Available online at www.sciencedirect.com
ScienceDirect
Computer Communications 30 (2007) 2201–2212
computer communications
www.elsevier.com/locate/comcom

A new approach to intrusion detection based on an evolutionary soft computing model using neuro-fuzzy classifiers ☆

Adel Nadjaran Toosi [a,*], Mohsen Kahani [b]

[a] Communication and Computer Research Laboratory, Faculty of Engineering, Ferdowsi University of Mashhad, Iran
[b] Department of Computer, Faculty of Engineering, Ferdowsi University of Mashhad, Iran

Received 8 October 2006; received in revised form 4 May 2007; accepted 11 May 2007
Available online 24 May 2007

Abstract

An intrusion detection system's main goal is to classify activities of a system into two major categories: normal and suspicious (intrusive) activities. Intrusion detection systems usually specify the type of attack or classify activities in some specific groups. The objective of this paper is to incorporate several soft computing techniques into the classifying system to detect and classify intrusions from normal behaviors based on the attack type in a computer network. Among the several soft computing paradigms, neuro-fuzzy networks, fuzzy inference approach and genetic algorithms are investigated in this work. A set of parallel neuro-fuzzy classifiers are used to do an initial classification. The fuzzy inference system would then be based on the outputs of neuro-fuzzy classifiers, making final decision of whether the current activity is normal or intrusive. Finally, in order to attain the best result, genetic algorithm optimizes the structure of our fuzzy decision engine. The experiments and evaluations of the proposed method were performed with the KDD Cup 99 intrusion detection dataset.

Keywords: Intrusion detection; Soft computing; Neuro-fuzzy; Fuzzy; Genetic algorithm; KDD Cup 99

1. Introduction

With the widespread use of computer networks, the number of attacks has grown extensively, and many new hacking tools and intrusive methods have appeared. Using an intrusion detection system (IDS) is one way of dealing with suspicious activities within a network.

An intrusion detection system monitors the activities of a given environment and decides whether these activities are malicious (intrusive) or legitimate (normal) based on system integrity, confidentiality and the availability of information resources. The intrusion detection system collects information about the system being observed. This collected audit data is processed by the detector. The detector eliminates unnecessary information from the audit data and then makes a decision to evaluate the probability that these activities can be considered as a sign of an intrusion [1,2].

Soft computing is an innovative approach to construct a computationally intelligent system which parallels the extraordinary ability of the human mind to reason and learn in an environment of uncertainty and imprecision [3]. Typically, soft computing consists of several computing paradigms, including neural networks, fuzzy sets, approximate reasoning, genetic algorithms, simulated annealing, etc.

Many soft computing approaches have been applied to the intrusion detection field [4–8]. In this paper, a novel intrusion detection system based on the integration of a few soft computing methods including neuro-fuzzy, fuzzy and genetic algorithms is described. The key contribution of this work is the utilization of outputs of neuro-fuzzy

☆ This work was partially supported by the Iran Telecommunication Research Center (ITRC).
* Corresponding author. Tel.: +98 915 5119823.
E-mail addresses: ad_na85@stu-mail.um.ac.ir (A.N. Toosi), kahani@um.ac.ir (M. Kahani).

0140-3664/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2007.05.002

network as linguistic variables which expresses how reliable current output is.

Fuzzy logic, as a robust soft computing method, has demonstrated its ability in intrusion detection systems [5,6,9–11]. Moreover, fuzzy systems have several important features which make them suitable for intrusion detection [9]. Most fuzzy systems make use of human expert knowledge to create their fuzzy rule base and hence lack adaptation, though. Therefore, building fuzzy systems with learning and adaptation capabilities has recently received much attention [11]. Various methods have been suggested for automatic generation and adjustment of fuzzy rules without using the aid of human experts; the neural fuzzy [12,13] and genetic fuzzy are two most successful approaches in this regard [14,15].

From the view point of classification, the main work of building an intrusion detection system is to build a classifier that can categorize normal and intrusive event data from the original dataset. ANFIS as an Adaptive neuro-fuzzy inference system [13] has the ability to construct models solely based on the target system sample data. This ability among others qualifies ANFIS as a fuzzy classifier for intrusion detection.

The proposed system has different layers which correspond to the needs in various modules of the proposed IDS system. First of all, several neuro-fuzzy classifiers use extracted features of the audit data to classify activities in the network. In this case fuzzy inference system as a decision-making engine based on outputs of the classifiers of previous layer makes the final decision on whether the current activity is normal or intrusive. Finally, genetic algorithms are employed to optimize the structure of fuzzy sets of the fuzzy decision-making engine.

In order to promote the comparison of different works in IDS area, the Lincoln Laboratory at MIT, under the Defense Advanced Research Project Agency (DARPA) and Air Force Research Laboratory (AFRL/SNHS) sponsorship, constructed and distributed the first standard dataset for evaluation of computer network IDS [16].

Afterward the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining with the purpose of demonstrating the learning contest, collected and generated TCP dump data provided by the aforementioned DARPA in the form of train-and-test sets whose features are defined for the connection records (a connection is a sequence of TCP packets starting and ending at some well-defined times). The main goal of the learning contest was to select classifiers with the best qualifications of recognizing normal and intrusive connections. The above dataset is named as KDD Cup 99 dataset [17] here, and has been used for the experiments.

The subsequent parts of this paper are organized as follows: At first, the related works done by the other researchers is briefly reviewed in Section 2. Section 3 describes KDD Cup 99 dataset on which the experiments are conducted. Then, the next section briefly outlines the basics of fuzzy inference systems and neuro-fuzzy concepts in general and ANFIS (Adaptive Neuro-Fuzzy Inference System) in particular. The last part of this section has been devoted to describing the subtractive clustering technique employed by ANFIS for automatic generation of the initial fuzzy inference system structure. Next, in Sections 5 and 6, the proposed system is explained and experimental results as well as evaluation of the proposed approach are discussed, respectively. Finally, Section 7 makes some concluding remarks and proposes further areas for future research.

2. Related work

There were a total of 24 entries submitted to the KDD Cup 99 contest. All the top three winners' approaches use some variants of decision trees. The KDD'99 contest winner entry made use of an ensemble of 50×10 C5 decision trees, using cost-sensitive bagged boosting [18]. The runner-up entry also used decision trees. A set of decision trees was constructed. Then a problem-specific global optimization criterion was used to select optimal subset of trees to give the final prediction [19]. The third-placed approach used two-layer decision trees. The first layer was trained on the connections which cannot be classified by security experts, whereas the second layer was built on the connections which cannot be classified by the first layer [20].

Thereafter, other approaches on the classification problem of KDD Cup 99 have emerged. One of the successful approaches based on data-mining framework used RIPPER rules which have been presented by Lee et al. [21]. Association rules and Frequent Episodes algorithms have been used to derive correlations between features and represent the sequentially of audit records, respectively. Agarwal and Joshi proposed a framework for learning a rule-based model (PNrule) to make classifier models on a dataset that has widely different class distributions in training data [22].

There are some works in which the performance of different machine learning algorithms and classification techniques were compared based on the KDD Cup 99 dataset. Sabhnani and Serpen analyzed the performance of comprehensive set of pattern recognition and machine learning algorithms according to the above dataset. Experiments outcomes show that a specific classification algorithm performs better for certain attack categories. The present fact was a motivation for the authors to use a multi-classifier model which utilizes different classifiers for each specific attack category of KDD dataset [23].

Recently, soft computing approaches are used for intrusion detection systems. Some of these methods have been evaluated on KDD dataset. Fuzzy rule-based classifiers, decision trees, support vector machines, linear genetic programming have been used in [8] by Abraham and Jain to illustrate the importance of soft computing paradigm for modeling intrusion detection systems. Abadeh et al. describe a fuzzy genetics-based learning algorithm and discuss its usage for intrusion detection in network [11]. Their experiments were performed on KDD dataset. Another

work which utilizes genetic algorithm for incorporating the capability of learning to fuzzy rules is the work of Gomez and Dasgupta [5]. Genetic programming based on RSS-DSS algorithm for dynamically filtering the dataset is another technique which exists in this area [4]. It's important to note that this model detects whether or not a record is intrusive not if attack records belong to a specific attack category.

Yeung and Chow proposed a novel detection approach using non-parametric density estimation based on Parzen-Window estimator with Gaussian kernels to build an anomaly intrusion detection system [25]. This model also only detects whether the current record is an intrusion or not.

It seems necessary to cite the works that criticize many aspects of the DARPA evaluation dataset [26,27]. McHugh [26], with respect to the collected traffic data by DARPA, criticizes the lack of statistical evidence of similarity to the typical Air Force network traffic, low traffic rates, relative uniform distribution of the four major attack categories, skewed distribution of victim hosts, and flat network topology. More detailed analysis of this dataset which is made by Mahoney and Chan confirms McHugh's criticism that the data is of statistically different characteristics from the real traffic. They also suggest an approach to mitigate the problem [27]. However, it is difficult to employ such solutions for the KDD Cup dataset. Moreover, since this work should be compared with other works in this area and certainly should be respectful to the experimental conditions of other compared works, the original KDD dataset have been used for the experiments.

3. KDD Cup 99 dataset

The KDD Cup 99 dataset includes a set of 41 features derived from each connection and a label which specifies the status of connection records as either normal or specific attack type. The list of these features can be found in Appendix A. These features had all forms of continuous, discrete, and symbolic, with significantly varying ranges falling in four categories [17]:

- The first category consists of the *intrinsic* features of a connection, which include the basic features of individual TCP connections. The duration of the connection, the type of the protocol (TCP, UDP, etc.), and network service (http, telnet, etc.) are some of the features.
- The *content* features within a connection suggested by domain knowledge are used to assess the payload of the original TCP packets, such as the number of failed login attempts.
- The *same host* features examine established connections in the past two seconds that have the same destination host as the current connection, and calculate the statistics related to the protocol behavior, service, etc.
- The *similar same service* features inspect the connections in the past two seconds that have the same service as the current connection.

Likewise, attacks fall into four main categories [17]:

- DoS (Denial of Service): making some computing or memory resources too busy to accept legitimate users access these resources.
- R2L (Remote to Local): unauthorized access from a remote machine in order to exploit machine's vulnerabilities.
- U2R (User to Root): unauthorized access to local superuser (root) privileges using system's susceptibility.
- Probe: host and port scans as precursors to other attacks. An attacker scans a network to gather information or find known vulnerabilities.

KDD dataset is divided into training and testing record sets. Total number of connection records in the training dataset is about 5 million records. This is too large for our purpose; as such, only concise training dataset of KDD, known as *10% training dataset*, was employed here. The distribution of normal and attack types of connection records in this subset have been summarized in Table 1.

As it can be seen in Table 1, sample distributions for different categories of attacks in training data differ significantly from each other. One of the main contributions of this work is to overcome this issue by using different classifier for each class of data.

The test data enjoys a different distribution. Moreover, the test data includes additional attack types not present in the training data which makes classifying more complicated. Table 2 summarizes the distribution of normal and attack types of connection records in the test dataset. And Table 3, based on major types of attack, shows the sample distribution of the new attacks in the test dataset. New attacks refer to those which were not present in the training dataset, but exist in the test dataset.

Table 1
The sample distributions on the subset of 10% data of KDD Cup 99 dataset

Class	Number of samples	Samples percent (%)
Normal	97277	19.69
Probe	4107	0.83
DoS	391458	79.24
U2R	52	0.01
R2L	1126	0.23
	492021	100

Table 2
The sample distributions on the test data with the corrected labels of KDD Cup 99 dataset

Class	Number of samples	Samples percent (%)
Normal	60593	19.48
Probe	4166	1.34
DoS	229853	73.90
U2R	228	0.07
R2L	16189	5.20
	311029	100

Table 3
The new attacks sample distributions on the test data with the corrected labels of KDD Cup 99 dataset

Class	Number of novel attack samples	Total number of samples	Samples percent (%)
Probe	1789	4166	43
DoS	6555	229853	3
U2R	189	228	83
R2L	10196	16189	63
	18729	250436	7.5

4. Fuzzy and neuro-fuzzy

4.1. Fuzzy inference system (FIS)

The past few years have witnessed a rapid growth in the number and variety of applications of fuzzy logic. Among various combinations of methodologies in soft computing, the one that has the highest visibility is that of fuzzy logic and neurocomputing, leading to so-called neuro-fuzzy systems. An effective method developed by Jang for this purpose is called ANFIS (Adaptive neuro-fuzzy inference system) [13].

The basic structure of most Fuzzy inference systems (FISs) that we have seen so far is a model that maps the input characteristics to the input Membership functions (MF). Three well-known types of FIS are employed in various systems. The *Mamdani Fuzzy Model* [24] was proposed as the very first attempt to map an input to an output space on top of the experiences of experts.

An example of two-input single-output Mamdani fuzzy model with two rules can be expressed as

if x *is* $A1$ and y is $B1$ then z is $C1$,

if x is $A2$ and y is $B2$ then z is $C2$,

where A and B are fuzzy sets of inputs with membership functions of $A1$, $A2$ and $B1$, $B2$, respectively, and C is the fuzzy output set.

Max and min as the choice for T-norm and T-conorm operator are adopted here, respectively. The resulting fuzzy reasoning is shown in Fig. 1. For more acquaintance with T-norm and T-conorm, and inference system of Mamdani fuzzy models the readers may refer to [13].

Since usual systems take only crisp values, we should use a defuzzifier to convert a fuzzy set to a crisp value. (Defuzzification refers to the way a crisp value is extracted from a fuzzy set as a representative value [13].) We use centroid of area defuzzification strategy to convert the output to a crisp value. An explanation of centroid of area defuzzification strategy is shown below.

Centroid of area Z_{COA} is:

$$Z_{\mathrm{COA}} = \frac{\int_Z \mu_{\mathrm{A}}(z) z \mathrm{d}z}{\int_Z \mu_{\mathrm{A}}(z) \mathrm{d}z}, \tag{1}$$

where $\mu_{\mathrm{A}}(z)$ is the aggregated output MF.

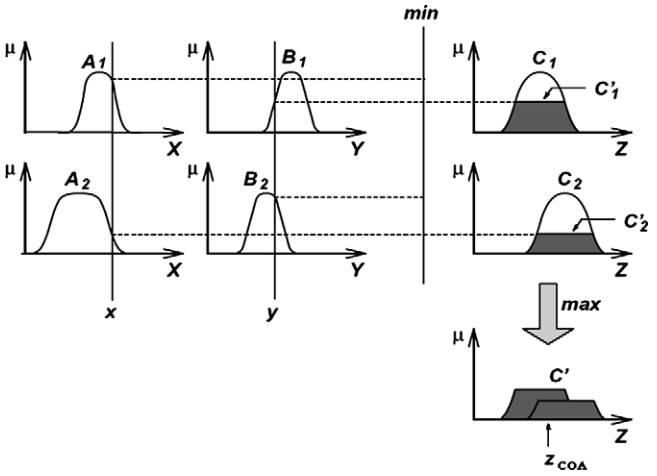

Fig. 1. The Mamdani fuzzy inference system using min and max for T-norm and T-conorm operators, respectively [13].

Before introducing the structure of ANFIS as our main classifier, it is important to mention that Mamdani fuzzy inference system (FIS) has been used for the final decision-making module. More details on structure of the system and decision-making engine will be explored at later sections.

In an effort to develop a systematic approach to generate fuzzy rules from a given input–output dataset, Takagi, Sugeno, and Kang proposed *TSK Fuzzy Model* (known as the *Sugeno Fuzzy Model*) [28]. A fuzzy rule in a Sugeno fuzzy model has the form of,

if x is A and y is B then $z = f(x, y)$,

where A and B are input fuzzy sets in antecedent and usually $z = f(x, y)$ is a zero- or first-order polynomial function in the consequent.

Fuzzy reasoning procedure for the first order Sugeno fuzzy model is shown in Fig. 2a. Here, defuzzification procedure in the Mamdani fuzzy model is replaced by the operation of weighted average in order to avoid the time-consuming procedure of the former [13].

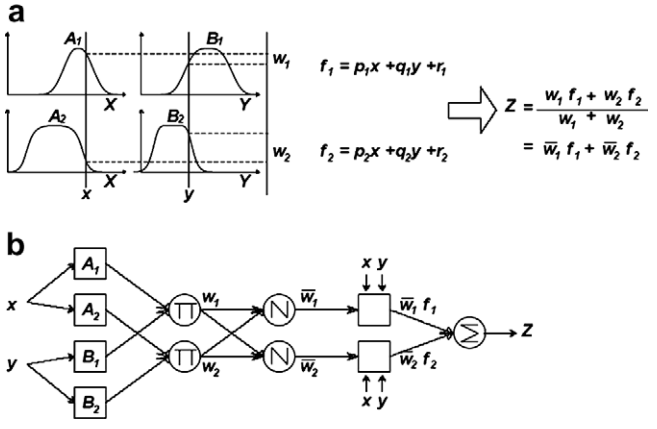

Fig. 2. (a) The Sugeno fuzzy model reasoning; (b) equivalent ANFIS structure [13].

4.2. Adaptive neuro-fuzzy inference system (ANFIS)

There are some modeling situations in which one cannot just look at the data and decides on the shape of membership functions. Rather than choosing the parameters associated with a given membership function arbitrarily, these parameters could be chosen so that they tailor the membership functions to the variation in the input/output data in order to account for these types of variations in the data values. This is where the so-called neuro-adaptive learning technique incorporated into ANFIS can help.

Assume a Fuzzy inference system with two inputs x, y and one output z with the first order of Sugeno Fuzzy Model. Fuzzy rule set with two fuzzy if–then rules is as follows:

$$\text{if } x \text{ is } A1 \text{ and } y \text{ is } B1, \text{ then } f1 = p1x + q1y + r1,$$
$$\text{if } x \text{ is } A2 \text{ and } y \text{ is } B2, \text{ then } f2 = p2x + q2y + r2,$$

Fig. 2a illustrates the reasoning mechanism for this Sugeno model.

As it is shown in Fig. 2b, the reasoning mechanism can be implemented into a feed-forward neural network with supervised learning capability, which is known as ANFIS architecture.

The square and circle nodes are for adaptive nodes with parameters and fixed nodes without parameters, respectively. The first layer consists of square nodes that perform fuzzification with chosen membership function. The parameters in this layer are called premise parameters. In the second layer, the T-norm operation is performed to produce the firing strength of each rule. The ratio of the ith rule firing strength to the sum of all rules' firing strength is calculated in the third layer, generating the normalized firing strengths. The fourth layer consists of square nodes that perform multiplication of normalized firing strengths with the corresponding rule. The parameters in this layer are called consequent parameters. The overall output is calculated by the sum of all incoming signals in the fifth layer [13].

ANFIS provides a method for the fuzzy modeling procedure to learn information about a dataset in order to compute the membership function parameters that best allow the associated Fuzzy inference system to track the given input/output data. This learning method works similarly to that of neural networks. The parameters associated with the membership functions will change through the learning process. ANFIS uses either back propagation or a combination of least square estimations and back propagation for membership function parameter estimations. The readers are referred to [13] for more details on these methods.

4.3. Subtractive clustering

The purpose of clustering is to identify natural groupings of data from a large dataset to produce a concise representation of a system's behavior. It is possible to use the cluster information to generate a Sugeno-type fuzzy inference system that best models the data behavior using a minimum number of rules. The rules partition themselves according to the fuzzy qualities associated with each of the data clusters.

Assume a 2-D training dataset (including input and desired output) and cluster center (x_i, y_i). The ith rule can be expressed in the form of

$$\text{if } X \text{ is close to } x_i, \text{ then } Y \text{ is close to } y_i.$$

After the structure is determined, back propagation or gradient decent and other optimization schemes can be applied to proceed with parameter identification.

However, before the start of the ANFIS training, the fuzzy inference system should be generated. FIS generation can implement in grid partitioning or subtractive clustering. In grid partitioning, all the possible rules are generated based on the number of MFs for each input. For example, in a two dimensional input space with three MFs in the input sets, the number of rules in grid partitioning results in 9 rules. This partitioning strategy needs only a small number of MFs for each input and encounters problems when we have moderately a large number of each input. So we use subtractive clustering to determine the number of rules, and the initial points of the membership functions.

Suppose that there is not a clear idea of how many clusters there should be for a given set of data. Subtractive clustering [29] is a fast one-pass algorithm for estimating the number of clusters and the cluster centers in a set of data. This method is used here, and it is an extension of the Mountain clustering method proposed by Yager and Filev [30].

Consider a collection of m data points $\{x_1, \ldots, x_m\}$ in an N-dimensional space. Subtractive clustering assumes each data point as a potential cluster center and calculates a measure of the potential for each data point based on the density of surrounding data points. Density measure at data point x_j is calculated as follows:

$$D_j = \sum_{i=1}^{m} \exp\left(-\frac{|x_j - x_i|^2}{(r_a/2)^2}\right), \tag{2}$$

where r_a is a positive constant value and it defines the neighborhood radius. The algorithm selects the data point with the highest density measure as the first cluster center and then eradicates the potential of data points near the first cluster center. The algorithm then selects the data point with the highest remaining potential (next highest density measure has been remained) as the next cluster center and eradicates the potential of data points near this new cluster center. This process of acquiring a new cluster center and eradicating the potential of surrounding data points repeats until the potential of all data points fall below a threshold. The range of influence of a cluster center in each of the data dimensions is called cluster radius. The cluster radius indicates the range of influence of a cluster when you consider the data space as a single hypercube. A small cluster radius will lead to find many small clusters in the data (resulting in many rules) and vice versa.

The clusters' information obtained by this method is used for determining the initial number of rules and

antecedent membership functions, which is used for identifying the FIS. An important advantage of using a clustering method to find rules is that the resulting rules are more tailored to the input data than they are in an FIS generated without clustering. In this study, we use Subtractive clustering has been used to determine the number of rules and antecedent membership functions. So one can obtain a FIS structure that contains a set of fuzzy rules to cover the feature space.

5. Proposed system

The principle motivation for this work was to provide a framework for using soft computing approaches to build a classifier that can act better than single algorithm using a single soft computing approach, e.g., neuro-fuzzy. The proposed system is discussed in details in this section. First, the system architecture is explained. Then, data sources, selected from KDD for training the system, are introduced. Afterward, layers of proposed framework are presented in more details.

5.1. System architecture

The proposed architecture for the Evolutionary Soft Computing Intrusion Detection System includes two layers. In the first layer, there are five ANFIS modules which are trained to explore the intrusive activity from the input data. Each ANFIS module belongs to one of the classes in the dataset each providing an output which specifies the degree of relativity of the data to the specific class 1 shows total membership while −1 is used otherwise. (It is important to mention that the ANFIS structure has only one output.) The most important motivation to using ANFIS in this way is that ANFIS is usually more appropriate as a binary classifier rather than a multi-classifier [31].

Second, a Fuzzy Inference module, based on empirical knowledge, is employed to make the final decision for recognition. The fuzzy inference module implements nonlinear mappings from the outputs of the neuro-fuzzy classifiers of the pervious layer to the final output space which specifies if the input data are normal or intrusive. Afterward, if the system recognizes that the current pattern is intrusive by nature, the classifier of the first layer, in which the output is the nearest value among all classifiers, specifies the class of the attack.

In order to attain the best results, genetic algorithm (GA) is used to optimize the structure of the fuzzy decision-making engine. The GA structure is discussed in more depth later. Fig. 3 depicts the schematic block diagram of the proposed system architecture.

5.2. The data sources

All of the above features have been applied to the inputs of the five neuro-fuzzy classifiers. From the classification point of view, any system mainly consists of two phases: (1) the training of the parameters of the classifier according to the training dataset and (2) using the classifier to categorize a test dataset. Here, 10% of the training dataset was used as the source of the training dataset. Since the number of records in the 10% dataset was still very large for our purposes, different subsets of the training and checking dataset were randomly selected from the subset of 10% of data, for the training phase. The basic idea behind using a checking dataset for model validation is that after a certain point in training, the model begins overfitting the training dataset. If overfitting does occur, we cannot expect the classifier to respond well to other independent datasets. In fact, if checking data is used for ANFIS training, the final FIS associated with the minimum checking error will be chosen.

Results of different machine learning algorithms show that anomaly detectors do better than signature-based detectors for KDD Cup 99 dataset [33]. This might be because the testing data has substantial new attacks with signatures not correlated with similar attacks in the training data. On the other hand, the number of training samples for signature-based detectors seems not to be ample

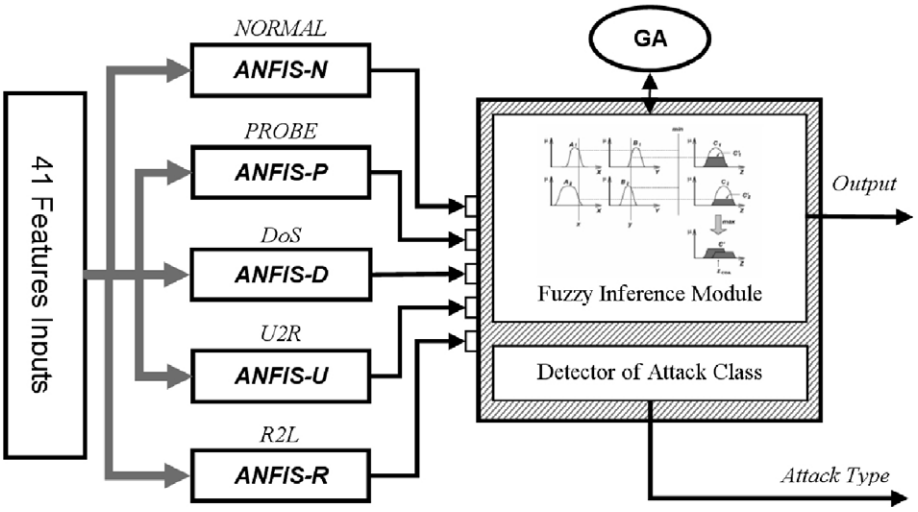

Fig. 3. System architecture block diagram.

to develop classifiers to function as efficiently as possible. The attack samples in the testing dataset, though, have rather enough deviation from normal or regular samples in the training dataset [4,25].

Since each classifier in first layer of the system acts as a signature based classifiers and the goal is to select a good training and checking dataset for the learning phase, training and checking dataset has been selected, as shown on the Tables 4 and 5, wherein numbers of samples in the normal class are approximately equal to the summation of the samples in the other classes. By this policy, in view of the fact that each classifier performs as binary classifier (current activity belongs to this class or not), each classifier somehow acts as an anomaly detector system.

The distribution of the samples in the two subsets that were used for the training is listed on Tables 4 and 5. Selected subsets enjoy different numbers of samples, the smaller one contains a few number of samples to show the system is still capable, despite the fact that a small portion of the training data has been used. The other one in more number of samples enjoys more number of samples to illustrate efficiency of proposed system as much as possible.

Due to the reduction of random sampling effects, 10 trails with the same distribution, have been selected for each subset of trainings (training sets in Tables 4 and 5). Therefore, all the evaluation results in the latter parts of the paper have been computed over these ten trials except those explicitly mentioned.

Table 4
Sample distributions on the *First Training and Checking* data randomly selected of 10% data of KDD Cup 99 dataset

		Normal	Probe	DoS	U2R	R2L	Total
ANFIS-N	Training	20000	4000	15000	40	1000	40040
	Checking	2500	107	2000	12	126	7245
ANFIS-P	Training	10000	4000	5000	40	1000	16040
	Checking	1000	107	500	12	126	10245
ANFIS-D	Training	25000	4000	20000	40	1000	45040
	Checking	6000	107	5000	12	126	10254
ANFIS-U	Training	200	50	50	46	50	246
	Checking	100	25	25	6	25	181
ANFIS-R	Training	4000	1000	2000	40	1000	6040
	Checking	2000	500	1000	12	126	3138

Table 5
Distribution of samples on the *Second Training and Checking* data randomly selected of 10% data of KDD Cup 99 dataset

		Normal	Probe	DoS	U2R	R2L	Total
ANFIS-N	Training	1500	500	500	52	500	3052
	Checking	1500	500	500	0	500	3000
ANFIS-P	Training	1500	500	500	52	500	3052
	Checking	1500	500	500	0	500	3000
ANFIS-D	Training	1500	500	500	52	500	3052
	Checking	1500	500	500	0	500	3000
ANFIS-U	Training	1500	500	500	46	500	3046
	Checking	1500	500	500	6	500	3006
ANFIS-R	Training	1500	500	500	52	500	3052
	Checking	1500	500	500	0	500	3000

Before concluding this subsection, it should be mentioned that to be fair, we did not have any access to the testing dataset during the training and optimization phase. Moreover, the standard conditions of the KDD Cup competition has been deployed.

5.3. The neuro-fuzzy classifiers

The subtractive clustering method with $r_a = 0.5$ (neighborhood radius) has been used to partition the training sets and generate an FIS structure for each ANFIS. For further fine-tuning and adaptation of membership functions, training sets were used for training ANFIS. Each ANFIS trains at 50 epochs of learning and final FIS that is associated with the minimum checking error has been chosen. All the MFs of the input and output fuzzy sets were selected in the form of Gaussian functions with two parameters.

5.4. The fuzzy decision module

The fuzzy inference module has five inputs, obtained from the output values of each ANFIS classifiers. The fuzzy inference module, based on these inputs, determines whether the current connection record is an attack or not. A five-input, single-output of Mamdani fuzzy inference system with centroid of area defuzzification strategy was used for this purpose. Each input fuzzy set includes two MFs and all the MFs are Gaussian functions which are specified by four parameters. The proposed fuzzy inference module uses the rules shown in the fuzzy associative memory in Table 6.

The output of the fuzzy inference engine, which varies between -1 and 1, specifies how intrusive the current record is, 1 to show completely intrusive and -1 for completely normal. Records with positive intrusive values are selected as intrusive patterns. After an attack is detected, its class is selected based on the ANFIS module class which returns the highest value.

5.5. The Genetic algorithm module

Genetic algorithm is a method for solving optimization problems that are based on natural selection – a process that derives from biological evolution [32]. The genetic algorithm repeatedly modifies a population (a set of individuals) by a set of genetic operators including mutation, crossover, and selection. It selects individuals evolving

Table 6
Fuzzy associative memory for the proposed fuzzy inference rules

Normal	PROBE	DoS	U2R	R2L	Output
High	–	–	–	–	Normal
–	¬High	¬High	¬High	¬High	Normal
–	High	–	–	–	Attack
–	–	High	–	–	Attack
–	–	–	High	–	Attack
–	–	–	–	High	Attack
Low	–	–	–	–	Attack
–	Low	Low	Low	Low	Normal

toward an optimal solution from the current population and uses them to produce children of the next generation. The algorithm stops when the stopping criterion is met. In the proposed system, each individual (chromosome) has genes codifying parameters of the MFs of the input fuzzy set of the fuzzy decision engine. A chromosome consists of 320 bits of binary data. Each 8 bits of a chromosome determines one parameter out of the four parameters of an MF. Fig. 4 illustrates the decoding process of each individual chromosome.

The genetic algorithm, which is used here to optimize the input MFs of the fuzzy decision-making module, uses a subset selected from 10% of KDD dataset for the optimization process. The distribution of samples for this subset is shown in Table 7.

In view of the fact that GA optimization process does not always provide an identical, the optimization phase was performed three times and the average of the experiments results was computed for each attained structures. Also, due to the reduction of the effects of randomly sampling, five different trails of subsets – not overlapping with each other – have been used for this phase.

The fitness function evaluates the fitness value for each individual. Fundamentally, the fitness function is the function that should be optimized. This works considers two different fitness functions.

Before discussing more about the fitness functions, it seems necessary to talk about standard metrics that has been developed for evaluating network intrusion detections. *Detection rate* and *false alarm rate* are the two most famous metrics that have already been used. Detection rate is computed as the ratio between the number of correctly detected attacks and the total number of attacks, while false alarm (false positive) rate is computed as the ratio between the number of normal connections that is incorrectly misclassified as attacks and the total number of normal connections. Another metric used here is the *classification rate*. Classification rate for each class of data is defined as the ratio between the number of test instances correctly classified and the total number of test instances of this class.

For the purpose of classifier algorithm evaluation, another comparative measure is defined which is *Cost Per Example* (CPE) [23].

CPE is calculated using the following formula:

$$\mathrm{CPE} = \frac{1}{N}\sum_{i=1}^{m}\sum_{j=1}^{m}\mathrm{CM}(i,j) * C(i,j), \tag{3}$$

where CM and C are confusion matrix and Cost Matrix, respectively, and N represents the total number of test instances, m is the number of the classes in classification. A confusion matrix is a square matrix in which each column corresponds to the predicted class, while rows correspond to the actual classes. An entry at row i and column j, $\mathrm{CM}(i,j)$, represents the number of misclassified instances that originally belong to class i, although incorrectly identified as a member of class j. The entries of the primary diagonal, $\mathrm{CM}(i,i)$, stand for the number of properly detected instances. Cost Matrix is similarly defined, as well, and entry $C(i,j)$ represents the cost penalty for misclassifying an instance belonging to class i into class j.

Cost Matrix values employed for the KDD'99 classifier learning contest are shown in Table 8a [17]. Lower values for cost per example measure show better classification for the intrusion detection system.

5.5.1. Fitness functions

This work considers two different fitness functions. The First fitness function considered here, represents the base-

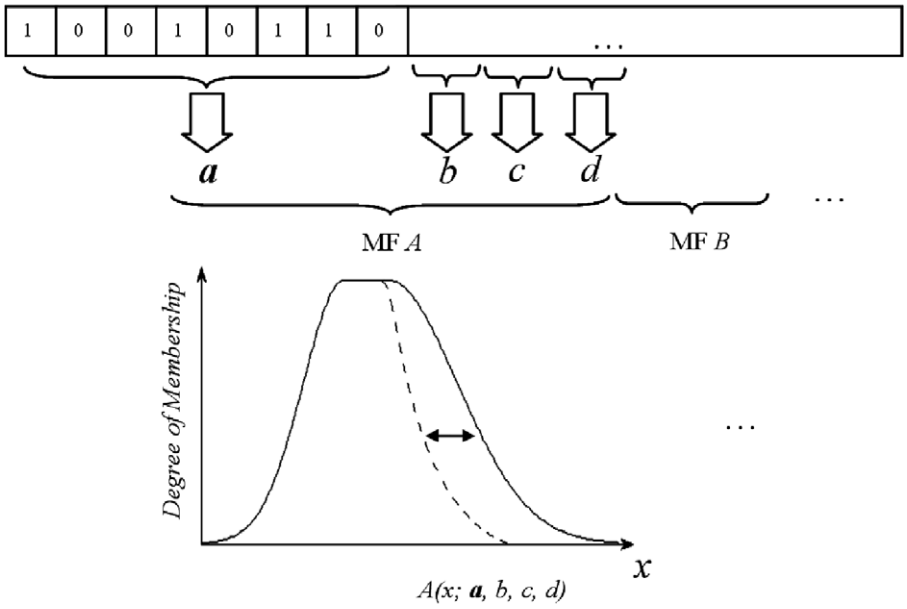

Fig. 4. Schematic decoding process of the individual chromosome.

Table 7
The sample distributions on the selected subset of 10% data of KDD Cup 99 dataset for the optimization process which used by GA

	Normal	Probe	DoS	U2R	R2L
Number of samples	200	104	200	52	104

Table 8
Characteristics of Cost Matrix; the columns correspond to predicted classes, rows correspond to actual classes

		Predicted				
		Normal	PROBE	DoS	U2R	R2L
(a) Actual	Normal	0	1	2	2	2
	PROBE	1	0	2	2	2
	DoS	2	1	0	2	2
	U2R	3	2	2	0	2
	R2L	4	2	2	2	0
(b) Actual	Normal	0	1	1	1	1
	PROBE	1	0	1	1	1
	DoS	1	1	0	1	1
	U2R	1	1	1	0	1
	R2L	1	1	1	1	0

(a) Cost Matrix values for the KDD'99 classifiers learning contest. (b) Cost Matrix values with equal misclassification costs.

Table 9
Abbreviations used for our approaches

Abbreviation	Approach
ESC-KDD-1	First Training set with fitness function of KDD
ESC-EQU-1	First Training set with fitness function of equal misclassification cost
ESC-KDD-2	Second Training set with fitness function of KDD
ESC-EQU-2	Second Training set with fitness function of equal misclassification cost

line case in which a Cost Per Example with equal misclassification costs (Table 8b) is employed. The genetic algorithm used to minimize the cost per examples is calculated in this way. Using the mentioned fitness function resolves the trade-off between detection rate and false alarm rate and leads to maximizing the overall detection rate and classification rate with low false alarm rate.

Another fitness function is employed based on the cost per examples used for evaluating results of the KDD'99 competition [17]. Using the Cost Matrix values employed for the KDD'99 classifier learning contest attained the best classification rate with respect to weighed misclassification cost.

6. Results

All samples of correctly labeled test dataset of KDD Cup 99 dataset (Table 2) as the testing data to evaluate the classifiers.

Before discussing the result, it should be mentioned that to perform the experiments, the structures obtained from 10 subsets of training data for both series were used for the classifiers. The genetic algorithm was performed three times, each time for one of the five series of selected subsets. Totatally, 150 different structures were used and the result is the average of the results of this 150 structures.

In the rest of this section, the performance of the proposed Evolutionary Soft Computing Intrusion Detection System (ESC-IDS) using two different training datasets (Tables 4 and 5) and two different fitness functions is compared. Two different training datasets for training the classifiers and two different fitness functions to optimize the fuzzy decision-making module were used. Table 9 shows the notation used for the special versions of ESC-IDS.

Table 10 shows results for the different versions of ESC-IDS on the test dataset with corrected labels of KDD Cup 99 dataset. Considerable outcomes can be seen on the third and fourth rows of the table. These statistics obtained from the structures which have been built on the second training set. This training set contains about 30,000 patterns, some of them are repeated and the whole is far less than total number of samples in the original training dataset. however, the results still demonstrate reasonable values. The variances of each averaged value in Table 10 has been shown on Table 11. Also, as an example, the confusion matrix of one out of the 150 obtained structures is shown in Table 12, which can be helpful in understanding the bias of the proposed classifier towards a particular class of attacks.

The performance of the ESC-IDS has been compared with some other machine learning methods tested on the KDD dataset and is shown in Table 13. The proposed method demonstrates better performances in a number of attacks categories and an unprecedented cost per examples of 0.1579. Based on the results shown in the Table 13, it can be easily seen that the proposed approach has a good performance for detecting intrusion in computer networks. Also, this method is flexible and can be adjusted for special situations using different fitness functions.

It should be noted that some values of Table 13 can be misleading. For example, Parzen-Window [25] algorithm detects only whether a record is intrusive or not and does not specify the attack category. Also, the authors did not

Table 10
Classification rate, Detection rate (DTR), False Alarm rate (FA) and Cost Per Example of KDD(CPE) for the different approaches of ESC-IDS on the test dataset with corrected labels of KDD Cup 99 dataset

Model	Normal	Probe	DoS	U2R	R2L	DTR	FA	CPE
ESC-KDD-1	98.2	84.1	99.5	14.1	31.5	95.3	1.9	0.1579
ESC-EQU-1	98.4	89.2	99.5	12.8	27.3	95.3	1.6	0.1687
ESC-KDD-2	96.5	79.2	96.8	8.3	13.4	91.6	3.4	0.2423
ESC-EQU-2	96.9	79.1	96.3	8.2	13.1	88.1	3.2	0.2493

Table 11
Variance classification rate, detection rate (DTR), False alarm rate (FA) and Cost Per Example of KDD (CPE) for the different approaches of ESC-IDS on the test dataset with corrected labels of KDD Cup 99 dataset

Model	Normal	Probe	DoS	U2R	R2L	DTR	FA	CPE
ESC-KDD-1	1.23E-4	11.74E-4	0.08E-4	5.61E-4	11.29E-4	0.10E-5	1.23E-4	1.29E-4
ESC-EQU-1	1.04E-4	26.15E-4	0.09E-4	8.19E-4	31.74E-4	0.16E-4	1.04E-4	2.84E-4
ESC-KDD-2	1.1850	15.7244	0.0578	2.6384	0.1309	0.7142	1.1850	2.01E-5
ESC-EQU-2	2.1679	25.9518	4.6407	2.8281	1.0419	4.4325	2.1679	1.04E-3

Table 12
Confusion Matrix for example obtained structure

		Predicted					Accuracy
		Normal	PROBE	DoS	U2R	R2L	
Actual	Normal	58809	478	251	774	281	98.47%
	PROBE	196	3541	276	49	104	84.97%
	DoS	534	49	228524	641	105	99.76%
	U2R	85	64	24	29	26	16.67%
	R2L	10698	22	17	56	5396	31.68%
False positive		16.37%	14.76%	0.25%	98.13%	8.73%	CPE = 0.1549

Table 13
Classification rate, Detection rate (DTR), False alarm rate (FA) and Cost Per Example of KDD (CPE) for the different algorithms performances on the test dataset with corrected labels of KDD Cup 99 dataset (n/r stands for not reported)

Model	Normal	Probe	DoS	U2R	R2L	DTR	FA	CPE
ESC-IDS	98.2	84.1	99.5	14.1	31.5	95.3	1.9	0.1579
RSS-DSS [4]	96.5	86.8	99.7	76.3	12.4	94.4	3.5	n/r
Parzen-Window [25]	97.4	99.2	96.7	93.6	31.2	n/r	n/r	0.2024
Multi-classifier [23]	n/r	88.7	97.3	29.8	9.6	n/r	n/r	0.2285
Winner of KDD [18]	99.5	83.3	97.1	13.2	8.4	91.8	0.6	0.2331
Runner Up of KDD [19]	99.4	84.5	97.5	11.8	7.3	91.5	0.6	0.2356
PNrule [22]	99.5	73.2	96.9	6.6	10.7	91.1	0.4	0.2371

report any information regarding the false alarm rates. Also, Parzen-Window and RSS-DSS are anomaly detection methods that only detect if a connection record is intrusive or not, and do not have any information regarding the attack type.

For systems that do not classify intrusions, correct classification concept is different from others. In classifying system, while a record has been corrected recognized as an intrusion, misclassification is considered as an error. Looking at Table 13 shows that the proposed system has correctly identified an intrusive record, while might has had problem classifying it.

It can be stated that all the machine learning algorithms tested on the KDD'99 dataset offered an acceptable level of detection performance only for DoS and PROBE attack categories and demonstrated poor performance on the U2R and R2L categories [33]. The proposed method shows improvement in these two classes (U2R and R2L).

7. Conclusions

In this paper, an evolutionary soft computing approach for intrusion detection was introduced and was successfully demonstrated its usefulness on the training and testing subset of KDD Cup 99 dataset. The ANFIS network was used as a neuro-fuzzy classifier for intrusion detection. ANFIS is capable of producing fuzzy rules without the aid of human experts. Also, subtractive clustering has been utilized to determine the number of rules and membership functions with their initial locations for better classification.

A fuzzy decision-making engine was developed to make the system more powerful for attack detection, using the fuzzy inference approach. At last, this paper proposed a method to use genetic algorithms to optimize the fuzzy decision-making engine. Experimentation results showed that the proposed method is effective in detecting various intrusions in computer networks.

Our future work will focus on reducing features for the classifiers by methods of feature selection. Also, the work will be continued to study the fitness function of the genetic algorithm to manipulate more parameters of the fuzzy inference module, even concentrating on fuzzy rules themselves.

Acknowledgements

The first author thank FUM Communication and Computer Research Lab. for their in-kind support and encouragement during this research. He also wishes to express his appreciations to the helpful suggestions and comments of dear colleagues E. Bagheri and M. Hosaini.

Appendix A. Lists of features in KDD Cup 99 dataset

Feature name	Type	Description
1. duration	continuous	length (number of seconds) of the connection
2. protocol_type	discrete	type of the protocol, e.g., tcp, udp, etc.
3. service	discrete	network service on the destination, e.g., http, telnet, etc.
4. src_bytes	continuous	number of data bytes from source to destination
5. dst_bytes	continuous	number of data bytes from destination to source
6. flag	discrete	normal or error status of the connection
7. land	discrete	1 if connection is from/to the same host/port; 0 otherwise
8. wrong_fragment	continuous	number of "wrong" fragments
9. urgent	continuous	number of urgent packets
10. hot	continuous	number of "hot" indicators
11. num_failed_logins	continuous	number of failed login attempts
12. logged_in	discrete	1 if successfully logged in; 0 otherwise
13. num_compromised	continuous	number of "compromised" conditions
14. root_shell	discrete	1 if root shell is obtained; 0 otherwise
15. su_attempted	discrete	1 if "su root" command attempted; 0 otherwise
16. num_root	continuous	number of "root" accesses
17. num_file_creations	continuous	number of file creation operations
18. num_shells	continuous	number of shell prompts
19. num_access_files	continuous	number of operations on access control files
20. num_outbound_cmds	continuous	number of outbound commands in an ftp session
21. is_hot_login	discrete	1 if the login belongs to the "hot" list; 0 otherwise
22. is_guest_login		1 if the login is a "guest"login; 0 otherwise
23. Count	continuous	number of connections to the same host as the current connection in the past two seconds
24. serror_rate	continuous	% of connections that have "SYN" errors
25. rerror_rate	continuous	% of connections that have "REJ" errors
26. same_srv_rate	continuous	% of connections to the same service
27. diff_srv_rate	continuous	% of connections to different services
28. srv_count	continuous	number of connections to the same service as the current connection in the past two seconds
29. srv_serror_rate	continuous	% of connections that have "SYN" errors
30. srv_rerror_rate	continuous	% of connections that have "REJ" errors
31. srv_diff_host_rate	continuous	% of connections to different hosts
32. dst_host_count	continuous	count for destination host
33. dst_host_srv_count	continuous	srv_count for destination host
34. dst_host_same_srv_rate	continuous	same_srv_rate for destination host
35. dst_host_diff_srv_rate	continuous	diff_srv_rate for destination host
36. dst_host_same_src_port_rate	continuous	same_src_port_rate for destination host
37. dst_host_diff_host_rate	continuous	diff_host_rate for destination host
38. dst_host_serror_rate	continuous	serror_rate for destination host
39. dst_host_srv_serror_rate	continuous	srv_serror_rate for destination host
40. dst_host_rerror_rate	continuous	rerror_rate for destination host
41. dst_host_srv_rerror_rate	continuous	srv_serror_rate for destination host

References

[1] H. Debar, M. Dacier, A. Wespi, Towards a taxonomy of intrusion detection systems, Computer Networks 31 (1999) 805–822.

[2] S. Axelsson, 2000. Intrusion detection systems: a survey and taxonomy, Department of Computer Engineering, Chalmers University, Report No. 99-15.

[3] L.A. Zadeh, Role of soft computing and fuzzy logic in the conception, design and development of information/intelligent systems, Lecture Notes in Computer Science 695 (1998) 1–9.

[4] D. Song, M.I. Heywood, A.N. Zincir-Heywood, Training genetic programming on half a million patterns: an example from anomaly detection, IEEE Transactions on Evolutionary Computation 9 (3) (2005) 225–239, doi:10.1109/TEVC.2004.841683.

[5] J. Gomez, D. Dasgupta, Evolving fuzzy classifiers for intrusion detection, in: Proceeding of 2002 IEEE Workshop on Information Assurance, United States Military Academy, West Point, NY, USA, 2001, pp. 68–75.

[6] K. Shah, N. Dave, S. Chavan, S. Mukherjee, A. Abraham, S. Sanyal, Adaptive neuro-fuzzy intrusion detection system, in: Proceeding of

IEEE International Conference on Information Technology: Coding and Computing (ITCC'04), 5–7 April 2004, IEEE Computer Society, vol. 1, USA, 2004, pp. 70–74.
[7] Z. Zhang, J. Li, C. Manikopoulos, J. Jorgenson, J. Ucles, HIDE: a hierarchical network intrusion detection system using statistical preprocessing and neural network classification, in: Proceedings of the 2nd Annual IEEE Systems, Mans, Cybernetics Information Assurance Workshop, West Point, NY, USA, 2001, pp. 85–90.
[8] A. Abraham, R. Jain, Soft computing models for network intrusion detection systems, Studies in Computational Intelligence 16 (2005) 191–211, doi:10.1007/b98152.
[9] J.E. Dickerson, J. Juslin, O. Koukousoula, J.A. Dickerson, Fuzzy intrusion detection, in: Proceeding of IFSA World Congress and 20th North American Fuzzy Information Processing Society (NAFIPS) International Conference. North American Fuzzy Information Processing Society (NAFIPS), 25–28 July 2001, Vancouver, Canada, IEEE Computer Society, vol. 3, 2001, pp. 1506–1510.
[10] M. Gao, M.C. Zhou, Fuzzy intrusion detection based on fuzzy reasoning Petri nets, in: Proceeding of the IEEE International Conference on Systems, October 5–8, 2003, Man and Cybernetics, Washington, DC, USA, vol. 2, 2003, pp. 1272–1277.
[11] M.S. Abadeh, J. Habibi, C. Lucas, Intrusion detection using a fuzzy genetics-based learning algorithm, Journal of Network and Computer Applications (2005), doi:10.1016/j.jnca.2005.05.002.
[12] D. Nauck, R. Kruse, NEFCLASS – a neuro-fuzzy approach for the classification of data, in: Proceeding of 1995 ACM Symposium on Applied Computing, Nashville, USA, February 26–28, 1995, ACM Press, New York, 1995, pp. 461–465.
[13] J.-S.R. Jang, ANFIS: adaptive-network-based fuzzy inference systems, IEEE Transactions on Systems, Man, and Cybernetics 23 (3) (1993) 665–685, doi:10.1109/21.256541.
[14] H. Ishibuchi, T. Nakashima, T. Murata, A fuzzy classifier system that generates fuzzy if–then rules for pattern classification problems, in: Proceedings of 2nd IEEE International Conference on Evolutionary Computation, Perth, Australia, 29 November–1 December 1995, IEEE, vol. 2, 1995, pp. 759–764.
[15] J. Liu, J. Kwok, An extended genetic rule induction algorithm, in: Proceedings of the Congress on Evolutionary Computation Conference, 16–19 July 2000, La Jolla, CA, USA, vol. 1, 2000, pp. 458–463.
[16] DARPA Intrusion detection evaluation: http://www.ll.mit.edu/SSt/ideval/result/result_index.html.
[17] KDD Cup 1999 Intrusion detection dataset: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.
[18] B. Pfahringer, Winning the KDD99 classification cup: bagged boosting, SIGKDD Explorations 1 (2) (2000) 65–66.
[19] I. Levin, KDD-99 classifier learning contest LLSoft's results overview. SIGKDD explorations, ACM SIGKDD 1 (2) (2000) 67–75.
[20] M. Vladimir, V. Alexei, S. Ivan, The MP13 approach to the KDD'99 classifier learning contest, SIGKDD Explorations 1 (2) (2000) 76–77.
[21] W. Lee, S.J. Stolfo, K Mok, A data mining framework for building intrusion detection models, in: Proceedings of IEEE Symposium on Security and Privacy, 9–12 May 1999, Oakland, CA, USA, pp. 120–132.
[22] R. Agarwal, M.V. Joshi, PNrule: A New Framework for Learning Classifier Models in Data Mining, Department of Computer Science, University of Minnesota, Report No. RC-21719, 2000.
[23] M.R. Sabhnani, G. Serpen, Application of machine learning algorithms to KDD intrusion detection dataset within misuse detection context, in: Proceedings of International Conference on Machine Learning: Models, Technologies, and Applications, 23–26 June 2003, Las Vegas, Nevada, USA, 2003, pp. 209–215.
[24] E.H. Mamdani, S. Assilian, An experiment in linguistic synthesis with a fuzzy logic controller, International Journal of Man-Machine Studies 7 (1) (1975) 1–13.
[25] D.Y. Yeung, C. Chow, Parzen-Window network intrusion detectors, in: Proceeding of 16th International Conference on Pattern Recognition, 11–15 August, 2002, IEEE Computer Society, vol. 4, pp. 385–388.
[26] J. McHugh, Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection, ACM Transactions on Information and System Security (TISSEC) 3 (4) (2000) 262–294.
[27] M. Mahoney, P.K. Chan, An analysis of the 1999 DARPA/Lincoln laboratory evaluation data for network anomaly detection. Recent advances in intrusion detection: 6th international symposium, RAID (2003) 220–237, doi:10.1007/b13476.
[28] T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, IEEE Transaction on Systems, Man, and Cybernetics 15 (1985) 116–132.
[29] S. Chiu, Fuzzy model identification based on cluster estimation, Journal of Intelligent & Fuzzy Systems 2 (3) (1994) 267–278.
[30] R. Yager, D. Filev, Generation of fuzzy rules by mountain clustering, Journal of Intelligent & Fuzzy Systems 2 (3) (1994) 209–219.
[31] A. Nadjaran Toosi, M. Kahani, R. Monsefi, Intrusion detection based on neuro fuzzy classification, in: Proceedings of IEEE International Conference on Computing and Informatics, 6–8 June 2006, Kuala Lumpur, Malaysia, 2006.
[32] E.D. Goldberg, Genetic Algorithms in Search, Optimization & Machine Learning, Addison-Wesley, Reading, MA, 1989.
[33] M.R. Sabhnani, G. Serpen, Why machine learning algorithms fail in misuse detection on KDD intrusion detection data set, Journal of Intelligent Data Analysis 8 (4) (2004) 403–415.

Adel Nadjaran Toosi received his B.S. degree in 2003 and his M.S. degree in 2006 both in Computer Software Engineering, Ferdowsi University of Mashhad, Iran. His research currently focused on developing soft computing techniques and machine learning algorithms for intrusion detection systems. His interests include fuzzy systems, evolutionary soft computing techniques, genetic algorithms, neuro-fuzzy, neural networks and classification and clustering methods.

Mohsen Kahani is currently an assistant professor and IT director at Ferdowsi University of Mashhad, Iran. He received his B.S. in 1990, from the University of Tehran, Iran, his M.S in 1994, and his Ph.D in 1998 both from University of Wollongong, Australia. His research interests include information technology, e-government, e-learning and network management.