An Efficient Clustering-Based Classification Approach for Intrusion Detection

Huu-Hoa Nguyen, Nouria Harbi and Jérôme Darmont

Université de Lyon (ERIC Lyon 2)
5 avenue Pierre Mendès-France, 69676 Bron Cedex, France
nhhoa@eric.univ-lyon2.fr, {nouria.harbi, jerome.darmont}@univ-lyon2.fr

Abstract. The need to increase accuracy in detecting sophisticated cyber attacks poses a great challenge not only to the research community but also to corporations. So far, many approaches have been proposed to cope with this threat. Among them, data mining has brought on remarkable contributions to the intrusion detection problem. However, the generalization ability of data mining-based methods remains limited, and hence detecting sophisticated attacks remains a tough task. In such a context, this paper presents a novel method based on both clustering and classification for developing an efficient intrusion detection system (IDS). The key idea is to take useful information exploited from fuzzy clustering into account for the process of building an IDS. To this aim, we first present theoretical cornerstones to construct additional cluster features for an intrusion detection dataset. Then, we come up with an algorithm to generate an IDS based on such cluster features and the original input features. Finally, we experimentally prove that our method is considerably superior to several state-of-the-art methods.

Keywords: classification, fuzzy clustering, intrusion detection, cyber attack.

1 Introduction

In recent years, with the dramatically increasing use of network-based services and the vast spectrum of information technology security breaches, more and more organizational information systems are subject to attack by intruders. Among many approaches proposed in the literature to deal with this threat, data mining (or machine learning), an approach regarding the task of detecting cyber attacks as a classification problem, brings on a noticeable success to the development of high performance intrusion detection systems (IDSs). The preeminence of such an approach lies in its good generalization abilities to correctly classify (or detect) both known and unknown attacks. However, as an inherent essence, the effectiveness of data mining-based IDSs depends heavily upon the quality of IDS datasets. In practice, IDS datasets are often extracted from raw traces in a chaotic system environment, and hence could hold implicit deficiencies, e.g., the existence of noise in class labels due to mistake in measurement, and the lack of base features. Moreover, due to the sophisticated

characteristic of attacks and diversification of normal events, different regions of data space could behave differently, i.e., true class labels could seriously be interlaced.

Such factors pose a great difficulty for inducers (learning algorithms) to identify appropriate decision boundaries from the input space (i.e., the initial or original feature set) of IDS datasets. In other words, when the input space is not robust enough to discriminate class labels, making further treatments from alternative knowledge sources as new supplemental features is highly desirable. To this aim, one common approach is to transform the input space into a higher dimensional space from which data are more separable. New additional features can be found by either manual ways based on prior knowledge or automatic analysis methods (e.g., factor or principle component analysis). For example, the XOR Boolean function classification problem is inseparable in the two-dimensional space $\{A_1, A_2\}$ but well separate in the three-dimensional space $\{A_1, A_2, A_1 \wedge A_2\}$. However, in a high dimensional input space, finding new relevant features is a tough task that often requires human's intervenient analyses, but derived features are sometimes not good as expectation. As a result, in practice, one often applies standard dimensional-transformation methods (e.g., polynomial, radial basic function, or sigmoid kernel) to application domains where class discrimination is ambiguous and additional features are hard to be identified. Yet, such methods are greatly affected by input parameters and data distribution, thus not always achieving a high performance classifier. In this sense, it is desirable to find additional features in such a less complex way that general-purpose algorithms such as standard Decision Tree (DT) or Support Vector Machines (SVM) with standard kernels can learn the data more efficiently.

Such a context motivates us to propose a novel approach that treats fuzzy cluster information as additional features. These features are selectively incorporated into the input space for building an efficient IDS. To this goal, we first present essences to construct and select fuzzy cluster features. Then, we propose a concrete algorithm for generating a high performance IDS. Eventually, we experimentally show that our proposed solution approach outperforms several other methods.

The remainder of this paper is organized as follows. Section 2 surveys some noticeable data mining-based intrusion detection methods, whereas Section 3 presents the problem formulation of our approach. Section 4 describes our proposed solution approach for generating an efficient IDS. Section 5 shows the experimental results. Section 6 finally gives a conclusion of highlight viewpoints and perspectives of the method we propose.

2 Related Work

This section provides a succinct survey of some noticeable studies related to data mining-based intrusion detection methods. These studies encompass both misuse and anomaly intrusion detection.

Portnoy et al. propose an algorithm that takes the radius of cluster as an input threshold parameter [9]. The algorithm partitions an unlabeled training dataset into clusters based on a distance measure (e.g., Euclidean distance) and the cluster radius threshold. Then, clusters are sorted according to the number of data points inside each

cluster. Subsequently, some smallest top percents of the sorted clusters are labeled as *normal*, whereas the rest is labeled as *attack*. In the detection phase, for a testing data point *p*, the algorithm calculates all distances from *p* to all clusters, and then takes the *k* smallest distances to apply a *majority voting* method for determining a class label for *p*. In another approach, Eskin et al. present an intrusion detection model based principally on calculating an outlier score for a given data point [10]. The fundamental notion is that the points in dense regions are assigned to a smaller outlier score than the points in sparse regions. The authors employ the *Canopy Clustering* technique to reduce complexity for computing the k-nearest neighbors of each point.

By contrast, Wang and Stolfo introduce a payload-based method by computing the profile byte-frequency distribution and standard deviation of payload network traffic during the training phase [11]. In the detection phase, a similarity measurement for a test instance is calculated based on a pre-built profile in the training phase. Unlike the header-based methods that are often used to detect *scanning* and *probing* attacks (e.g., *DoS*, *Probe*), the payload-based method is thought of as efficient in detecting worms that transfer bad data to a service or application.

Giacinto et al. describe an IDS architecture including multiple one-class k-means classifiers [12]. Each classifier is trained from a training subset containing a specific attack type belonging to a specific attack class. Then, the final result is achieved by fusing classifier outputs using the *Decision Template* method.

Fan et al. present a method that uses a decision tree inducer together with injection of artificial anomalies to detect unknown and known network intrusions [16]. Unlike other methods that assign a default class to *unknown* instances, the authors introduce a sampling method to generate a new training set that can handle both misuse and anomaly detection. The idea behind this method is to amplify the density of points in sparse regions in the initial training set, based on the distribution of attribute values, to discover a boundary between known classes and anomalies. In other words, the proposed method automatically injects *distribution*-based artificial anomalies into the initial training set to increase the ability of the classifier in distinguishing *anomalous* from *normal* network traffic.

On the other hand, Wu and Yen construct a decision tree-based IDS by introducing as a parameter the different proportions of *normal* instances in a training set to obtain the best tree [18]. More specifically, the training set is sampled into several different proportions based on the instance space of the *normal* class. The overall evaluation is based on the average value of results.

Many hybridized intrusion detection methods that employ both clustering and classification techniques are widely introduced by the community. For example, Shon and Moon develop a hybrid IDS using multiple techniques [13]. This study proposes a hybrid algorithm, called Enhanced SVM, by combining *Soft-Margin* SVM (a supervised classifier) and *One-Class* SVM (an unsupervised classifier). The idea behind this IDS includes two phases. In the first phase, the system uses TCP/IP Fingerprinting as an online filter to drop malformed incoming packets, a Genetic Algorithm (GA) to select relevant feature sets, and a Self-Organizing Map (SOM) to create packet profiles. In the second phase, the system trains the Enhanced SVM classifier by using the packet profiles generated from the SOM.

In addition, Xiang et al. come up with a multi-level hybrid prototype by combining two techniques, i.e., Supervised Decision Tree and Unsupervised Bayesian Net [14]. The prototype is hierarchically structured in forms of class labels in training set.

Finally, Depren et al. present a hybrid IDS architecture comprising three modules, i.e., anomaly analyzer, misuse analyzer, and decision support system (DSS) [15]. The anomaly analyzer uses an SOM to model *normal* network traffic, whereas the misuse analyzer employs a Decision Tree to classify attacks. DSS aims at interpreting the combined results of anomaly and misuse analyzers. The anomaly analyzer is further specialized into three sub-analyzers based on three different protocol types, i.e., TCP, UDP, and ICMP. Given a test instance, both anomaly and misuse analyzers are operated concurrently to identify types of the instance. Then, DSS assigns a class label for the test instance based on the results from the two analyzers.

3 Problem formulation

Clustering aims to organize data into groups (clusters) according to their similarities measured by some concepts. In metric spaces, similarity is often defined in term of a distance norm measured between data vectors. Because the merit degree of dimensions is different, dimensions are often weighted by a merit measure (e.g., entropy) regarding to a given distance formula. Unlike crisp clustering that crisply assigns each data point to a separate cluster, fuzzy clustering allows each data point to belong to various clusters with different membership degrees (or weights). Fuzzy clusters are expressed by their centers (or centroids) that are simultaneously found in the partitioning process of a fuzzy clustering algorithm. The number of clusters (k) is often inputted as a parameter to a fuzzy clustering algorithm. The $n \times k$ membership matrix $W=\{w_{ij} \in [0,1]\}$ of n data points is found in the fuzzy clustering process. For example, Figure 1 describes the instance space of a training set partitioned into four fuzzy clusters, where membership weights that data point x_1 belongs to clusters '1', '2', '3', and '4' are 0.3, 0.14, 0.16, and 0.4, respectively.

Let us first denote $S=\{X,Y\}$ the original training set of n data points $X=\{x_1,\ldots,x_n\}$, where each point x_i is an m-dimensional vector $(x_{i1},\ldots,x_{im})$ and assigned to a label $y_i \in Y$ belonging one of the c classes $\Omega=\{\omega_1, \ldots,\omega_c\}$.

Let $B = \{b_i \mid b_i = \max(w_{ij}),\ j = 1...k,\ i = 1...n\}$ hold the maximum membership weight of each point x_i, and $Z = \{z_i \mid z_i = \arg\max_j (w_{ij}),\ i = 1...n\}$ contains the cluster (symbolic) number assigned to each point x_i.

For conciseness in describing the approach, we term two column matrices Z and B as two "basic cluster features". In addition, we name the j^{th} column of the membership matrix (W) as P_j, and term the columns P_1, ..., P_k as "extended cluster features". We also term the training set added cluster features $\{X, Z, B, P_1, \ldots, P_k, Y\}$ as a "manipulated training set". These notations and terminologies are specifically depicted Figure 1.

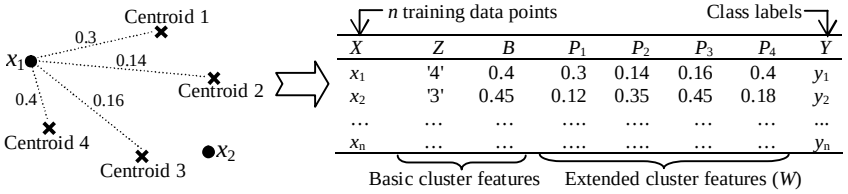

Fig. 1. A manipulated training set, resulting from adding cluster features into the input space.

The problem formulation follows: "Given a training set $S=\{X,Y\}$ and an inducer I, the goal is to find a high performance classifier induced by I over the m initial features of S and the supplemental cluster features $\{Z, B, P_1, P_2, \ldots, P_k\}$ resulting from a parameterized-by-k fuzzy clustering based on X".

Undoubtedly, fuzzy clustering has a great potential in expressing the latently natural relationships between data points. Here, an arising question is that whether information about fuzzy clusters benefits certain inducing types. In our investigation, there exist several types of inducers to which fuzzy cluster features bring about helpfulness. For example, in Support Vector Machine (SVM) context, SVM's decision boundary often falls into a low density region, but the true boundary might not pass through this region, thus resulting in a poor classifier. However, when supplemented with relevant cluster features, data points in high dimensional space are likely to be more uniform, hence avoiding an improper separation across this region. In fact, the crucial factor to the success of SVM lies in a kernel trick that plays the role as mapping the initial input space to a much higher dimensional feature space, where the transformed data are expected to be more separable from a linear hyper-plane function. In order words, while other inducers somewhat find dimensionality a curse, blessing of dimensionality could enable SVM to be more effective. In this sense, incorporating relevant cluster features into the initial input space benefits SVM inducers.

Another consideration relates to Univariate Decision Tree (DT) setting. Due to its greedy characteristic, DT inducer examines only one ahead partitioning step for growing child trees, rather than considering deeper partitioning steps that can achieve a better tree. This characteristic can lead to an improper tree-growing termination, and thus generate a poor classifier. In this sense, incorporating relevant cluster features into the initial input space helps DT inducer choose splits more appropriately for tree growing, hence achieving a more efficient classifier (decision tree).

4 Fuzzy Cluster Feature-based Classification

In this section, we point out the ways to generate and select cluster features for inducers. Then, we come up with a straightforward algorithm for building a high performance classifier from the initial features and supplemental cluster features.

4.1 Cluster Feature Generation and Selection

Basically, cluster features can be generated by any fuzzy clustering algorithm. However, for concreteness, we express cluster features in term of the fuzzy *c*-means clustering [17], which solves the minimization problem to the objective function of Formula 1. In a common form, the objective function (Formula 1) reaches to a minimum over *W* (membership matrix) and *V* (centroids), by Formulas 2 and 3.

$$f_{obj}(X,W,V) = \sum_{j=1}^{k}\sum_{i=1}^{n} w_{ij}^{\alpha} d(x_i, v_j)^2 \text{ , subject to the constrain } \sum_{j=1}^{k} w_{ij} = 1 \quad (1)$$

$$v_j = \left(\sum_{i=1}^{n} (w_{ij})^{\alpha} x_i\right) \Big/ \left(\sum_{i=1}^{n} (w_{ij})^{\alpha}\right) \quad (2)$$

$$w_{ij} = \left(\frac{1}{d(x_i, v_j)^2}\right)^{\frac{1}{\alpha-1}} \Bigg/ \sum_{q=1}^{k} \left(\frac{1}{d(x_i, v_q)^2}\right)^{\frac{1}{\alpha-1}} \quad (3)$$

where k is the number of clusters, α is a fuzzy constant, and $d(x_i, v_j)$ is the distance from point $x_i \in X$ to centroid $v_j \in V$.

The fuzzy *c*-means clustering tries to find the best fit for a fixed value of *k*, the number of clusters. However, as an essential problem of clustering, determining an appropriate parameter *k* is a tough task. Even though prior knowledge of target classes is known, the number of clusters corresponding to the number of classes or subclasses is not always meaningful. The most common way to find the reasonable number of clusters is to run the algorithm with various values of $k \in \{2,\ldots, k_{max}\}$ and then employ a validity measure to evaluate cluster fitness. There are several validity measures proposed in the literature. Some of them are known as partition coefficient, classification entropy, partition index, separation index, Xie and Beni's index, and Dunn's index [1], [2], [15].

In our approach, however, the need is that data should be grouped in the way that reveals helpful information for inducers, not for the sense of clustering itself, even though the number of clusters might be wrong. In other words, using validity measures to determine the best number of clusters is not reliable enough to derive good cluster features for classifiers. In such a vision, instead of endeavoring to find the best *k* with validity measures, we use the *over-production* method to generate several candidate classifiers for different values of *k* and then evaluate their performance to determine the best one. Evaluating the performance of candidate classifiers can be based either on a validation set or Cross-Validation method. By such a way, a proper value of *k* is simultaneously found in the process of finding a maximum performance classifier from candidate classifiers.

In addition, the use of cluster features should be examined individually for a concrete inducing type. In our observation, two basic cluster features (*Z*, *B*) are benefic enough for DT inducer, instead of including *k* extended cluster features ($P_1,\ldots,P_k$). By contract, in the SVM context, it is applicable to employ either only the basic cluster features (*Z*, *B*) or all the cluster features (*Z*, *B*, $P_1,\ldots,P_k$) for building a

SVM classifier. Another solution that can be applied for any inducing type is to employ feature selection techniques (e.g., filter, wrapper) to pick out high merit features from both m initial input features and all (k+2) cluster features. The desire to such a way is that applying feature selection techniques on (m+k+2) features brings about a smaller but more qualitative feature subset than those only on m initial features. Here, note is that feature selection is simultaneously carried out in the process of building candidate classifiers. In a nutshell, formally, there are three possibilities to incorporate cluster features into the initial features (A_1, …, A_m), i.e., (A_1, …, A_m, Z, B), (A_1, …, A_m, Z, B, P_1, …, P_k), or *Feature Selection*(A_1, …, A_m, Z, B, P_1, …, P_k).

4.2 Algorithm

The proposed algorithm, namely *CFC*, is depicted by the pseudo-code from Figure 2. Related notations are indicated in Table 1. The key idea is that, for each clustering with different number of clusters ($k \in K$), the algorithm builds and valuates a candidate classifier from the training set manipulated with a given feature selection type, by q-fold cross validation. The resulting classifier is the one exhibiting maximum performance.

Table 1. Notations used to describe the algorithm in Figure 2.

Notation	Description
Ω	A set of c class labels $\{\omega_1, \ldots, \omega_c\}$.
$S=\{X,Y\}$	The original training set, where X comprises n data points (x_i) defined from m initial features $\{A_1,\ldots,A_m\}$ and Y contains class labels ($y_i \in \Omega$) of x_i.
I	A base inducer (e.g., SVM, Decision Tree).
K	A predefined integer set representing possible number of clusters.
ξ	A feature selection technique that returns a specific feature subset.
C_k	A candidate classifier resulting from a clustering with k fuzzy clusters.
C_{k^*}	The best classifier among $\|K\|$ candidate classifiers.
V^k	A $k \times m$ matrix of k centroids obtained from clustering X into k clusters.
V^{k^*}	A $k^* \times m$ matrix of k^* centroids, corresponding to C_{k^*}.
W^k	An $n \times k$ membership matrix of n data points $x_i \in X$, corresponding to V^k.
B^k	A column matrix containing the maximum membership weight of each $x_i \in X$.
Z^k	A column matrix representing the cluster (symbolic) number of each $x_i \in X$.
Θ	A horizontal concatenation operator between two matrices, say M_1 and M_2, having the same number of rows. $M_1 \Theta M_2$ results in a new matrix, say M, by horizontally joining M_2 to M_1. Columns in M are columns in M_1 plus columns in M_2. Matrix M has the same number of rows as M_1 and M_2.

In the training phase, the algorithm first normalizes continuous features (e.g., by a variance-based spread measure) to avoid the dispersion in different ranges (Line 1). Here, it is noticed that the normalized data (X') is merely for clustering purpose, whereas classifiers are built by using the original data (X). In addition, instead of executing clustering with parameter k ranging from 2 to a given *kmax* value, the algorithm uses a predefined set $K=\{k\}$ to mainly focus on important values of k,

which can be recognized by experiment or prior knowledge (Line 2). As mentioned in Subsection 4.1, there are three cases to select relevant features from the (m+k+2) features of the manipulated training set for building classifiers. Hence, for general purpose, the algorithm introduces an input parameter $\mathfrak{I}$ for specifying the way to select features from the manipulated training set (Lines 6-10). Subsequently, the algorithm builds and evaluates one candidate classifier for each clustering (Lines 11, 12). Here, note is that evaluating candidate classifiers is based on the averaged performance of q-fold stratified cross validation from the manipulated training set. Finally, the algorithm determines one best classifier from $|K|$ candidate classifiers, together with a corresponding centroid set (Lines 14, 15).

Training phase
Input: $S=\{X, Y\}$, I, K, ξ, and $\mathfrak{I}$
Output: C_{k^*}, V^{k^*}
1: $X' \leftarrow \text{Normalize}(X)$
2: *For* each $k \in K$ *do*
3: $\{W^k, V^k\} \leftarrow \text{FuzzyClustering}(X'_1, k)$
4: $B^k \leftarrow \{b_i \mid b_i = \max(w_{ij}), i = 1...n, j = 1...k\}$
5: $Z^k \leftarrow \{z_i \mid z_i = \arg\max_j (w_{ij}), i = 1...n, j = 1...k\}$
6: *Case* *// Choose features for classifier; D is a manipulated training set*
7: $\mathfrak{I} = 1$: $D \leftarrow (X \ \Theta \ Z^k \ \Theta \ B^k)$ *//Initial features and basic cluster features*
8: $\mathfrak{I} = 2$: $D \leftarrow (X \ \Theta \ Z^k \ \Theta \ B^k \ \Theta \ W^k)$ *//Initial features and all cluster features*
9: $\mathfrak{I} = 3$:
Begin
$F \leftarrow \xi(X \ \Theta \ Z^k \ \Theta \ B^k \ \Theta \ W^k, Y)$ *//Apply a feature selection technique*
$D \leftarrow (X \ \Theta \ Z^k \ \Theta \ B^k \ \Theta \ W^k)\,[F]$ *//Project data by the derived feature subset*
End
10: *End Case*
11: $C_k \leftarrow I(D,Y)$ *//Build a candidate classifier, using inducer I*
12: *Performance*$(C_k) \leftarrow$ (Averaged performance of q-fold cross validation based on (D,Y) and I)
13: *End For*
14: $C_{k^*} \leftarrow \arg\max_{C_k}$ *Performance*(C_k), $k \in K$ *//Determine one best classifier*
15: Return C_{k^*}, V^{k^*}
Operation phase
16: For a unlabeled testing instance x:
17: $x' \leftarrow \text{Normalize}(x)$
18: Compute cluster features for x', based on V^{k^*} (Formula 3)
19: Label x (with taking the derived cluster features into account) by C_{k^*}

Fig. 2. Algorithm *CFC*: generating a classifier from the initial features and cluster features.

As a whole, determining one best classifier from $|K|$ candidate classifiers can be grounded under different angles, e.g., focusing on hard classes rather than only basing on a generalization performance metric. For example, in the intrusion detection problem, both *R2L* attack and *Normal* classes exhibit similar behaviors and therefore are often misclassified to each other. In such a case, the accuracy ratio of *R2L* to *Normal* classes can be chosen as an additional trade-off threshold to evaluate classifier ability. Moreover, it is noted that heuristic-based clustering algorithms might produce different centroids for different runs with the same parameters. Hence, the resulting classifier should be determined by some runs of the proposed algorithm.

In the operation phase, for an unlabeled testing instance x, the algorithm first normalizes x in the same way as those applied to the training set. Then, cluster features of x are calculated based on the centroid set V^{k^*}, using Formula 3. Finally, such corresponding features are input to classifier C_{k^*} for final prediction.

5 Experiments

5.1 Dataset

Our experiments are conducted on the intrusion detection dataset, KDD99 [4]. This dataset was derived from the DARPA dataset, a format of TCPdump files captured from the simulation of normal and attack activities in the network environment of an air-force base, created by MIT's Lincoln Laboratory [5]. Although the DARPA dataset (and hence KDD99 dataset) has been criticized by the research community mainly because it is unrepresentative of a real-life network scenario [6, 7], so far it is still the only intrusion detection evaluation benchmark for research purposes.

The (*ten-percent*) KDD99 dataset comprises 494,021 training instances and 311,029 testing instances. Due to such a large data volume, the research community mostly uses small subsets of the dataset for evaluating IDS methods. Each instance in the dataset represents a network connection, i.e., a sequence of network packets starting and ending at some well defined times, between which data flows to and from a source IP address to a target IP address under some well defined protocol. Such a connection instance is described by a 41-dimensional feature vector and labeled with respect to five classes: *Normal*, *Probe*, *DoS* (denial of service), *R2L* (remote to local), and *U2R* (user to root).

To facilitate experiments without losing generality, we only use a smaller set of the KDD99 dataset for the purpose of evaluating and comparing our method to others. In particular, the training and testing sets used in our experiments are made up of 19,752 instances and 36,748 instances that are selectively extracted from the KDD99 training and testing sets, respectively. The principle for forming such reduced sets is to get all instances in each small group (attack type), but only a limited amount of instances in each large group, from both the KDD99 training and testing sets. More explicitly, we randomly select 3,000 instances in each large group *Neptune* and *smurf*, while gathering all instances in the remaining groups from both the KDD99 training and testing sets to form the reduced training and testing sets, respectively. For class *normal*, we randomly choose 5,000 instances from the KDD99 training set and 10,000

instances from the KDD99 testing set for the reduced training and testing sets, respectively. Class distribution of these two reduced sets is shown in Table 2.

Table 2. Class distribution of the reduced training and testing sets used in experiments.

Class	Training Set	Testing Set
DoS	9,467	13,761
Probe	4,107	4,163
R2L	1,126	8,751
U2R	52	70
Normal	5,000	10,000
Total	**19,752**	**36,748**

5.2 Experiment Setup

In our experiments, the predefined set K is set to {2, 3, …, 50}. In addition, the fuzzy c-means clustering is employed to generate cluster features. Convergence criterion (termination tolerance) of the fuzzy clustering is set to 10^{-6}, whereas the fuzzy degree (exponent α in Formula 3) is set to 4. On the other hand, continuous futures are normalized by *max_min* value ranges.

To handle different feature types as well as to express different merit contributions of features in the Euclidian space, we propose a distance measure as in Formula 4.

$$d(x_i, v_j)^2 = \sum_{q}^{m} G_q \times d_q(x_{iq}, v_{jq})^2 \tag{4}$$

where

$$d_q(x_{iq}, v_{jq}) = \begin{cases} 1, \left(x_{iq}, v_{jq} \in \{\text{symbolic}\}\right) \wedge (x_{iq} \neq v_{jq} \vee \left(x_{iq}, v_{jq} \in \{\text{unknown}\}\right) \\ |\, x_{iq} - v_{jq}\,|, \; x_{iq}, v_{jq} \in \{\text{continuous}\} \\ \dfrac{|\, x_{iq} - v_{jq}\,|}{t-1}, \; x_{iq}, v_{jq} \in \{\text{ordinals}\}, t = \left|\{\text{ordinals}\}\right| \\ 0, \text{ otherwise,} \end{cases}$$

and $$\begin{cases} G_q = Information\,Gain\,(\,feature\,q) = Entropy(S) - \sum\limits_{v \in Dom(q)} \dfrac{|\,S_v\,|}{|\,S\,|} Entropy(S_v) \\ Entropy(S) = \sum\limits_{\omega_j \in \Omega} -P_{\omega_j} \log_2 P_{\omega_j}, \;\; P_{\omega_j} \; is\; prior\; probability\; of\; class\; \omega_j \end{cases}$$

The base inducers (I) tested in our method are the C4.5 decision tree [8] and the support vector machine [20] with polynomial kernel and radial basic function kernel. The parametric setting of these inducers is specifically shown in Table 4.

The feature selection technique (ξ) used in this experiment is the correlation-based feature subset evaluation [19] and the genetic search [21]. This technique evaluates the merit of a feature subset by considering the individual predictive ability of each

feature along with the degree of redundancy between them. Those subsets that are highly correlated with the class while having low intercorrelation are preferred.

Candidate classifiers are evaluated by an attack type-based stratified cross validation (10 folds). The maximum performance classifier is determined based on *overall accuracy* (i.e., the ratio of the number of correctly classified instances to the total number of instances in the training set).

5.3 Experiment Results

The experimental results of our algorithm (implemented in Matlab) and others are shown in Tables 3 and Figure 3. Specifically, Table 4 presents True Positive (TP) and False Positive (FP) rates of classifiers with respect to each class label, whereas Figure 3 portrays TP rates of classifiers based on Table 4. Here, all compared classifiers are built from the same training set and tested on the same testing set as described in Subsection 5.1.

To have a wider comparative view, we run our algorithm (*CFC*) with different settings of two input parameters (i.e., *I*: base inducer, and $\Im$: the way to employ cluster features for classifiers). The results of such runs are listed in Rows 10-22 of Table 3. Here, it is mentioned that *stand-alone* DT and SVM (Rows 4, 6, 8) have the same settings as those parameterized to *CFC*.

Figure 3 shows that, as a whole, our method considerably outperforms the others with respect to TP rates in all five classes and on weighted average. Particularly, our method is significantly better than all the others in detecting hard classes (i.e., R2L and U2R). In addition, as shown in Table 3, FP rates of all the tested methods are virtually equivalent (i.e., approximately one percent).

It also noticed that the proposed method considerably improve the classification ability of base inducers (SVM and DT). More precisely, by using the same feature selection technique as described in Subsection 5.2, SVM classifier built from the manipulated training set (i.e., *CFC*(*I*=SVM, $\Im$=3)) is considerably superior to SVM classifier built from the original training set (i.e., SVM_FS). Similarly, *CFC*(*I*=DT, $\Im$=3) considerably outperforms DT_FS. This tells that applying a feature selection technique on the manipulated training set produces a higher qualitative feature subset (including base features and cluster features) than that on the original training set.

Regarding SVM classifiers, although we further test PSVM with exponent degrees ranging from 2 to 6, its performance remains worse than *CFC*(PSVM(degree=2), $\Im$={1,2,3}). More specifically, on average, *CFC*(PSVM(degree=2), $\Im$={1,2,3}) gives an 81.06% TP rate (with a 0.66% FP rate), whereas PSVM(degree={2, …, 6}) produces a 72.18% TP rate (with a 0.81% FP rate). This tells that cluster features benefit SVM in high dimensionality.

Table 3. True Possitive and False Possitive rates of classifiers.

Classifier		DoS	Probe	R2L	U2R	Normal	Average
1. Boosting	TP	94.82	81.19	16.84	22.86	97.16	70.04
	FP	0.62	0.63	0.93	0.31	0.44	1.87
2. Bagging	TP	97.34	81.74	12.32	25.71	96.25	69.87
	FP	0.29	0.33	0.32	0.41	0.54	1.25
3. NBTree	TP	94.49	82.24	14.26	21.43	95.47	69.1
	FP	0.12	0.26	0.88	0.11	0.97	2.25
4. DT	TP	95.83	81.07	10.27	18.57	97.08	68.54
	FP	0.85	0.44	0.86	0.76	0.63	2.06
5. DT_FS	TP	95.83	81.07	10.27	18.57	97.08	68.54
	FP	0.45	0.7	0.56	0.82	0.59	2.06
6. PSVM	TP	92.11	80.34	8.29	17.14	94.32	65.85
	FP	0.84	0.12	0.87	0.99	0.4	1.62
7. PSVM_FS	TP	92.11	80.34	8.29	17.14	94.32	65.85
	FP	0.34	0.82	0.42	0.41	0.4	1.62
8. RSVM	TP	90.32	83.54	9.34	15.71	93.18	65.66
	FP	0.25	0.27	0.31	0.22	0.31	0.89
9. RSVM_FS	TP	90.32	83.54	9.34	15.71	93.18	65.66
	FP	0.97	0.32	0.64	0.38	0.34	2.06
10. *CFC*(*I*=DT,ℑ=1)	TP	98.72	94.29	38.21	57.14	99.62	80.29
	FP	0.42	0.48	0.98	0.33	0.18	1.43
11. *CFC*(*I*=DT,ℑ=2)	TP	98.11	93.66	37.15	43.12	98.71	79.02
	FP	0.62	0.28	0.36	0.29	0.41	1.02
12. *CFC*(*I*=DT,ℑ=3)	TP	98.26	88.71	39.17	58.57	98.12	79.43
	FP	0.33	0.21	0.82	0.55	0.22	1.21
13. *CFC*(*I*=PSVM,ℑ=1)	TP	99.26	95.66	37.65	61.43	99.45	80.56
	FP	0.81	0.52	0.63	0.32	0.37	1.52
14. *CFC*(*I*=PSVM,ℑ=2)	TP	98.27	95.18	40.28	64.29	99.52	80.82
	FP	0.63	0.96	0.68	0.16	0.71	2.16
15. *CFC*(*I*=PSVM,ℑ=3)	TP	99.38	96.68	41.19	60	99.72	81.82
	FP	0.64	0.93	0.5	0.73	0.33	1.48
16. *CFC*(*I*=RSVM,ℑ=1)	TP	99.65	96.89	41.78	62.86	99.55	82.14
	FP	0.14	0.49	0.72	0.27	0.56	1.65
17. *CFC*(*I*=RSVM,ℑ=2)	TP	99.02	94.35	36.86	58.57	98.15	79.9
	FP	0.68	0.95	0.13	0.68	0.63	1.58
18. *CFC*(*I*=RSVM,ℑ=3)	TP	99.21	95.02	42.27	65.71	98.77	81.75
	FP	0.11	0.33	0.71	0.73	0.28	1.18

Note:
- DT: Decision Tree(pessimistic pruning, confidence=0.2, min(# instances per leaf)=7).
- Boosting: AdaBoost(DT, 10 classifiers) [22]
- Bagging: Bagging(DT, 10 classifiers) [23]
- PSVM: SVM with Polynomial Kernel (degree = 2).
- RSVM: SVM with Radial Basic Function Kernel (gama = 0.1).
- Classifier 5, 7, 9: trained on the original training set, with applying the feature selection technique as described in Subsection 5.2.
- Classifier 1-4, 6, 8: trained on the original training set, without applying feature selection.
- Classifier 10-18: built from our algorithm.

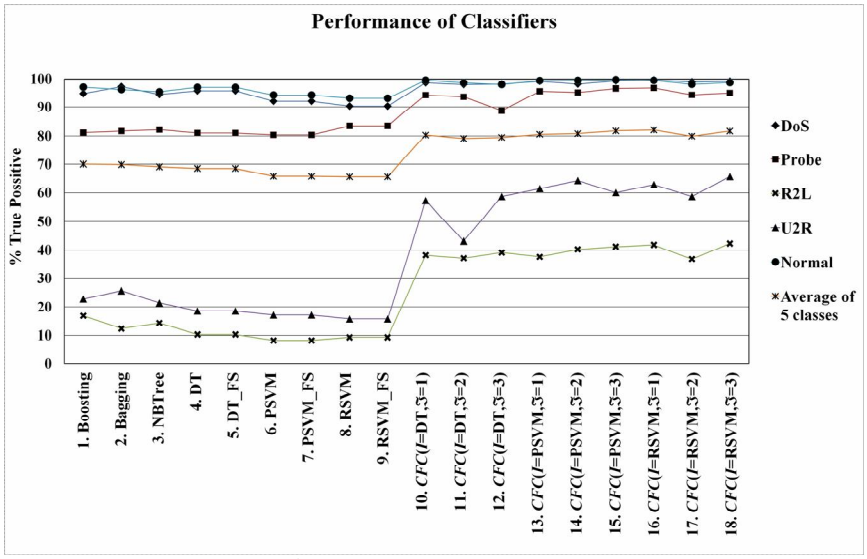

Fig. 3. Chart representing classifier comparisons

6 Conclusion and Future Work

This study presents an efficient method in applying data mining to the intrusion detection problem. The incorporation of cluster features resulting from a fuzzy clustering into the training process is proved to be efficient for enhancing the strength of a base classifier. The way to achieve a high performance classifier from a training set supplemented with cluster features is straightforwardly addressed. It is empirically shown that, as a whole, our method clearly outperforms several methods, when evaluated on the KDD99 dataset.

However, to be more objective in evaluating any data mining solution as well as overcome criticized drawbacks of the KDD99 intrusion detection dataset, our future work will be to test the proposed method on other real datasets. In particular, our current effort is fulfilling a honeypot system for the goal of gathering both real intrusion and normal traffic activities. Such a real dataset will then be employed to evaluate the method we proposed.

Finally, although the data mining domain is mature, the need to increase accuracy of classification models, in general, and IDSs, in specific, is still challenging to researchers. We believe that our attempt in this paper is a useful contribution to the research community.

References

[1] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York, 2 edition, 1987.

[2] F. Hoppner, F. Klawonn, R. Kruse, and T. Runkler. *Fuzzy Cluster Analysis*. Wiley, Chichester, 1999.

[3] A.M. Bensaid, L.O. Hall, J.C. Bezdek, L.P. Clarke, M.L. Silbiger, J.A. Arrington, and R.F. Murtagh. Validity-guided (Re) Clustering with applications to imige segmentation. *IEEE Transactions on Fuzzy Systems*, 4:112-123, 1996.

[4] UCI KDD ARCHIVE. 1999. KDD Cup 1999 Data. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[5] MIT LINCOLN LAB. DARPA intrusion detection data sets. Available:http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/index.html.

[6] J. McHugh. Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. *ACM Transactions on Information and System Security*, 3(4):262–294, Nov. 2000.

[7] G. Vigna, E. Jonsson, and C. Krügel, editors. *An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection*, volume 2820 of *Lecture Notes in Computer Science*. Springer, 2003.

[8] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[9] L. Portnoy, E. Eskin, and S. Stolfo. Intrusion detection with unlabeled data using clustering. *In Proceedings of ACM CSS Workshop on Data Mining Applied to Security*, 2001.

[10] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. *Applications of Data Mining in Computer Security*, 2002.

[11] K. Wang and S. Stolfo. Anomalous payload-based network intrusion detection. *In Proceedings of Recent Advance in Intrusion Detection* (RAID) (Sophia Antipolis, France, 2004), 203-222.

[12] G. Giacinto, R. Perdisci, and F. Roli. Network intrusion detection by combining one-class classifiers. *Image Analysis and Processing – ICIAP 2005*, LNCS 3617, 58-65, 2005.

[13] T. Shon and J. Moon. A hybrid machine learning approach to network anomaly detection. *Inf. Sci*, 177(18):3799–3821, 2007.

[14] C. Xiang, P. C. Yong, and L. S. Meng. Design of multiple-level hybrid classifier for intrusion detection system using bayesian clustering and decision trees. *Pattern Recognition Letters*, 29(7):918–924, 2008.

[15] O. Depren, M. Topallar, E. Anarim, and M. K. Ciliz. An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks. *Expert Syst. Appl*, 29(4):713–722, 2005.

[16] W. Fan, M. Miller, S. J. Stolfo, W. Lee, and P. K. Chan. Using artificial anomalies to detect unknown and known network intrusions. *Knowl. Inf. Syst*, 6(5):507–527, 2004.

[17] J. C. Bezdek. Pattern Recognition with Fuzzy Objective Function Algoritms. *Plenum Press*, New York, 1981.

[18] S. Y. Wu and E. Yen. Data mining-based intrusion detectors. *Expert Syst. Appl*, 36(3):5605–5612, 2009.

[19] M. A. Hall (1998). Correlation-based Feature Subset Selection for Machine Learning. Hamilton, New Zealand.

[20] J. Platt: Fast Training of Support Vector Machines using Sequential Minimal Optimization. In B. Schoelkopf and C. Burges and A. Smola, editors, Advances in Kernel Methods - Support Vector Learning, 1998.

[21] David E. Goldberg (1989). Genetic algorithms in search, optimization and machine learning. Addison-Wesley.

[22] Yoav Freund, Robert E. Schapire: Experiments with a new boosting algorithm. In: Thirteenth International Conference on Machine Learning, San Francisco, 148-156, 1996.

[23] Leo Breiman (1996). Bagging predictors. Machine Learning. 24(2):123-140.