takakura, and Y. Okabe, "Cooperation of intelligent honeypots to detect unknown malicious codes," in *Information Security Threats Data Collection and Sharing, 2008. WISTDCS'08. WOMBAT Workshop on*. IEEE, 2008, pp. 31–39.

[12] J. Song, H. Takakura, Y. Okabe, M. Eto, D. Inoue, and K. Nakao, "Statistical Analysis of Honeypot Data and Building of Kyoto2006+ Dataset for NIDS Evaluation," *BADGERS*, pp. 27–34, April 2011.

[13] J. Song, H. Takakura, Y. Okabe, and T. Kwon, "A robust feature normalization scheme and an optimized clustering method for anomaly-based intrusion detection system," *Advances in Databases: Concepts, Systems and Applications*, pp. 140–151, 2010.

[14] J. Song, H. Takakura, Y. Okabe, and Y. Kwon, "Unsupervised anomaly detection based on cluster and multiple one-class SVM," *IEICE Transactions on Commun*, vol. 92, no. 5, p. 1981, 2009.

[15] F. Wood, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometr. Intel. Lab. Syst*, vol. 2, pp. 37–52, 1987.