Intrusion detection by integrating boosting
genetic fuzzy classifier and data mining criteria
for rule pre-screening

Tansel Özyer\textsuperscript{a,*}, Reda Alhajj\textsuperscript{a,b}, Ken Barker\textsuperscript{a}

\textsuperscript{a}Department of Computer Science, University of Calgary, Calgary, Alberta, Canada
\textsuperscript{b}Department of Computer Science, Global University, Beirut, Lebanon

Received 28 June 2005; accepted 28 June 2005

Abstract

The purpose of the work described in this paper is to provide an intelligent intrusion detection system (IIDS) that uses two of the most popular data mining tasks, namely classification and association rules mining together for predicting different behaviors in networked computers. To achieve this, we propose a method based on iterative rule learning using a fuzzy rule-based genetic classifier. Our approach is mainly composed of two phases. First, a large number of candidate rules are generated for each class using fuzzy association rules mining, and they are pre-screened using two rule evaluation criteria in order to reduce the fuzzy rule search space. Candidate rules obtained after pre-screening are used in genetic fuzzy classifier to generate rules for the classes specified in IIDS: namely Normal, PRB-probe, DOS denial of service, U2R-user to root and R2L-remote to local. During the next stage, boosting genetic algorithm is employed for each class to find its fuzzy rules required to classify data each time a fuzzy rule is extracted and included in the system. Boosting mechanism evaluates the weight of each data item to help the rule extraction mechanism focus more on data having relatively more weight, i.e., uncovered less by the rules extracted until the current iteration.

\textsuperscript{*}Corresponding author.

E-mail addresses: ozyer@cpsc.ucalgary.ca (T. Özyer), alhajj@cpsc.ucalgary.ca (R. Alhajj), barker@cpsc.ucalgary.ca (K. Barker).

1084-8045/$ - see front matter \copyright 2005 Elsevier Ltd. All rights reserved.
Each extracted fuzzy rule is assigned a weight. Weighted fuzzy rules in each class are aggregated to find the vote of each class label for each data item.

**Keywords:** Intrusion detection; Genetic classifier; Fuzziness; Data mining; Weighted fuzzy rules

1. Introduction

Data mining methods and techniques have been successfully used for different application areas including bioinformatics, stock market, web analysis, among others. In simple terms, given a data set, data mining methods are designed for extracting previously unknown significant relationships and regularities out of huge heaps of details in large data collections (Grabmeier, 2003); it often refers to a particular step called knowledge discovery from database (KDD Cup 199 Data). Two popular methods used for data mining are association rule mining and classification. The goal of association rule mining is to generate all relationships between data items having support and confidence greater than user specified minimum support and minimum confidence. For a set of items, support is the number of transactions that contain all the items together; while the confidence of a rule is the number of transactions that contain all items that appear on the right hand side by considering only transactions that contain all items that appear on the left-hand side. Association rules detect the common usage of data items. Clustering is used to describe methods to group unlabeled data. Supervised clustering has two-way mechanism: either there is an oracle and responds whether it is right or not after the prediction, or the data set is split into two data sets: training and test data sets. On some occasions, one more extra data set is used for the validation. In the second type termed unsupervised clustering, the classes are known a priori and the task is to determine the classes from the data instances. Classification is used to find logical description that correctly classifies the novel cases. The three basic requirements for classification are: accuracy, simplicity and efficiency.

An intrusion is defined as any set of actions that attempt to compromise the integrity, confidentiality or availability of a resource (Heady et al., 1990). Denning (1987) summarizes the generic model of an intrusion detection mechanism by means of system input and the specific intrusions to be monitored. An intrusion detection system (IDS) monitors and restricts user access (behavior) to the computer system by applying certain rules. The rules are based on expert knowledge extracted from skilled administrators who construct attack scenarios and apply them to find system exploits. The system identifies all intrusions by users and takes or recommends necessary action to stop an attack on the database. As discussed in Allen et al. (1999), misuse detection and anomaly detection are two approaches of the intrusion detection system.

Misuse detection is the ability to identify intrusions based on a known pattern for the malicious activity. Manufacturers of IDS (using misuse detection) ensure that users have the most up to date patterns, also known as signatures, by sending them
updates to protect against new vulnerabilities. In anomaly detection, the system administrator defines the baseline, or normal state of the network’s traffic load, breakdown, protocol, and typical packet size. The anomaly detector monitors network segments to compare their state to the normal baseline and looks for anomalies. A Firewall differs from IDS in that it monitors and restricts access into and out of a network. IDS assesses a suspected intrusion after it happens and takes necessary action based on its restriction level.

Misuse detection is an effective approach to handle attacks that are known by the system. However, it is unpredictable in a dynamic system to know different various attacks. Besides, such an attempt would be time consuming and too complex. Anomaly detection is the complementary to the misuse detection. One disadvantage of anomaly detection is the likelihood of false alarms raised by the system. There are many systems proposed for intrusion detection. These systems can be grouped under three categories based on the type of detection they apply: (1) misuse detection based; (2) anomaly detection based; and (3) hybrid systems that employ a combination of misuse and anomaly detection.

The purpose of the work described in this paper is to demonstrate the usability of a boosting genetic fuzzy classifier spurred with the narrowed fuzzy rule search space. We initialize our population with the association rules chosen randomly among the pre-defined first $N$ rules having the largest product of support and confidence (pre-screening). Initially, all training instances will have equal weight. In our process, our genetic algorithm will try to extract one rule for one label iteratively. At each rule extraction, the weight of misclassified or uncovered instances will remain untouched, whereas others will be reduced mimicking; those instances are classified correctly and removed from the training data set. This process will continue until the test set error stagnates. The proposed approach has been tested using the public KDD CUP’99 intrusion detection data set available at the University of California, Irvine web site. The obtained results are promising; they demonstrate the effectiveness and applicability of the proposed method.

The rest of the paper is as follows. Related work on intrusion detection is introduced in Section 2. Section 3 describes the background knowledge required for the proposed architecture, including fuzzy logic, fuzzy association rule mining, and fuzzy genetic classifier based on boosting a genetic fuzzy classifier and pre-screening. The proposed work is introduced in Section 4. Experiment results are reported in Section 5. Section 6 is conclusions.

2. Related work

There are several approaches for solving intrusion detection problems. Lee et al. (1998) built an intrusion detection model by used association rule and frequent episode techniques on system audit data. Axis attribute(s) as a form of item constraints are used only to compute relevant patterns and an iterative level-wise approximate mining procedure is used to uncover the low frequency patterns in semi-automated way. NIDES system (Lunt, 1999) performs anomaly detection by using statistical
approaches. It generates profiles by using statistical measurements that tip into activity of subjects and profile generation. In general, there four types of statistical measurements: activity intensity, audit record distribution, categorical and ordinal.

Neural networks are trained to detect intrusion systems. An $n$-layer network is constructed and abstract commands are defined in terms of sequence of information units, the input to the neural in the training data. Each command is considered with pre-defined $w$ commands together to predict the next coming command expected from the user. After training, the system has the profile of the user. At the testing step, the anomaly is said to occur as the user deviates from the expected behavior (Fox et al., 1990; Ryan et al., 1998). Short sequences of system calls carry out the prediction process. In this system, Hamming distance comparison with a threshold is used to discriminate the normal sequence from the abnormal sequence (Bridges and Vaughn, 2000; Hofmeyr et al., 1998). Natural immune system is another proposed method to deal with the intrusion detection problem in distributed manner. Distributed positive and negative detectors are used to distinguish self and non-self behaviors (Hofmeyr, 1999). According to the work described in Balasubramaniyan et al. (1998) a multi-agent architecture detects the intrusion of multiple independent entities by autonomous agents working collectively. Another multi-agent architecture consisting of autonomous agents that are built on genetic programming method is also proposed in Crosbie (1995). Agents exploiting the learning power of genetic programming are evaluated with their performance and agents having highest performance are chosen to detect intrusions. Clustering techniques were applied on unlabeled data in order to discover anomalies in the data (Portnoy et al., 2001; Sequeira and Zaki, 2002).

Evolving fuzzy classifiers have been studied for possible application to the intrusion detection problem (Gomez and Dasgupta, 2001). System audit training data is used to extract rules for each normal and abnormal behavior by the genetic algorithm. Rules are represented as complete expression tree with identified operators, such as conjunction, disjunction and not. This paper aims at getting two rules, one for normal and one for abnormal, which are limited with a range (2–6). Further, the tree structure has been used to represent and classify data into the five classes: Normal, PRB-probe, DOS-denial of service, U2R-user to root and R2L-remote to local (Gómez et al., 2002). In both papers, three fitness functions are used with their corresponding weights for the multi-objective optimization process. Consequently, the user is required to determine the fitness function weights affecting each component existing in the rule; and each instance is labeled with the class having the largest fitness value.

3. The necessary background

3.1. Fuzzy logic

A classical set is characterized by having the membership degree of an element takes only one of two values as either 0 or 1. It is a set with a crisp boundary, where
there are no unambiguous boundaries. In other words, an object is either entirely in the set or not. Whereas a fuzzy set as its name implies is a set without crisp boundaries. The transition from “belonging to a set” to “not belonging to a set” is gradual; and this smooth transition is characterized by membership functions that give flexibility in modeling commonly used linguistic expressions. In simple terms, membership is not restricted to two values; rather it may take any value from the range (0, 1). This reflects a degree of membership; and this represents and models uncertainty as practiced daily by humans. So, fuzziness comes from the uncertain and imprecise nature of abstract thoughts and concepts (Jang and Sun, 1995).

$X$ is usually represents the universe of discourse. If $X$ is a collection of objects denoted each denoted $x$, then fuzzy set $A$ is defined as a set of ordered pairs as follows:

$$A = \{ (x, \mu_A(x)) | x \in X \},$$

where $\mu_A$ is called the membership function that maps each object $x$ from domain $X$ to a continuous membership value between 0 and 1.

There are two alternative ways to denote a fuzzy set $A$:

$$A = \left\{ \sum_{x_i \in X} \mu_A(x_i) | x_i \right\} \quad \text{if } X \text{ is discrete},$$

$$\int_X \mu_A(x) | x \right\} \quad \text{if } X \text{ is continuous}. (2)$$

There are several classes of parameterized ways to define membership functions; and the mostly used functions among them are: trapezoidal, bell functions, gaussian and triangular.

A parameterized membership function can be defined in terms of a number of parameters. For example, a triangular membership function is specified by three parameters $(a, b, c)$; and for a given value $x$, with known $a$, $b$, and $c$, the membership of $x$ may be computed as

$$\text{triangle}(x; a, b, c) = \max\left( \min\left( \frac{x - a}{b - a}, \frac{c - x}{c - b} \right), 0 \right). (3)$$

A fuzzy space having a normalized domain may be partitioned with five linguistic variables (L, LM, M, MH, H) and each linguistic variable is a parameterized triangular membership function as shown in Fig. 1.

A given object $x$ may be member of a given fuzzy set with a certain membership degree. Object $x$ may also be member of other fuzzy sets at the same time, but with different membership degree values.

An example conjunctive fuzzy rule $R_q$ is

**IF** $x_1$ is $A_{q1}$ and $x_2$ is $A_{q2}$ and … and $x_n$ is $A_{qn}$ **THEN** class is $c_q$, where $R_q$ is the qth fuzzy rule, $x = (x_1, x_2, \ldots, x_n)$ is an $n$-dimensional object of $X$, $c_q$ is the consequent class and each $A_{qi}$ is an antecedent fuzzy set. If the degree of membership of an object with each corresponding antecedent $A_{qi}$ is denoted $m_i$, then the firing strength $\mu_{A_q}$ of a rule using the max-min composition is

$$\mu_{A_q} = \min(m_1, m_2, \ldots, m_n). (4)$$
In general, fuzzy logic is appropriate for intrusion detection for two major reasons. First, many quantitative features are involved in intrusion detection and hence it is more natural to model them in terms of fuzziness than using sharp boundary intervals. Second, the fuzziness employed in representing these quantitative features helps to smooth the abrupt separation of normality and abnormality; it provides a measure of the degree of normality or abnormality for a given particular measure (Bridges and Vaughn, 2000).

3.2. Fuzzification of association rules

Association rules are used to represent the relationships between given data items. It refers to the usage of items and tries to establish relationships.

The first and still well-known example to association rules mining is the market-basket analysis. Customers fill their baskets with items selected from those existing in the market. We get to know which of the items are sold more together. Market owners check the relationships, and they organize promotions, special deals or relocate the items by considering the established relationships.

The association rule-mining problem may be stated as follows. Given a set of items \( I = \{I_1, I_2, \ldots, I_m\} \) and a database of transactions \( D = \{t_1, t_2, \ldots, t_n\} \) where and transaction \( t_i = \{I_{i1}, I_{i2}, \ldots, I_{ik}\} \) and \( I_{ik} \in I \), an association rule is a correlation of the form \( X \rightarrow Y \) where \( X, Y \subset I \) are disjoint sets of items called itemsets, i.e., \( X \cap Y = \phi \) and \( |D| \) denotes the number of transactions in \( D \).

As already stated above, the support for an association rule \( X \rightarrow Y \) is the percentage of transactions in the database that contain \( X \cup Y \). The confidence for an association rule \( X \rightarrow Y \) is the ratio of the number of transactions that contain \( X \) and \( Y \) together to the transactions that contain \( X \). In other words, support measures how often items occur in the database, whereas confidence expresses the strength of the rule. Especially, low support and high confidence values are used in order to find

![Fig. 1. Fuzzy space partitioned with five fuzzy classes (L—low, LM—low medium, M—medium, MH—medium high, H—high).](image-url)
interesting associations. The first and most common approach for finding association rule consists of two main steps:

1. Find large itemsets.
2. Generate rules from large (frequent) itemsets.

An itemset with support above a pre-specified (mostly by the user) threshold is called large or frequent itemset and denoted as $L$. All subsets of a frequent itemset are also frequent. After the large itemsets are found, generating the association rules is straightforward. A popular algorithm for mining association rules is the Apriori algorithm described in (Agrawal and Srikant, 1994).

Briefly, as the two criteria are determined for the association rule mining: support and confidence, we use these two evaluation criteria to carry out the pre-screening process. These definitions can be adapted to the fuzzy rules of the form $A_q \Rightarrow C_q$ (Hong et al., 2001; Ishibuchi et al., 2001), where the antecedent part is fuzzy but the consequent is not.

Assume that $\mu_{A_q}$ is the firing strength of the rule antecedent determined max–min composition. Support of the fuzzy association rule can be computed as

$$s(A_q \Rightarrow C_q) = \frac{|D(A_q) \cap D(C_q)|}{|D|} = \frac{\sum_{p \in Class_c} c_q \mu_{A_q}(x_p)}{m}. \quad (5)$$

And the confidence of the fuzzy rule can be computed as

$$s(A_q \Rightarrow C_q) = \frac{|D(A_q) \cap D(C_q)|}{|D(A_q)|} = \frac{\sum_{p \in Class_c} c_q \mu_{A_q}(x_p)}{\sum_{p=1}^{m} \mu_{A_q}(x_p)}. \quad (6)$$

Assuming that the $N$-dimensional pattern uses 5 linguistic variables for each dimension as shown in Fig. 1, and including “don’t care” condition, the total number of combinations for a rule becomes $(5 + 1)^N$. Our approach for reducing the search space at the end of the pre-screening process works as follows: among all existing association rules, for each class, the first $N$ rules are selected out as candidate keys. And those rules are used for the classification as described in Ishibuchi and Yamamoto (2002).

3.3. Boosting an evolving fuzzy classifier

There is a great interest in evolutionary algorithms that adapt the fuzzy knowledge base (Cordon et al., 2001). The role of evolutionary algorithm is either to tune the parameters of the fuzzy system or to automate the fuzzy knowledge base design entirely.
Recently, there are several genetic fuzzy classifiers, e.g., Hoffmann (2001); or evolutionary optimization of fuzzy rule based classification systems, e.g., Ishibuchi and Yamamoto (2002) and Reyes and Sipper (1999).

Boosting method for the genetic classifier approach (Hoffmann, 2001) is based on iterative rule learning. The rule base is built in incremental fashion repeatedly by invoking the genetic fuzzy rule generation algorithm. At each iteration, the fuzzy rule that classifies the current distribution best is selected out to be included in the fuzzy rule base as depicted in the block diagram shown in Fig. 2. The idea behind using the boosting mechanism is to aggregate multiple hypotheses generated by the same learning algorithm invoked over different distributions of the training data into a single composite classifier.

In this classifier, a fuzzy rule base is induced by the genetic algorithm. Genetic algorithms (GA) encode data in chromosome-like data structure and try to find potential solution to the problem by applying recombination (cross-over), mutation and selection. It is inspired by evolution of chromosomes (Goldberg, 1989).

Finding an artificial chromosome which when decoded and mapped back into the search space of the problem corresponds to a globally minimum or near minimum point in the original search space of the problem is the main goal of GA in multi-dimensional optimization. GA starts with the initial population, which consists of individuals. It has the data representation of individual character strings. Each individual (chromosome) string represents a solution, i.e., a fuzzy rule. The fitness function of individual strings of the initial population is evaluated. At each step, the population is evolved by the recombination of individuals by applying selection, crossover and mutation operators. After the evaluation, the termination condition is checked. If the algorithm converges to an optimal solution, the algorithm ends but in general this is not the usual case in GA process.

Each rule is coded in a chromosome. The population starts with initial rule representations. Those codified rules evolve during a pre-specified number of iterations. The process extracts one rule at a time. Given rule:

IF $x_1$ is $A_{q_1}$ and $x_2$ is $A_{q_2}$ and ... and $x_n$ is $A_{qn}$ THEN class is $c_{q'}$

![Fig. 2. Architecture of boosted genetic fuzzy classifier (taken from Allen et al., 1999).](image-url)
The corresponding chromosome representation is shown in Fig. 3. Each gene consists of one condition at the antecedent part where \( x_i \) is the variable and \( A_{qi} \) is the corresponding linguistic variable. If the variable is not assigned to a linguistic variable then we use the symbol “*” for “don’t care” as illustrated in Fig. 3.

The fitness function considers two objectives, namely the number of training instances covered by the rule \( R_i \) compared to the number of training instances that have the rule class label \( c_i \):

\[
f_1 = \frac{\sum_{x_k \in \text{Pos}} w_k \mu_{R_i}(x^k)}{\sum_{x_k \in \text{Neg}} w_k} \quad (7)
\]

and the negative examples covered by a rule:

\[
f_2 = \frac{\sum_{x_k \in \text{Neg}} w_k \mu_{R_i}(x^k)}{\sum_{x_k \in \text{Pos}} w_k} \quad (8)
\]

The fitness function result is gathered by using both objective functions:

\[
f = \begin{cases} 
0 & \text{if } f_2 > k_{\text{max}}, \\
 f_1 \times (1 - (f_2/k_{\text{max}})) & \text{otherwise}
\end{cases} \quad (9)
\]

In the system, each training instance \( d_k \in D \) is assigned a value \( d_k \), and initially \( w_k = 1 \).

At each step, GA is run and rule \( R_t \) with the best fitness value is inserted into the fuzzy rule base. Since each inserted rule is an incomplete weak classifier, rules in the fuzzy rule base have a classification error value, denoted \( E(R_t) \):

\[
E(R_t) = \frac{\sum_{x_k \in \text{Neg}} w_k \mu_{R_t}(x^k)}{\sum_{x_k \in \text{Pos}} w_k} \quad (10)
\]

After each rule extraction process, instances that are classified correctly will end up having the same weight, and those misclassified are reduced by some factor \( \beta^{(k)} \). Hence, after the extraction of rule \( R_t \), the weight at \( t + 1 \) becomes

\[
w_k(t + 1) = \begin{cases} 
w_k(t) & \text{if } c_i \neq c_k, \\
w_k(t) \times \beta^{(k)} & \text{if } c_i = c_k,
\end{cases} \quad (11)
\]

where \( \beta^{(k)} \) is calculated for each instance by using the following equation:

\[
\beta^{(k)} = \left( \frac{E(R_t)}{1 - E(R_t)} \right)^{\mu(R_t)^{\phi(k)}}. \quad (12)
\]
Eventually, rules generated for class labels are aggregated to give a cast decision for each class label. The class label which gets the majority of the votes will determine the label of the instance $x^k$ instead of the winner takes all approach:

$$C_{\text{max}}(x^k) = \arg \max_{R_i | c_i = C_m} \mu_{R_i}(x^k). \hspace{1cm} (13)$$

In our system, Eqs. (7)–(13) are used for: (1) fitness evaluation to find the rule to be included in the fuzzy rule base; (2) instance weight assignment to guide the search space to the misclassified instances; and (3) determining the class label of instance $x^k$.

4. The proposed approach

In the proposed architecture, fuzzy association rule mining is used to pre-screen the training data in order to reduce the search space. The product of support and confidence is considered for ranking the rules obtained, because support and confidence together directly affect the quality of a rule. The first $M$ strongest fuzzy association rules among the obtained $N$ association rules are chosen for each class.

The main idea in this paper is to run the genetic algorithm using Eqs. (7)–(12) for each class label individually and obtain some results as long as a fitness value greater than zero is obtained. In fact, all collected rules are extracted with classification error rate. During the genetic algorithm process, classified instances are considered with different weight reduced by Eqs. (11) and (12).

The main algorithm of the proposed approach is described in Table 1. Basically, for each class label $i$, at the beginning, the corresponding weight of each instance is initially set to 1. Randomized individuals are populated randomly among the rules obtained from association rules. After applying the genetic algorithm, the individual marked as the best individual (classification rule $R_t$) is inserted into the fuzzy base with its corresponding $E(R_t)$. This continues until a best individual with fitness score greater than zero is located.

At each rule extraction done by Step 5.2 of the proposed classification algorithm in Table 1, our system in Table 2 is run for pre-specified number of iterations. At the beginning, the population selects $K$ individuals randomly from the first $M$ fuzzy association rules according to the ranking procedure.

There exist two populations: one is used to be searched as the current population and the other for the best population that exists to keep track of the best individuals found under the current iteration. Initially, current population and best population are the same.

The next population consists of the newly generated individuals by the genetic operations. After every GA step, individuals having worse fitness values are replaced with individuals in the current population having better fitness values. At the same time, $K_{\text{elite}}$ elements are randomly chosen from the best population to be inserted in the current population. Therefore, the next population is generated by
(\(K_{\text{elite}} + (K - K_{\text{elite}})\)) individuals. Note that \((K - K_{\text{elite}})\) individuals are generated by the binary tournament (choosing the best two fit individuals to reproduce is likely to have better children for the next generation), recombination (uniform crossover (Syswerda, 1989) with probability \(p_c\)) and biased mutation operators. This means that in the biased mutation, we used two mutation probabilities, namely \(p_m(1 \rightarrow 0)\) and \(p_m(0 \rightarrow 1)\) as described in Ishibuchi and Yamamoto (2002); \(p_m(0 \rightarrow 1)\) is assigned to the mutation for moving the antecedent variable from inactive to active. In other words, gene addition and \(p_m(1 \rightarrow 0)\) is the mutation for changing the antecedent variable from active to inactive; gene deletion from the antecedent of the rule. \(p_m(1 \rightarrow 0)\) is assigned a larger value in order to decrease the length of the rules for interpretability. After specified number of iterations, the individual (classification rule) having the best fitness score is picked to be inserted in the fuzzy rule base with its \(E(R_c)\) (Eq. (10)). At the end, corresponding weight of each instance is evaluated by using the reduce factor \(\beta^{(k)}\) (Eq. (11)) to guide the genetic algorithm to find rules for the misclassified instances in the data set.

Table 1
The proposed classification algorithm

1. Generate fuzzy association rules in order to obtain rules of 1, 2, and 3 itemset as the length of antecedent for each class label \(i\).
2. Rank rules with two rule evaluation criteria.
3. Specify the population size \(K\) and the number of elite solutions \(K_{\text{elite}}\). The crossover probability \(p_c\) and the biased mutations \(p_m(1 \rightarrow 0)\) and \(p_m(0 \rightarrow 1)\).
4. For each class label \(i:\)
5. Repeat
   5.1 Initialize the GA population by selecting randomly \(K\) individuals from 1.
   5.2 Apply the genetic algorithm in Table 2.
   5.3 Insert the picked best individual (rule) into the fuzzy rule base with its corresponding \(\beta^{(k)}\).

Until the score is greater than zero.

Table 2
The employed Genetic algorithm

1. Copy the current population individuals to the best population.
2. While iteration number \(\leq n\)
   2.1 Generate \((K - K_{\text{elite}})\) individuals by using the genetic operations: binary tournament, crossover, and biased mutation.
   2.2 Evaluate the individual fitness functions of the current and best populations.
   2.3 Replace the worst individuals of the best population with individuals of the current population that has better fitness scores.
   2.4 Randomly select \(K_{\text{elite}}\) individuals from the best population and include them in the current population.
3. Mark the individual in best population with the highest fitness value and evaluate its \(E(R_c)\) to be picked as the result.
4. Re-evaluate \(w^k\) of each instance \(k\), by using its \(\beta^{(k)}\).
5. Experiments

We conducted our experiments on Intel Xeon 1.40 GHz CPU, 512 MB RAM and running Windows XP Dell PC. Oracle database 8i Personal Server Edition is used on the server side to store the training data, the test data and the obtained fuzzy association rules. Materialized view is utilized to store the membership values of the normalized values of the attributes.

Apriori algorithm (Agrawal and Srikant, 1994) for the fuzzy association rule mining was implemented in Microsoft Visual Studio 6.0 Visual Basic for the prescreening process. We implemented the genetic algorithm part by using GAlib, C++ genetic algorithm library (Wall). This system is run five times with chosen different 2% samples from the training data set and fuzzy association rules mining and GA for each at every step. KDD Cup 99 data consists of 42 data attributes representing the network traffic behaviour of one data instance (KDD Cup 199 Data). The training data set is composed of 494,014 data records (see Table 3). Test data set is composed of 311,029 data records (see Table 4); and class label information is obtained by using categorization awk script from Elkan.

The data set was pre-processed. Fig. 1 is used as the fuzzy space partitioning for domains of the quantitative attributes. Quantitative values are normalized for mapping the value to be between 0 and 1. Categorical attributes were represented by two valued true/false linguistic variables (e.g., logged in attribute is controlled by two linguistic variables: true/false).

Before implementing the system, the training data set was sampled in 2% by randomly selecting instances from the entire training data set. This is achieved by

<table>
<thead>
<tr>
<th>Table 3</th>
<th>Training data set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Class name</td>
</tr>
<tr>
<td>0</td>
<td>Normal</td>
</tr>
<tr>
<td>1</td>
<td>Probe</td>
</tr>
<tr>
<td>2</td>
<td>Denial of service (DOS)</td>
</tr>
<tr>
<td>3</td>
<td>User-to-root (U2R)</td>
</tr>
<tr>
<td>4</td>
<td>Remote-to-local (R2L)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 4</th>
<th>Test data set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Class name</td>
</tr>
<tr>
<td>0</td>
<td>Normal</td>
</tr>
<tr>
<td>1</td>
<td>Probe</td>
</tr>
<tr>
<td>2</td>
<td>Denial of service (DOS)</td>
</tr>
<tr>
<td>3</td>
<td>User-to-root (U2R)</td>
</tr>
<tr>
<td>4</td>
<td>Remote-to-local (R2L)</td>
</tr>
</tbody>
</table>
considering the percentages of each type of class label. Two rule evaluation criteria are used to generate the fuzzy association rules (see Table 5).

The first 1000 rules are selected as candidate rules to be used for initializing the population for each rule extraction (5000 in total). The following parameters are used: Population size: $K = 100$; number of elite solutions: $K_{\text{elite}} = 20$; replacement ratio for the genetic algorithm: 0.8; uniform crossover probability; $p_c = 0.9$; biased mutations; $p_m(1 \rightarrow 0) = 0.09$ and $p_m(0 \rightarrow 1) = 0.009$; number of iterations for each rule extraction, iteration number $= 30$ and $k_{\text{max}}$ value for the fitness is 0.2.

The target is to search extensively where the best fit individuals occur with large ratio of replacement in steady genetic algorithm. It will check enough number of rules for the classification rules with better fitness score. This is possible by using uniform crossover and biased mutation that will make effect on finding rules of minimized length as proposed in Ishibuchi and Yamamoto (2002). The current population is used for finding the best solution. Finally, the best individual among the individuals in the best population will be picked in order to be inserted in the fuzzy rule base. The fitness value does not critically depend on $k_{\text{max}}$. We used the same value as suggested in Hoffmann (2001).

At the end of the experiment, the average correct detection rates given below are obtained. Although it is not clear that the same training and test data sets are used with the winning entry, we also give the winning entry described in Elkan as a reference (see Table 6).

Out of 311,029 records, 187 records remained unclassified since the result was 0 for Eq. (13).

6. Conclusions

A Genetic fuzzy classifier has been implemented by using commercial of the shelf tools (COTS). Ideally, all instances are loaded to the database and the idea of
materialized views is used to prepare the existing membership values of the attributes.

We used the idea of boosting with classification. This classification algorithm only uses a specified number of fuzzy association rules obtained after the pre-screening process and at each step, the population is initialized with the individuals randomly chosen among the pre-screened rules in order to reduce the search space. In case no pre-screening occurs, the CPU time will be too long.

The GA is run iteratively. The search process is automatically led to discover new rules for the misclassified instances by considering the weight value of each instance individually. Negative coverage criteria used in the rules designate the error rate of the rules. Rather than adopting the winner takes all approach, rules have the cast vote and among all class labels, the one having the maximum vote decides on the class to which the instance should belong.

Our results are closer to the results of the winning entry (see Table 6). The reason of having low rates for the correct detection is apparent that there are fewer training instances for classes 1, 3 and 4. Furthermore, a large number of attributes can be reduced by using feature selection methods. Another helping idea would be to use another pre-screening criterion by considering another interestingness measurement in order to determine attribute subsets on which the class label can depend.

References


Elkan C. Results of the KDD’99 learning classifier contest (http://www-cse.ucsd.edu/users/elkan/clresults.html).


