International Journal of Reliability, Quality and Safety Engineering
Vol. 14, No. 2 (2007) 169–187

World Scientific
www.worldscientific.com

CLUSTERING-BASED NETWORK INTRUSION DETECTION

SHI ZHONG* and TAGHI M. KHOSHGOFTAAR†
Computer Science and Engineering
Florida Atlantic University
777 West Glades Road
Boca Raton, FL 33431, USA
**zhong@cse.fau.edu*
†*taghi@cse.fau.edu*

NAEEM SELIYA
Computer and Information Science
University of Michigan – Dearborn
4901 Evergreen Road
Dearborn, MI 48128, USA
nseliya@umich.edu

Received 25 February 2005
Revised 1 March 2007

Recently data mining methods have gained importance in addressing network security issues, including network intrusion detection — a challenging task in network security. Intrusion detection systems aim to identify attacks with a high detection rate and a low false alarm rate. Classification-based data mining models for intrusion detection are often ineffective in dealing with dynamic changes in intrusion patterns and characteristics. Consequently, unsupervised learning methods have been given a closer look for network intrusion detection. We investigate multiple centroid-based unsupervised clustering algorithms for intrusion detection, and propose a simple yet effective self-labeling heuristic for detecting attack and normal clusters of network traffic audit data. The clustering algorithms investigated include, k-means, Mixture-Of-Spherical Gaussians, Self-Organizing Map, and Neural-Gas. The network traffic datasets provided by the DARPA 1998 offline intrusion detection project are used in our empirical investigation, which demonstrates the feasibility and promise of unsupervised learning methods for network intrusion detection. In addition, a comparative analysis shows the advantage of clustering-based methods over supervised classification techniques in identifying new or unseen attack types.

Keywords: Network intrusion detection; clustering algorithms; classification techniques.

1. Introduction

Considerable attention has been given to data mining approaches for addressing network security issues.[1–3] This is particularly due to the increasing dependence on computer networks for personal, business, and government activities. An intrusion

attack can result in several severity levels of incapacity, from loss of personal privacy to an enormous loss of business capital. An intrusion is any use of the given network that compromises its stability and/or security of information stored across the network. Network intrusion detection models are used for detecting intrusions or anomalous behavior.[4–6] There are generally two types of approaches taken toward network intrusion detection: anomaly detection and misuse detection.

In misuse detection, each network traffic record is identified as either normal or one of many intrusion types.[7] A classifier is then trained to discriminate one category from the other, based on network traffic attributes. Lee *et al.*[8] proposed a data mining framework for such a system. They used a series of data mining techniques, such as frequent episodes and association rules, to help extract discriminative features, which include various network traffic statistics. Obtaining correctly-labeled data instances, however, is difficult and time intensive, especially for new attack types and patterns. On the other hand, anomaly detection amounts to training models for normal traffic behavior and then classifying as intrusions any network behavior that significantly deviates from the known normal patterns.[9] Traditional anomaly detection algorithms often require a set of purely normal traffic data from which models can be trained to represent normal traffic patterns.

Clustering algorithms have recently gained attention[10–15] in related literature, since they can help current intrusion detection systems in several respects. An important advantage of using clustering or unsupervised learning to detect network attacks is the ability to find new attacks not seen before. This implies that attack types with unknown pattern signatures can be detected. A main objective of this study is to demonstrate this advantage of unsupervised learning using a simple (clustering-based attack detection) method. The proposed method is also shown to improve (assist) traditional classification-based intrusion detection models that often have difficulty in classifying new or unseen attacks correctly.

Clustering results can also assist the network security expert with labeling network traffic records as normal or intrusive. The amount of available network traffic audit data is usually large, making the expert-based labeling process of all records very tedious, time-consuming, and expensive. Additionally, labeling a large number of network traffic records can lead to errors being incorporated during the process. Grouping similar data together eases the task of labeling by experts. Instead of evaluating each data instance one by one, the expert can simultaneously label all (tens or hundreds) data instances in a cluster by observing the common characteristics of the cluster, with possibly very few mistakes, provided the clusters obtained are relatively "pure". A completely (100%) pure cluster is one that contains data instances only from one category (normal or a specific attack type or category). We investigate the performance of several clustering algorithms in terms of cluster purity as well as other performance criteria.

Recent works on clustering-based intrusion detection focus on constructing a set of clusters (based on unlabeled[11,13] or labeled[12] training data) to classify (future) test data instances. Such approaches do not completely exploit the advantages of

clustering-based methods. In contrast, we advocate clustering the (future) test data instances to be classified, and then use heuristics or existing labeled data to help label the clustered instances.

Portnoy *et al.*[11] presented a clustering method for detecting intrusions from unlabeled data. Unlike traditional anomaly detection methods, they cluster data instances that contain both normal behaviors and attacks, using a modified incremental k-means algorithm. After clustering, each cluster is labeled (as normal or attacks) based on the number of instances in the cluster. The heuristic is that very small clusters tend to be attacks. The self-labeled clusters are then used to detect attacks in a separate test dataset. Guan *et al.*[13] worked on the same idea of detecting intrusions but with a different clustering algorithm, namely an improved k-means algorithm that addresses the selection of number of clusters and the elimination of empty clusters.

Although unsupervised intrusion detection in general looks promising, we feel the approach used by Portnoy *et al.*[11] and Guan *et al.*[13] has a few problems. First, each cluster is self-labeled as attacks or normal, based purely on the number of instances in it. This is not reliable since we have seen in our experiments many relatively large attack clusters as well as small normal clusters. Secondly, the idea of detecting intrusions in a new dataset using the self-labeled clusters of the training dataset seems misguided. We feel that the purpose of unsupervised intrusion detection is to discover new attacks in a new dataset. It is more practical to run clustering algorithms on the new dataset and identify attacks by self-labeling.

Ye and Li[12] proposed a supervised clustering technique that constructs clusters from labeled training data and uses them to score the possibility of being attacks for test data instances. Performance better than decision tree classification models was reported. Lee *et al.*[8] emphasized the data-flow environment of network intrusion detection, aiming at real-time feature extraction and classification from network traffic data. We agree that an online, real-time, and adaptive intrusion detection system is the ultimate goal, towards which our online clustering-based approaches have provided a promising tool.

This paper extends our previous study[14,15] to demonstrate that our clustering-based method can outperform and enhance the state-of-the-art support vector machine algorithm in detecting unseen intrusion (attack) types. The primary contributions of this paper are:

1. A comprehensive comparative study of multiple clustering algorithms for analyzing large intrusion detection datasets was conducted. We analyzed the suitability of different clustering algorithms and empirically compared several centroid-based algorithms — k-means,[16] Mixture-Of-Spherical Gaussians,[17] Self-Organizing Map,[18] and Neural-Gas,[19] in terms of both clustering quality and run-time efficiency. To our knowledge the different clustering algorithms (in the paper) have not been presented in a relative comparative study for the network intrusion detection problem. Such a study holds practical merit and will benefit

future studies in this area, especially given the existence of many clustering methods.

2. A simple and effective self-labeling heuristic is proposed to detect and label attack clusters. The detection performance is evaluated by detection accuracies and ROC (Receiver's Operating Characteristics) curves for each of the aforementioned clustering algorithms.
3. A set of empirical investigations are designed to show the main advantage of our clustering-based intrusion detection method in identifying new attack instances. Results also demonstrate that our method can be used to help classification methods achieve better overall intrusion detection accuracies. More specifically, useful intrusion detection performance are obtained using clustering for classification of network traffic data.

This paper is structured as follows. The clustering algorithms investigated in this paper are introduced in the next section. We then present the proposed heuristic for self-labeling of clusters, followed by a description of the intrusion detection dataset and empirical settings, and a discussion of our empirical results. Finally, we present concluding remarks and some suggestions for future work. Some key abbreviations/notations used in this paper are summarized in Table 1.

2. Clustering Techniques

Clustering techniques can generally be divided into two categories: pairwise clustering and central clustering. Pairwise clustering algorithms are based on the pairwise

Table 1. Key notations.

Notation	Description
adr	attack detection rate
avg-pur	average purity metric of cluster
dos	denial of service attack type
EM	expectation-maximization algorithm
fpr	false positive rate
kmo	online k-means clustering algorithm
MOSG	mixture-of-spherical gaussian clustering algorithm
mse	mean squared error metric
N	number of instances in dataset
Neural-Gas	online Neural-Gas clustering algorithm
probe	reconnaissance attack type
r2l	remote-to-local attack type
ROC	Receiver's Operating Characteristic curve
SOM	self-organizing maps clustering algorithm
SVM	support vector machine classification algorithm
u2r	user-to-root attack type
x_i	the ith data vector
y_k	the kth cluster
η	percentage of normal instances
μ_k	centroid of the kth cluster

proximity affinities between the instances of a dataset. As pairwise distances need to be computed for all pairs in the dataset, such algorithms are based on efficient Eigen vector calculations and implement a combinatorial optimization method for data grouping based on proximity data.[20] Graphical clustering and spectral clustering methods fall into this category.[21,22] The computational complexity of pairwise clustering increases quickly with the number of instances, i.e., dataset size.

Central or centroid-based clustering refers to algorithms such as k-means where a cluster is defined by its centroid values for the different attributes. The cluster centroid is a model for a given cluster, and hence; such clustering methods are also known as model-based clustering. Data points are clustered based on closest distance to the centroids of the different clusters. Such clustering algorithms generally need an initialization parameter where the initial value of the centroid is specified. Central clustering algorithms are often more efficient than pairwise clustering algorithms.[23] Regular k-means [16] and EM (Expectation Maximization) clustering algorithms [17] fall into this category.

We choose centroid-based clustering over pairwise-based clustering due to the large size of the network traffic audit dataset. This was decided after initial studies on a variety of methods, including the CLUTO toolkit,[22] which is based on graph partitioning techniques for clustering. Preliminary experiments using CLUTO show that it takes several hours to cluster about 110K network traffic instances and the algorithm generated $500 \sim 700$ clusters due to required (for efficiency) sparsity for the constructed graph.[a] We could not efficiently get a desired (for comprehensibility) number of clusters, e.g., 100 or 200 as set by users.

Pairwise-based algorithms usually have a complexity of at least $O(N^2)$ (for computing the data-pairwise proximity measures), where N is the number of data instances. In contrast, centroid-based algorithms are more scalable, with a complexity of $O(NKM)$, where K is the number of clusters and M the number of batch iterations. In addition, all centroid-based clustering techniques have an online version which can be suitably used for adaptive attack detection in a data-flow environment.[8] An online version, compared to batch version, generally refers to how the centroid values are updated — either after each data point cluster assignment or after cluster assignments for a group of data points.

2.1. *K-means*

The widely used standard k-means algorithm[16] can be found in many other papers and its details are omitted here. It minimizes the mean-squared error (mse) objective function

$$E = \frac{1}{N} \sum_{n} \|x_n - \mu_{y_n}\|^2 \ , \tag{1}$$

[a]See the CLUTO toolkit manual for more details.

Algorithm: online k-means (kmo)

Input: A set of N data vectors $X = \{x_1, \ldots, x_N\}$ in $\mathbb{R}^d$ and number of clusters K.

Output: A partition of the data vectors given by the cluster identity vector $Y = \{y_1, \ldots, y_N\},\ y_n \in \{1, \ldots, K\}$.

Steps:

1. Initialization: initialize the cluster centroid vectors $\{\mu_1, \ldots, \mu_K\}$;
2. Loop for M iterations

   For each data vector x_n, set $y_n = \arg\min_k \|x_n - \mu_k\|^2$, and update the centroid μ_{y_n} as

$$\mu_{y_n}^{(new)} = \mu_{y_n} - \frac{\partial E}{\partial \mu_{y_n}} = \mu_{y_n} + \xi(x_n - \mu_{y_n}),$$

   where ξ is a learning rate usually set to be a small positive number (e.g., 0.05). The number can also gradually decrease in the learning process.

Fig. 1. Online k-means algorithm.

where $y_n = \arg\min_k \|x_n - \mu_k\|^2$ is the cluster identity of data vector x_n and μ_{y_n} is the centroid of cluster y_n. Unless specified otherwise, $\|\cdot\|$ represents L_2 norm. The popularity of the k-means algorithm is largely due to its simplicity, low time complexity, and fast convergence. Since batch k-means has been described in many papers,[24] here we instead present an online k-means algorithm in Fig. 1. Both are studied in Sec. 4.3, and they are named batch k-means and kmo, respectively.

2.2. *Mixture-Of-Spherical Gaussians (MOSG)*

The MOSG clustering using the EM algorithm[25] is shown in Fig. 2. It is a special case of Mixture-Of-Gaussians clustering, with an identity covariance matrix used for the Gaussian distribution of a cluster. Using an identity covariance matrix makes the algorithm more scalable to higher data dimensionality. A detailed derivation of the parameter estimation for the Mixture-Of-Gaussians and an excellent tutorial on the EM algorithm can be found in Blimes.[26] The MOSG algorithm is a "soft" clustering technique because each data vector is fractionally assigned to multiple clusters. In contrast, the data assignment in the k-means algorithm is considered "hard" (i.e., each data vector is assigned to only one cluster).

2.3. *Self-Organizing Map (SOM)*

Self-organizing map[18] is a competitive learning technique that can extract structural information from data and provide low-dimensional (1, 2, or 3-D) visualization through a topological map. Each cluster has a fixed coordinate in the topological map.

Algorithm: Mixture-Of-Spherical Gaussians clustering

Input: A set of N data vectors $X = \{x_1, \ldots, x_N\}$, model structure $\Lambda = \{\mu_k, \sigma_k, \alpha_k\}_{k=1,\ldots,K}$, where μ's and σ's are the parameters for Gaussian models and α's are prior parameters that are subject to $\alpha_k \geq 0, \forall k$ and $\sum_k \alpha_k = 1$.

Output: Trained model parameters Λ that maximizes the data likelihood $P(X|\Lambda) = \prod_n \sum_k \alpha_k p(x_n|\lambda_k)$, and a partition of the data vectors given by the cluster identity vector $Y = \{y_1, \ldots, y_N\},\ y_n \in \{1, \ldots, K\}$.

Steps:

1. Initialization: initialize the model parameters Λ;
2. E-step: the posterior probability of model k, given a data vector x_n and current model parameters Λ, is estimated as

$$P(k|x_n, \Lambda) = \frac{\alpha_k p(x_n|\lambda_k)}{\sum_j \alpha_j p(x_n|\lambda_j)},$$

   where the pdf $p(x|\lambda)$ is given by

$$p(x_n|\lambda_k) = \frac{1}{(\sqrt{2\pi}\sigma)^d} \exp\left(-\frac{\|x_n - \mu_k\|^2}{\sigma_k^2}\right);$$

3. M-step: the maximum likelihood re-estimation of model parameters Λ is given by

$$\mu_k^{(new)} = \frac{\sum_n P(k|x_n, \Lambda) x_n}{\sum_n P(k|x_n, \Lambda)},$$

$$\sigma_k^{(new)} = \frac{1}{d} \frac{\sum_n P(k|x_n, \Lambda) \|x_n - \mu_k\|^2}{\sum_n P(k|x_n, \Lambda)},$$

   and

$$\alpha_k^{(new)} = \frac{1}{N} \sum_n P(k|x_n, \Lambda)\ ;$$

4. Stop if $P(X|\Lambda)$ converges, otherwise go back to Step 2;
5. For each data vector x_n, set $y_n = \arg\max_k (\alpha_k p(x_n|\lambda_k))$.

Fig. 2. Mixture-Of-Spherical Gaussians clustering algorithm.

Let the map location of cluster k be ϕ_k; $K_\alpha(\phi_1, \phi_2) = \exp\left(-\frac{\|\phi_1 - \phi_2\|^2}{2\alpha^2}\right)$ be a neighborhood function; and $y_n = \arg\min_y \|x_n - \mu_y\|^2$. The batch SOM algorithm amounts to iterating between the following two steps:

$$P(y|x_n) = \frac{K_\alpha(\phi_y, \phi_{y_n})}{\sum_{y'} K_\alpha(\phi_{y'}, \phi_{y_n})},$$

and

$$\mu_y = \frac{\sum_x P(y|x)x}{\sum_x P(y|x)},$$

where α is a parameter controlling the width of the neighborhood function and decreases gradually during the clustering process. Note that the α parameter has the same functionality of a temperature parameter in an annealing process.

A unique feature of SOM is that the calculation of $P(y|x)$ is constrained by a topological map structure, which gives SOM the advantage that all resulting clusters are structurally related according to the user-specified topological map (which is good for visualization). However, SOM is usually not as good as k-means or Neural-Gas (discussed next) in terms of minimizing the mse function (1).

2.4. *Neural-Gas*

The (batch-version) Neural-Gas algorithm differs from the SOM clustering, only in how $P(y|x)$ is computed, i.e.,

$$P(y|x) = \frac{e^{-r(x,y)/\beta}}{\sum_{y'} e^{-r(x,y')/\beta}},$$

where β is an equivalent temperature parameter and $r(x, y)$ is a rank function that takes the value $k - 1$ if y is the kth closest cluster centroid to data vector x.

The original Neural-Gas algorithm[19] was proposed as an online algorithm, in which all cluster centroids are updated each time a data instance x_n is given, according to

$$\mu_y = \mu_y + \xi e^{-r(x_n,y)/\beta}(x_n - \mu_y).$$

In the online learning process, both the learning rate ξ and the temperature β gradually decrease, simulating an annealing procedure. It has been shown that the online version can converge faster and find better local solutions than clustering with SOM for certain problems.[19]

3. Intrusion Detection with Clustering

In addition to cluster sizes, inter-cluster distances are used in our clustering-based detection process. For the k-means, SOM, and Neural-Gas algorithms, the inter-cluster distance is defined as the Euclidean distance between two cluster centroids. For the MOSG algorithm, it is defined as the Mahalanobis distance between two cluster centroids,[27] calculated as $D(y_1, y_2) = (\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2})\|\mu_1 - \mu_2\|^2$.

We use a practical assumption about the intrusion data — the number of normal instances is much larger than that of attack instances. This is usually true in reality. However, unlike in Portnoy *et al.*[11], we do not make the strict hypothetical requirement that the percentage of attacks has to be less than a certain threshold (e.g., $\sim 1.5\%$).

Based on the assumption that normal instances dominate attack instances, our simple self-labeling heuristic for unsupervised intrusion detection consists of the following steps:

- Find the largest cluster, i.e., the one with the most number of instances, and label it *normal*. Assume its centroid is μ_0.
- Sort the remaining clusters in ascending order of the distance from each cluster centroid to μ_0. Within a cluster, sort the data instances in the same way (i.e., ascending order of distance from each data instance to μ_0).
- Select the first $N_1 = \eta N$ instances, and label them as *normal*, where η is the percentage of normal instances. The parameter η is the (given or estimated) fraction of all data instances as normal ones. By varying η, we can get a series of accuracy numbers can be used to draw ROC curves.
- Label all the other instances as *attacks*.

4. Empirical Investigation

4.1. *Case study description*

The network traffic audit data used in our case study comes from the data supplied as part of the 1998 DARPA off-line intrusion detection project. The original dataset of about 5 million records was collected from a controlled experiment in which a real-life military network was intentionally subjected to various attacks at specified time periods.

Each record, representing a connection between two network hosts according to some well defined network protocol, is described by 41 attributes (38 continuous or discrete numerical features and three categorical features) such as, duration of connection, number of bytes transferred, number of failed logic attempts, etc. The attributes are of three types[28]: intrinsic, content-based, and traffic-based. A record was labeled as either normal or one of four intrusion categories: denial of service (*dos*), reconnaissance (*probe*), remote-to-local (*r2l*), and user-to-root (*u2r*). A 10% sample consisting of about 500,000 records obtained from the UCI machine learning data repository was used in our study.

The *dos* attack type accounted for about 80% of the data, and consisted mostly of redundant *Neptune* and *Smurf* attacks. To reduce such a redundancy and the computational complexity of the problem (i.e., a dataset with mostly *dos* instances), the dataset was reduced to *109,910* records by randomly selecting only 1% of *Neptune* and 1% of *Smurf* attack types from the *dos* category. This is possible since the attack types of all instances in the dataset is known. For example, among 1000 *Neptune* attack instances, any 10 are randomly selected. Similarly, among 1000 *Smurf* attack instances, any 10 are randomly selected. This process is commonly adopted by other researchers. The *u2r* type was almost negligible.

We are interested in anomaly detection via unsupervised clustering algorithms; hence, all records labeled as attacks were considered as intrusion, while the

remaining were considered as normal. All clustering algorithms were implemented to cluster the network records without their intrusion/normal labels, thereby making it an unsupervised approach for intrusion detection. Once the set of clusters are formed we use a self-labeling heuristic to assign intrusion/normal labels to the traffic records. The actual labels are not used during the clustering process, but are only used for evaluating the detection performance of the algorithms.

The clustering algorithms used in our comparative study do not directly handle categorical data. The 3 categorical features, i.e., protocol type, service type, and connection flag, in the dataset were converted using the 1-of-N encoding scheme, which expanded the categorical features into 67 dimensions based on the counts for each categorical feature. Thus, the dataset used in our study consisted of 109,910 records, each having 105 feature dimensions. All feature dimensions were scaled to be within $[0, 1]$ to avoid potential scale problems.

It is worth mentioning here that feature construction (data preprocessing) is an important step in intrusion detection[8,29] and involves tremendous efforts. However, in this paper we simply use the same set of features that recent papers[11,30] have used, since our focus is on unsupervised intrusion detection algorithms.

4.2. *Empirical setting*

The k-means and Neural-Gas algorithms are written in C and compiled into mex files which can be run from Matlab. We implemented the MOSG algorithm in Matlab and used the SOM algorithm from the SOM Toolbox for Matlab provided by the Helsinki University of Technology.[b] Note that the Matlab-coded MOSG and SOM algorithms are relatively efficient due to vectorized programming and active optimization. All experiments are run on a PC with a 3.06GHz Pentium-4 CPU with 1GB DRAM and running Windows XP.

For the kmo algorithm, we use a learning rate that follows $\xi_m = 1.0(0.01/1.0)^{\frac{m}{NM}}$, where m is the online iteration number (from 0 to NM, N is the number of instances and M the number of batch iterations). For the online Neural-Gas algorithm, the learning rate follows $\xi_m = 0.5(0.005/0.5)^{\frac{m}{NM}}$, and the temperature follows $\beta_m = \beta_0(\beta_f/\beta_0)^{\frac{m}{NM}}$, where $\beta_0 = K/2$ (K is the number of clusters) and $\beta_f = 0.01/M$.

In order to study the effect of the total number of clusters on the intrusion detection results, we performed empirical studies with 100 and 200 total number of clusters. For clustering quality, we use the mean squared error (*mse*) objective (1) and average purity (*ave-pur*). The purity of a cluster is defined as the percentage of the most dominated instance category in the cluster, and average purity is the mean over all clusters. Its value can range from 0 to 1, with higher values representing better average purity. The run time of each algorithm is also recorded and compared.

[b]Available from http://www.cis.hut.fi/projects/somtoolbox/.

Each experiment is run ten times and the averages and standard deviations of the above measures are reported.

For evaluating intrusion detection results, we report false positive rate (*fpr*), attack detection rate (*adr*), and overall accuracy. The false positive rate is the percentage of normal instances that are labeled as attacks. The attack detection rate represents the percentage of all attack instances that are detected, i.e., labeled as attacks. The overall accuracy measures the percentage of all instances that are correctly labeled. We also report *ROC* curves, by varying the parameter η used in the detection method, to show the tradeoff between the false positive rate and the detection rate.

In the next two sections, we present two sets of experiments, each designed to demonstrate a different point. The first set is used to compare the algorithms presented in Sec. 2 whereas the second set is used to show the situations in which clustering-based intrusion detection methods outperform classification-based techniques (in terms of detection unseen attacks). We also show that clustering methods can be employed to help the performance of classification-based intrusion detection techniques.

4.3. *Experimental results — I*

In this section, we compare the aforementioned clustering algorithms on the whole data set (with 109,910 instances). Our results are promising based on a network traffic audit dataset with approximately 110K-instances, of which around 11.5% were attack instances. The proposed self-labeling heuristic (with $\eta = 11.5\%$) achieved an overall accuracy of 93.6%, a false positive rate of 3.6%, and a detection rate of 72%. It is logical that these numbers are not comparable to the accuracy that can be achieved by a supervised classifier since unsupervised intrusion detection tries to learn attacks without any labeled data. Detailed comparisons are presented below.

The *mse*, *average purity*, and *run time* results for the clustering algorithms with 100 clusters and 200 clusters, are shown in Tables 2 and 3, respectively. A standard deviation of 0.0 in the table signifies a very small (< 0.001) number. The *mse* results are not indicated for the MOSG algorithm, because it optimizes a different objective (maximizing data likelihood as seen in Fig. 2).

The kmo and Neural-Gas algorithms perform significantly better ($p < 5\%$) than the others in terms of *mse* and average purity, but kmo achieves the same high

Table 2. Summary of clustering results with 100 clusters.

	mse	*ave-pur*	*time* (seconds)
k-means	0.15 ± 0.03	0.93 ± 0.01	78.6 ± 14.6
kmo	0.06 ± 0.0	0.964 ± 0.0	249.7 ± 9.6
SOM	0.28 ± 0.0	0.932 ± 0.0	124.8 ± 1.3
Neural-Gas	0.05 ± 0.0	0.962 ± 0.003	1481.9 ± 204.5
MOSG		0.958 ± 0.006	366.7 ± 8.0

Table 3. Summary of clustering results with 200 clusters.

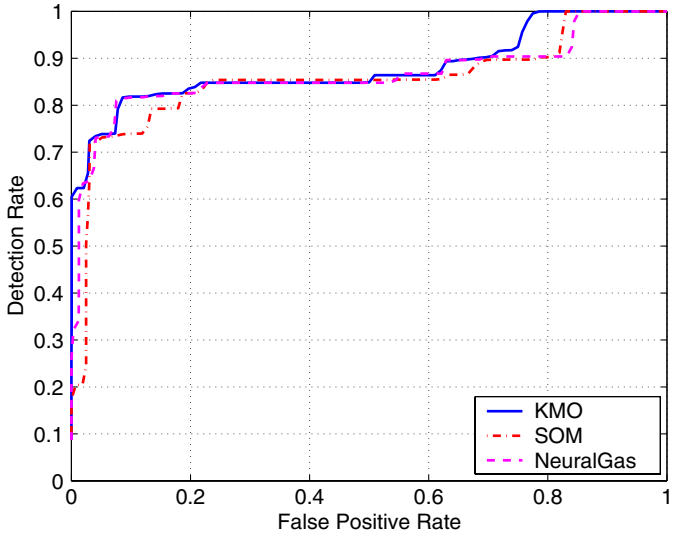
	mse	*ave-pur*	*time* (seconds)
k-means	0.11 ± 0.01	0.932 ± 0.01	155.1 ± 22.8
kmo	0.04 ± 0.0	0.977 ± 0.004	363.6 ± 19.0
SOM	0.14 ± 0.0	0.956 ± 0.0	147.5 ± 1.7
Neural-Gas	0.03 ± 0.0	0.970 ± 0.003	2524.9 ± 214.2
MOSG		0.963 ± 0.003	923.5 ± 31.8

quality with much less run time. Comparing the results for 100 clusters and those for 200 clusters, we observe that the k-means and Neural-Gas algorithms scale linearly with the number of clusters whereas the SOM algorithm does sub-linearly and the MOSG algorithm super-linearly. Although the batch k-means and SOM algorithms are computationally efficient, they do not generate coherent clusters like the other algorithms, as indicated by mse and purity measures. The kmo algorithm seems to be a desirable choice, with high clustering quality and relatively low time complexity.

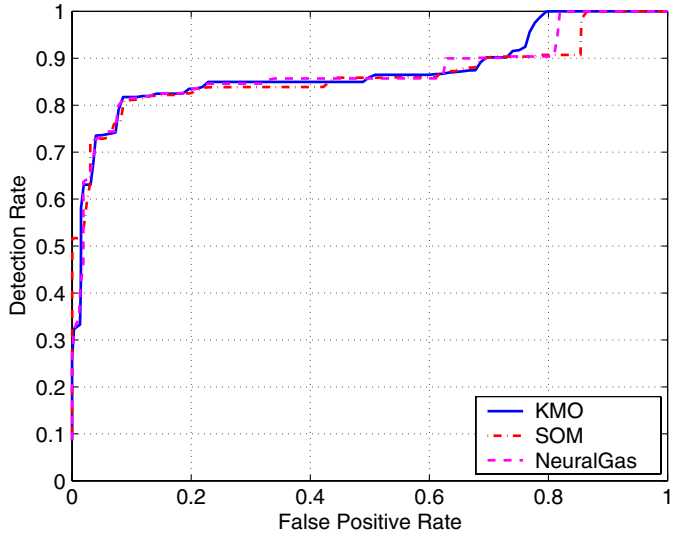
Since our aim is to detect network intrusion using clustering algorithms, we now analyze the unsupervised intrusion detection accuracies. Using the simple self-labeling heuristic presented in Sec. 3, we sort clusters according to their possibility of being normal in decreasing order and arrange data instances in a cluster in the same way. The possibility of being normal is measured by the distance to the centroid of the largest cluster. *ROC* curves can be constructed by dividing the sorted data instances into normal and intrusive categories at a series of cutting points.

Figures 3(a) and 3(b) show the *ROC* curves for the kmo, SOM, and Neural-Gas algorithms, with 100 clusters and 200 clusters, respectively. The curve for the batch k-means and MOSG algorithms are omitted for comprehensibility and better visualization, particularly because they are visibly worse than the other three algorithms. It can be seen that for 100 clusters, the SOM algorithm is the worst and the k-means and Neural-Gas algorithms work equally well for most of the false positive rates. In contrast, for 200 clusters, all three algorithms perform comparably most of the time and the SOM algorithm clusters performs extremely well at very low false positive rates, e.g., it can detect more than 50% attacks with a false positive rate of almost 0. Overall, the kmo and Neural-Gas algorithms seem to be the better ones and stable across different number of clusters.

We now discuss the accuracy results at one cutting point from the sorted list of clusters. Suppose the approximate percentage of attack instances is known *a priori* or from heuristics, we split the sorted cluster list at a point that generates the desired percentage. In this paper, we group the clusters as normal or intrusive in such a way that the number of data instances in attack clusters account for about 11.5% of the total population, reflecting the assumed distribution of the training data. We run each experiment 10 times and report the average and standard



(a)



(b)

Fig. 3. ROC curves for kmo, SOM, and Neural-Gas, with (a) 100 clusters and (b) 200 clusters.

deviation for overall accuracy, false positive rate, and attack detection rate, in Tables 4 and 5. The kmo and Neural-Gas algorithms again perform better than others, generating low *fpr* values and high overall accuracies; the performance difference to other algorithms are significant ($p < 5\%$) according to paired t-tests.

Table 4. Detection accuracy results with 100 clusters.

	accuracy(%)	*fpr*(%)	*adr*(%)
k-means	92.6 ± 0.7	4.7 ± 0.7	72.1 ± 2.7
kmo	93.7 ± 0.0	3.6 ± 0.0	72.7 ± 0.0
SOM	92.4 ± 0.0	5.0 ± 0.0	72.2 ± 0.0
Neural-Gas	93.5 ± 0.1	3.7 ± 0.1	72.0 ± 0.1
MOSG	91.5 ± 1.0	5.9 ± 1.0	71.4 ± 3.0

Table 5. Detection accuracy results with 200 clusters.

	accuracy(%)	*fpr*(%)	*adr*(%)
k-means	93.1 ± 0.3	4.2 ± 0.3	72.8 ± 0.7
kmo	93.6 ± 0.0	3.6 ± 0.0	72.1 ± 0.0
SOM	93.1 ± 0.0	4.3 ± 0.0	72.9 ± 0.0
Neural-Gas	93.6 ± 0.1	3.6 ± 0.1	72.5 ± 0.1
MOSG	92.7 ± 0.9	4.6 ± 0.9	71.8 ± 1.4

4.4. *Experimental results — II*

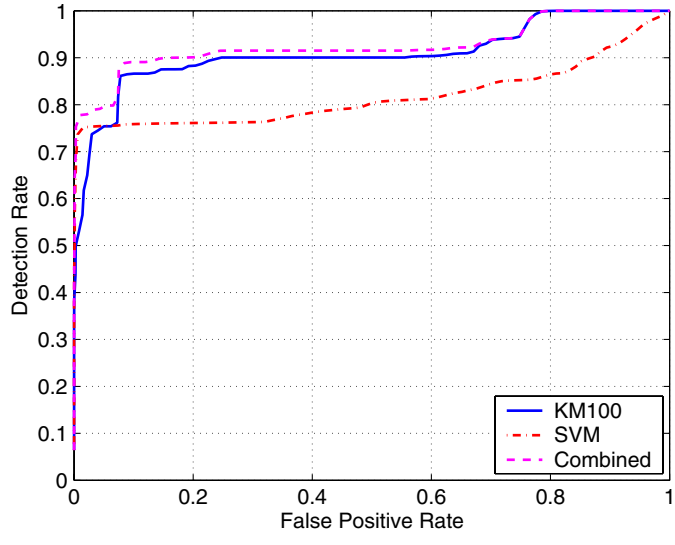
Now we present results to verify our hypothesis that clustering-based methods have an edge over classification techniques in identifying new or unseen attack types. The classification model to be compared is the state-of-the-art support vector machine method. We used the svm-light software package available online.[c] Default settings in the package are used.

We divided the 109,190 data instances into two parts — a training set and a test set. The training set is formed by randomly picking half of all *dos* attack instances and half of all normal instances. The rest goes into the test set. Therefore, the test set contains new attack types (*probe*, *r2l*, and *u2r*) that are not seen in the training set. The SVM algorithm is trained using the training set and then evaluated on the test set. An ROC curve can be obtained since the SVM algorithm outputs a continuous numerical score that indicates the probability of being normal (or intrusive) and can be used to order data instances. The kmo algorithm (picked according to our studies in the previous section) is applied directly to the test set.

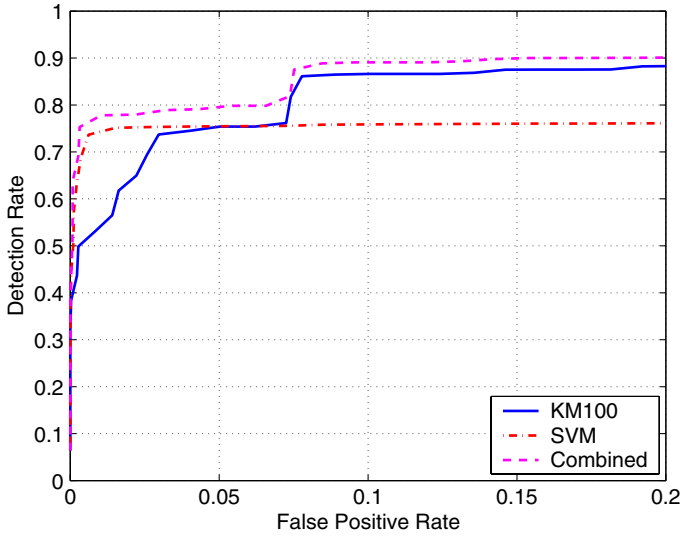
In addition to the SVM and kmo-based intrusion detection algorithms, we also compare a combined method, which orders data instances using both SVM and kmo results. Here we only present a simple method and show the benefit of combining classification and clustering for intrusion detection. More sophisticated combining methods can be studied in our future work. The simple combining strategy works like this: first we use the order in SVM results for those instances that are classified as attacks; then we use the order in kmo results for the remaining instances. The idea is to trust SVM for identifying seen attacks in the training data but kmo for identifying unseen attacks.

[c]http://www.cs.cornell.edu/People/tj/svm_light/

Figure 4 presents the ROC curves for the kmo, SVM, and combined methods. It can be seen that, for very small *fpr* values, SVM delivers better detection rates, but for $fpr > 5\%$, kmo works better. The combined method always generates better performance than both SVM and kmo methods.



(a)



(b)

Fig. 4. (a) ROC curves for kmo (100 clusters), SVM, and the combined method; and (b) a close-up look at low fpr area.

Table 6. Detection accuracy results with 100 clusters using k-means.

		Detection accuracy (%)				
		dos	*probe*	*r2l*	*u2r*	*overall*
fpr = 5%	kmo	69.81	99.62	6.48	49.45	75.39
	SVM	99.9	67.31	29.09	0	75.47
	combined	75.76	99.61	22.24	49.45	79.80
fpr = 10%	kmo	69.81	99.92	92.27	100	86.61
	SVM	99.96	68.1	29.16	0	75.86
	combined	75.76	99.93	92.53	100	89.08

To inspect the details of detection accuracies for individual attack types, we present the results for two fixed *fpr* values (5% and 10%, respectively) in Table 6. A quick glance at the table would suggest the overall accuracy measure does not provide a good insight into relative performances (especially at $fpr = 10\%$) of the clustering methods, and may lead to an erred conclusion that there is little significant difference in the relative performances. While this is reflective of the skewed distribution of the dataset with respect to the five categories (*dos*, *probe*, *r2l*, *u2r*, and *normal*), the classification accuracy results of the individual attack categories is of more interest, e.g., for the *dos* attack category SVM is clearly better than kmo.

The kmo-based method detects more *probe* and *u2r* attacks. In fact, the SVM method detects zero *u2r* attacks for the two given *fpr* levels, which indicates that *u2r* attacks are quite different from *dos* attacks in the current representation. As stated earlier, the SVM method always detects more *dos* attacks than the kmo-based method due to its exposure to *dos* attacks in the training data. For *r2l* type attacks, SVM works better at $fpr = 5\%$ but kmo is better at $fpr = 10\%$. The combined method catches more other-than-*dos* attacks than SVM and produces higher overall detection accuracies, demonstrating the potential of combining classification and clustering for intrusion detection. There is also room for further improvements, as indicated by the observation that the combined method still cannot compare with SVM in detecting *dos* attacks.

5. Conclusion

The feasibility of unsupervised intrusion detection using multiple centroid-based clustering algorithms is investigated in this study. Considering the dynamic nature of network traffic intrusions, unsupervised intrusion detection is more appropriate for anomaly detection than classification-based intrusion detection methods. Since the attack labels are not used during unsupervised intrusion detection, changes in the behavior and characteristics of network attacks can be accommodated with relative ease.

An empirical study consisting of different clustering algorithms is performed with a case study of network traffic data obtained from the DARPA 1998 offline

intrusion detection project. A comparative analysis and evaluation of the clustering algorithms yielded reasonable intrusion detection rates.

In addition, a simple yet effective self-labeling heuristic for labeling clusters as normal or intrusive (anomalous) is proposed and evaluated. Comparisons with classification-based intrusion detection methods on a completely new network traffic dataset demonstrated the usefulness and feasibility of clustering-based methods in both detecting new attack types and assisting classification techniques in achieving better intrusion detection performance.

Promising clustering and detection results encourage us to proceed our future work in several directions. A further detailed analysis of individual clusters can be done by identifying the precise attack category associated with a cluster and the discriminating features that are unique to a given cluster. In addition, feature selection/weighting for clustering will be investigated. This will eventually enhance our understanding and detection of new attack categories. Sophisticated self-labeling techniques, taking into consideration of additional network security domain knowledge, can be developed to improve the performance of clustering-based intrusion detection. Finally, incremental algorithms can be built by extending the competitive learning versions of k-means (e.g., according to[31,32]), making them more suitable for adaptive intrusion detection systems.

Acknowledgments

We thank Dr. Hoang Pham, Editor-in-chief, and the anonymous referees for their comments and critique which contributed in improving this paper. We are also grateful for getting assistance with reviews from the current and former members of the Empirical Software Engineering Laboratory and the Data Mining and Machine Learning Laboratory, both located at Florida Atlantic University, Boca Raton, FL.

References

1. V. Kumar, Parallel and distributed computing for cybersecurity. *IEEE Distributed Systems* **6**(10) 2005.
2. Z.-X. Yu, J.-R. Chen and T.-Q. Zhu, A novel adaptive intrusion detection system based on data mining, in *Proceedings IEEE International Conference on Machine Learning and Cybernetics* **4** (IEEE Computer Society) (2005), pp. 2390–2395.
3. X. Zhu, Z. Huang and H. Zhou, Design of a multi-agent based intelligent intrusion detection system, in *Proceedings 1st International Symposium on Pervasive Computing and Applications* (Urumqi, China, 2006) (IEEE Computer Society), pp. 290–295.
4. W. Li, K. Zhang, B. Li and B. Yang, An efficient framework for intrusion detection based on data mining, in *Proceedings 2005 ICSC Congress on Computational Intelligence Methods and Applications* (IEEE Computer Society) (2005), p. 4.
5. C.-T. Lu, A. P. Boedihardjo and P. Manalwar, Exploiting efficient data mining techniques to enhance intrusion detection systems, in *Proceedings IEEE International Conference on Information Reuse and Integration* (Las Vegas, NV) (IEEE Computer Society, 2005), pp. 512–517.
6. T. M. Khoshgoftaar, C. Seiffert and N. Seliya, Labeling network event records for intrusion detection in a wireless lan, in *Proceedings IEEE International Conference on*

Information Reuse and Integration (Waikoloa Village, HI) (IEEE Computer Society) (2006), pp. 200–206.
7. J. Zhang and M. Zulkernine, A hybrid network intrusion detection technique using random forests, in *Proceedings 1st International Conference on Availability, Reliability and Security* (IEEE Computer Society) (2006), p. 8.
8. W. Lee, S. Stolfo and K. Mok, Mining in a data-flow environment: Experience in network intrusion detection, in *Proc. 5th ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining* (San Diego, CA) (1999), pp. 114–124.
9. D. E. Denning, An intrusion detection model, *IEEE Trans. Software Engineering* **13** (1987) 222–232.
10. E. Eskin, Anomaly detection over noisy data using learned probability distributions, in *Proc. 17th Int. Conf. Machine Learning* (San Francisco, CA) (2000), pp. 255–262.
11. L. Portnoy, E. Eskin and S. Stolfo, Intrusion detection with unlabeled data using clustering, in *ACM Workshop on Data Mining Applied to Security* (Philadelphia, PA) (2001).
12. N. Ye and X. Li, A scalable clustering technique for intrusion signature recognition, in *Proc. 2nd IEEE SMC Information Assurance Workshop* (2001), pp. 1–4.
13. Y. Guan, A. A. Ghorbani and N. Belacel, Y-means: A clustering method for intrusion detection, in *Canadian Conference on Electrical and Computer Engineering*, Montral, Qubec, Canada (2003), pp. 1–4.
14. T. M. Khoshgoftaar, S. V. Nath, S. Zhong and N. Seliya, Intrusion detection in wireless networks using clustering techniques with expert analysis, in *Proceedings 4th International Conference on Machine Learning and Applications* (Los Angeles, CA) (2005), p. 6.
15. S. Zhong, T. M. Khoshgoftaar and N. Seliya, Evaluating clustering techniques for network intrusion detection, in *10th ISSAT Int. Conf. on Reliability and Quality Design* (Las Vegas, Nevada, USA) (2004), pp. 173–177.
16. J. MacQueen, Some methods for classification and analysis of multivariate observations. In *Proc. 5th Berkeley Symp. Math. Statistics and Probability* (1967), pp. 281–297.
17. J. D. Banfield and A. E. Raftery, Model-based Gaussian and non-Gaussian clustering, *Biometrics* **49**(3) (1993) 803–821.
18. T. Kohonen, *Self-Organizing Map* (Springer-Verlag, New York, 1997).
19. T. M. Martinetz, S. G. Berkovich and K. J. Schulten, Neural-Gas network for vector quantization and its application to time-series prediction, *IEEE Trans. Neural Networks* **4**(4) (1993) 558–569.
20. B. Fischer, T. Zoller and J. M. Buhmann, Path based pairwise data clustering with application to texture segmentation, *Lecture Notes in Computer Science* **2134** (2001) 235–250.
21. G. Karypis, E.-H. Han and V. Kumar, Chameleon: Hierarchical clustering using dynamic modeling, *IEEE Computer* **32**(8) (1999) 68–75.
22. G. Karypis, *CLUTO — A Clustering Toolkit*, Dept. of Computer Science, University of Minnesota, May 2002. http://www-users.cs.umn.edu/∼karypis/cluto/.
23. S. Zhong and J. Ghosh, A unified framework for model-based clustering, *Journal of Machine Learning Research* **4** (2003) 1001–1037.
24. S. Zhong, T. M. Khoshgoftaar and N. Seliya, Analyzing software measurement data with clustering techniques, *IEEE Intelligent Systems* **19**(2) (2004) 20–27.
25. A. P. Dempster, N. M. Laird and D. B. Rubin, Maximum-likelihood from incomplete data via the EM algorithm, *Journal of the Royal Statistical Society B* **39**(1) (1977) 1–38.

26. J. A. Blimes, A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models, Technical report, University of California at Berkeley (1998).
27. C. M. Bishop, *Neural Networks for Pattern Recognition* (Oxford University Press, 1995).
28. T. M. Khoshgoftaar and M. E. Abushadi, Resource-sensitive intrusion detection models for network traffic, in *Proceedings of the 8th IEEE International Symposium on High Assurance Systems Engineering* (Tampa, Florida) (2004), pp. 249–258.
29. W. Lee, S. Stolfo and K. Mok, A data mining framework for building intrusion detection models, in *Proc. IEEE Symposium on Security and Privacy* (1999).
30. A. H. Sung and S. Mukkamala, Identifying important features for intrusion detection using support vector machines and neural networks, in *Proc. IEEE Symposium on Applications and the Internet* (Orlando, FL) (2003), pp. 209–216.
31. P. S. Bradley, U. M. Fayyad and C. Reina, Scaling clustering algorithms to large databases, in *Proc. 4th ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining* (1998), pp. 9–15.
32. F. Farnstrom, J. Lewis and C. Elkan, Scalability for clustering algorithms revisited, *SIGKDD Explorations* **2**(1) (2000) 51–57.

About the Authors

Shi Zhong was an Assistant Professor in Florida Atlantic University. Currently, he works at Yahoo! Data Mining and Research Group in Sunnyvale, California. He holds a Ph.D. in Computer Engineering from The University of Texas at Austin. His research interest includes data mining, machine learning, and intelligent information retrieval. He is a member of IEEE and IEEE computer society.

Taghi M. Khoshgoftaar is a Professor of the Department of Computer Science and Engineering, Florida Atlantic University and the Director of the Empirical Software Engineering and Data Mining and Machine Learning Laboratories. His research interests are in software engineering, software metrics, software reliability and quality engineering, computational intelligence, computer performance evaluation, data mining, machine learning, and statistical modeling. He has published more than 300 refereed papers in these areas. He is a member of the IEEE, IEEE Computer Society, and IEEE Reliability Society. He was the program chair and General Chair of the IEEE International Conference on Tools with Artificial Intelligence in 2004 and 2005 respectively. He has served on technical program committees of various international conferences, symposia, and workshops. Also, he has served as North American Editor of the Software Quality Journal, and is on the editorial boards of the journals Software Quality and Fuzzy systems.

Naeem Seliya is an Assistant Professor of Computer and Information Science at the University of Michigan — Dearborn. He received his Ph.D. in Computer Engineering from Florida Atlantic University, Boca Raton, FL in 2005. His research interests include software engineering, data mining and machine learning, computer data and network security, and computational intelligence. He is a member of IEEE, IEEE Computer Society, and ACM.