Using Naïve Bayes Classifier to Accelerate Constructing Fuzzy Intrusion Detection Systems

Mehran Amiri, Mahdi Eftekhari, Farshid Keynia

Abstract- A Bayesian classifier is one of the most widely used classifiers which possess several properties that make it surprisingly useful and accurate. It is illustrated that performance of Bayesian learning in some cases is comparable with neural networks and decision trees. Bayesian theorem suggests a straight forward process which is not based on search methods. This is the major point which satisfies the marvelous time complexity of Bayesian classifier. At the other hand, constructing phase of fuzzy intrusion detection systems suffer from time consuming processes which are based on search methods. In this paper we propose a novel method to accelerate such processes using Bayesian inference. Experimental results show meaningful time reduction.

Keywords: Fuzzy intrusion detection systems, Naïve Bayes classifier, Rule`s consequent class, Time complexity.

I. INTRODUCTION

Nowadays, intrusion detection systems (IDS) have become an indispensable component of security infrastructure of computer networks. Since Denning first proposed an intrusion detection model in 1987 [33], many research efforts have been focused on how to effectively and accurately construct detection models. An intrusion detection system dynamically monitors the events taking place in a system, and decides whether these events are symptomatic of an attack or constitute a legitimate use of the system [34]. In general, IDSs fall into two categories according to the detection methods they employ, namely misuse detection and anomaly detection. Misuse detection identifies intrusions by matching observed data with pre-defined descriptions of intrusive behavior. Anomaly detection builds models for normal behavior and detects anomaly in observed data by noticing deviations from these models. When the fuzzy systems applied to intrusion detection systems for the first time, experts of security have the burden of generating necessary rules for such systems [35]. From the mid-1990s to the late 1990s, acquiring knowledge of normal or abnormal behavior had turned from manual to automatic. Artificial intelligence and machine learning techniques were used to discover the underlying models from a set of training data. To generate fuzzy rules, commonly employed methods are based on a partition of overlapping areas [36], or based on fuzzy implication tables [37], or by fuzzy decision trees [31] or association rules [1]. Recent methods of computational intelligence such as neural networks, Evolutionary computation and artificial immune systems can be used too.

Manuscript received on January 2013.

Mehran Amiri, Computer engineering department of Science and Research branch of Islamic Azad University, Kerman, Iran.

Mahdi Eftekhari, Computer engineering department of Science and Research branch of Islamic Azad University, Kerman, Iran.

Farshid Keynia, Computer engineering department of Science and Research branch of Islamic Azad University, Kerman, Iran.

Nowadays, Fuzzy systems have been applied to many aspects of human life and you can see a vast variety of methods, applications, and also commodities which are based on fuzzy logic, all around [2], [3]. One of the most important parts of this logic is Fuzzy rule-based systems. They have been applied successfully on classification problems [4], [5]. These systems also have been applied to computer security problems and results have shown their successfulness in that area [6], [7]. One of the most important features of Fuzzy rule-based systems which arises their popularity is their comprehensibility because they easily can be interpreted by a human users [6], [8]. Current approach for designing such systems is to generate the antecedent part of the fuzzy rules from data automatically and then determining the consequent class for each rule which each rule can best cover the relevant data points. Generating the antecedent part of the fuzzy classification rules, can be accomplished in many ways. The simplest way is to generate them randomly, but random search cannot generate good rules, especially in large pattern spaces. The heuristic and meta heuristic search approaches can be used in this case such as evolutionary algorithms, ant colony, bee colony particle swarm optimization and so on. The state of the art approach is to extract rules from data using rough set theory [13] and it`s extensions [14], such as variable precision rough set [15], rough fuzzy hybrids [16], fuzzy-rough hybrids [17] and recently flourished vaguely quantified rough sets [18]. They use different approaches to generate rules. After generating antecedent part of fuzzy rules, a typical process should determine the consequent class of them. This process examines all classes of dataset to find the best consequent class for each rule. One pass over all dataset instances is necessary in this case. Since the classification accuracy fully depends on how good the rules are, so lots of attempts have been accomplished to generate rules which best fit data. For example in evolutionary algorithms domain, many different fitness functions have been introduced to tackle this problem. For example Abadeh. et al. [6] counts the number of patterns which fall into covering area of the rules and uses this measure as a fitness function. Cordon et al. [9] utilizes confidence from the field of data mining for this purpose. Rule weighting is another option for improving the accuracy of fuzzy rule-based systems [5], [11], [12]. It is stated that rule weighting approach has a significant effect on the classification performance of fuzzy rule-based systems [10]. Ishibuchi et al. [5] compares 4 rule weighting measures on both artificial and real world datasets. In [11], Mansoori et al. propose a rule weighting method to improve the performance of fuzzy classification systems. This approach assigns weights to rules which exceeds unique interval [0,1].

In [12] Zolghadri et al. try to utilize ROC curves for rule weight tuning. Some researchers have focused on the generalization ability of such systems. For example Mansoori et al. [8] utilizes a measure in fitness function which tends to discard rules with longer antecedent parts. Such rules are more specific than shorter rules and tend to overfit the classifier. This is accomplished by finding lonely instances which fall into covering area of generated rules. Another way to improve the performance of fuzzy rule based systems is through the use of approaches which try to tune the parameters of the membership functions used to partition the pattern space. This approach is usually utilized when rule weighting approaches are not used [19]. Other approach is adapting measures from other fields such as data mining [4], [9]. For example Eftekhari et al. [4] have adapted precision and recall from data mining field to measure the effectiveness of rules. Other methods to improve the effectiveness of fuzzy rule-based systems are utilizing rules with multiple consequent classes [20] and some methods to inference with a fuzzy rule base like single winner rule and weighted vote [21]. All mentioned approaches which most of them are iterative, try to find the promising generated rules in each step, which offer better classification accuracy and also avoid time wasting by discarding not promising generated rules. Improvement in time complexity of such approaches is a side effect which is not studied explicitly. As we mentioned before, the classification accuracy depends on rules ability to classify unseen data points correctly. Thus authors tend to use methods which can determine the consequent class of rules with high degree of certainty. The reason why researchers were not used another methods to determine the consequent class of rules is justified in this way. In this paper we are going to introduce a fast and accurate method using Naïve Bayesian classifier to determine the consequent class of generated rules. This paper is organized as follows:

First we discus general design of fuzzy rule-based intrusion detection systems. In the next section the Bayesian inference is proposed. The following section describes our proposed method. In the next section the time complexity analysis is proposed. The experimental results form the next section and finally the last section concludes the paper.

II. DESIGNING FUZZY RULE-BASED CLASSIFICATION SYSTEMS

Assume R_j is a fuzzy if-then rule, in the form:

Rule R_j: if x_1 is A_{j1} and … and x_n is A_{jn}, then Class C_j j=1,2,…,N (1)

Where $x=[x_1,\ldots,x_n]$ is an *n* dimensional pattern vector, A_{ji} (i=1,…,n), is an antecedent linguistic value, C_j is the consequent class of R_j and *N* is the number of fuzzy rules. Generally for an *M*-class problem with *m* labeled patterns $x_p=[x_{p1},\ldots,x_{pn}]$, p=1,…,m, we should generate a set of *N* fuzzy if-then rules, in the form (1) to classify patterns. Normalizing attributes is a conventional process before designing the classifier.

After normalizing attributes, the pattern space is partitioned into fuzzy subspaces, and for each subspace, one fuzzy rule will be in charge of classifying patterns existing in that subspace [22]. To perform the partitioning, usually *k* suitable membership functions- indicating *k* linguistic values- are assigned to each input attribute. The use of triangular membership functions, because of their simplicity and interpretability is popular. There are two types of partitioning. Grid-type -or homogeneous- and accurate. Grid type partitioning is used when interpretability of fuzzy rules is important, while accurate partitioning preserves overall accuracy. We use first type of partitioning in this paper because interpretability of rules is more important for us. Fig. 1. illustrates these membership functions for four different values of *k*.

The relevant number of membership functions used to partition the feature space, has undeniable impact on accuracy of fuzzy rule-based classification systems. There is a delicate tradeoff between time complexity and accuracy with the number of feature space partitions. A partitioned feature space with a few membership functions could not achieve a convincing accuracy. On the other hand, using more membership functions to perform partitioning imposes lots of computation overheads on system and therefore increases time complexity. We use membership functions illustrated in Fig. 1. in this paper.

Given an input partitioning of pattern space, one approach is to consider all possible combinations of antecedent linguistic values and generate a fuzzy rule for each combination. For example, for a dataset containing *n* input attributes, and considering 14 mentioned membership functions of Fig. 1. For each attribute, the process should generate 14^n rules. This is clear that it is impractical to handle such a huge number of rules, especially for high dimensional problems. One approach to deal with this problem is to employ some criteria to select a small subset of best rules amongst all [23]. In [24] a solution is presented which adds one fuzzy set to the predefined set of fuzzy sets called *'don`t care'* (with linguistic label L_0), which is defined as $\mu_{L0}(x) = 1$ for all values of x. Every feature in the antecedent part of rule R_j which contains L_0, is not considered as a valid feature in the rule. Therefore shorter fuzzy rules, with limited number of antecedents can be generated.

In the field of data mining [25], *confidence* is frequently used to evaluate association rules. But before introducing that, we should be able to measure compatibility grade of each data instance with the antecedent part of rules. The fuzzy rule in (1) can be viewed as a fuzzy association rule $A_j \Rightarrow C_j$ where $A_j = (A_{j1},\ldots,A_{jn})$. The compatibility grade of pattern $x_p=[x_{p1},\ldots,x_{pn}]$ with the antecedent part of rule $R_j : (A_j \Rightarrow C_j)$ is computed using the product operator as:

$$\mu_j(x_p) = \prod_{i=1}^{n} \mu_{ji}(x_{pi}) \quad (2)$$

Where $\mu_{ji}(.)$ is the membership function of the antecedent fuzzy set A_{ii} and $A_{ii} \in \{L_0, L_1, \ldots L_{14}\}$.

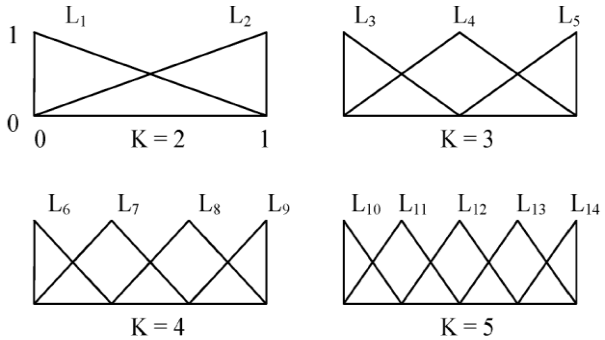

Fig. 1. Different partitioning of each input attribute

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

The fuzzy version of confidence is presented in [26], [27] as rule evaluation measure. The confidence of the fuzzy rule $A_j => C_j$ is written as follows:

$$confidence(A_j => C_j) = \frac{\sum_{x_p \in ClassC_j} \mu_j(x_p)}{\sum_{p=1}^{m} \mu_j(x_p)}. \quad (3)$$

The confidence can be viewed as a measure of ($A_j => C_j$) validity [5]. This means rules with higher values of confidence provide better classification accuracies.

Traditional process to find the consequence class C_j of fuzzy rule R_j is as follows:

C_j = arg max {confidence ($A_j => C_h$) | h= 1,…, M} (4)

This means that class h with maximum degree of confidence is selected as consequent class of the rule ($A_j => C_h$).

Although tuning of membership functions can improve classification accuracy of fuzzy rule-based systems, but this can be lead to degradation in interpretability of fuzzy rules. We use fuzzy rules with no weights and also no membership function tuning procedures in this paper.

III. BAYESIAN INFERENCE

In statistics, Bayesian inference is a method of inference in which Bayes' rule is used to update the probability estimate for a hypothesis as additional evidence is learned. Bayesian inference has found application in a range of fields including science, engineering, medicine, and law and has become famous for representing the best performance in some fields such as text mining [28]. It is illustrated that performance of Bayesian learning in some cases is comparable with neural networks and decision trees [29]. Bayesian inference computes the probabilities according to Bayes' rule:

$$P(A \mid B) = \frac{P(B|A) \times P(A)}{P(B)}. \quad (5)$$

Bayesian theorem suggests a straight forward process to find the hypothesis with maximum probability which is not based on search methods. This is the major point which satisfies the marvelous time complexity of Naïve Bayesian classifier.

A Naïve Bayesian classifier is one of the most widely used classifiers and possesses several properties [30] that make it surprisingly useful and accurate. A naive Bayesian classifier is a simple probabilistic classifier based on applying Bayes' theorem with strong (naive) independence assumptions. In simple terms, a naive Bayes classifier assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature, given the class variable. Depending on the precise nature of the probability model, naive Bayes classifiers can be trained very efficiently in a supervised learning setting.

Assume $f: x_p \rightarrow C$, is a function which maps x_p to C. x_p is a training pattern and C is a set of classes of training patterns. We want to find the class of pattern x_{test} using Naïve Bayes:

$$f_{NB}(x_{test}) = C_{NB} = \arg\max\{P(C_j) \times \prod_{i=1}^{n} P(a_i = v) \mid C_j\}. \quad (6)$$

It is obvious from formula (6) that all features assumed independent.

The value of $P(C_j)$ can be calculated by simply counting the number of instances that belong to class C_j and divide that to the cardinality of training set. The value of P (a_i = v| C_j) can be calculated by the same way too. Just count the number of instances belong to class C_j which the value of their i`th attribute is equal to v and then divide that to the cardinality of instances of class C_j. These can be written as follows:

$$P(C_j) = \frac{\left|\{x_p \in T \mid x_p \in C_j\}\right|}{\left|\{T\}\right|}. \quad (7)$$

$$P(a_i = v \mid C_j) = \frac{\left|\{x_p \in T \mid x_p \in C_j \bigcap x_p(a_i) = v\}\right|}{\left|\{x_p \in T \mid x_p \in C_j\}\right|} \quad (8)$$

Which *T* represents training set. We use this notation to show training set through this paper.

{.} Shows a set and |.| represents the cardinality of a set.

$x_p \in T$ represents instances which belong to T.

And at last $x_p(a_i) = v$ informs us that the i`th feature of instance x_p is equal to v.

IV. PROPOSED METHOD

In this section we are going to propose a novel method, which offers a faster method to determine the consequent classes of the generated rules. The main idea is to predict the consequent classes by the aid of Naïve Bayesian classifier. We use the Naïve Bayesian classifier that presented in previous section.

Naïve Bayesian classifier can predict the consequent class of test data instances in classification problems, but here we are going to utilize this kind of classifier to determine the consequent class of rules. This offers a fast, accurate and nearly optimal method which accelerates rule generation methods. In this paper, the terms *'feature'* and *'column'* are assumed equal and used interchangeably.

One of the key steps in constructing fuzzy rules is the action which replaces the values of data features of each data instance with some relevant fuzzy membership functions. If we look at this action from the viewpoint of discretization, we can look at each membership function as a potential discretized feature value. Our proposed method is based on this idea.

The first step of our proposed method consists of computing probabilities needed for Naïve Bayesian classifier. These probabilities for a dataset with discrete features could be calculated by simply counting some relevant values of features for each class. But for a dataset with continuous features, we should perform discretization first. We don`t have any explicit discretization phase here, but we use membership functions as discretized features, implicitly. This process consists of simply replacing each feature value with some relevant membership functions. The replacement strategy is as follows:

Assume x_p is a training example. Each membership function L_q, which can cover $x_p(a_i)$- the value of i`th feature of x_p, could be assumed as a potential candidate to replace $x_p(a_i)$. This means $\mu_{L_q}(x_p(a_i)) > 0$ .Since the membership functions partitioning the feature space, have overlaps with each other, there are more than one membership function that could cover $x_p(a_i)$. The process of computing probabilities of Naïve Bayesian is as follows:

Assume the training set T, consists of m training examples with n+1 attributes which n+1`th attribute represents the consequent class of the instances. Assume T is a matrix with m rows and n+1 columns. We construct the matrix $U_{m \times (n+1)}$ which every cell of U can consist of a *set* of objects. This matrix is used to save membership functions which can cover the values of features of data instances. In this paper we use x_p to represent an instance of T and x'_p to represent an instance of U. U(p,i) which represents $x'_p(a_i)$, $(1 \le i \le n,\ 1 \le p \le m)$, consists of a set of membership functions which can cover $x_p(a_i)$. U(p,n+1) represents the consequent class of $x_p(a_i)$. It is obvious that the last column of matrixes T and U are completely equal.

We can use formula (6) to determine the consequent class of the rule R_{test}. This can be rewritten as:

$$f_{NB}(R_{test}) = C_{NB} = \arg\max\{P(C_j) \times \prod_{i=1}^{n} P(a_i = L_q) \mid C_j\}. \quad (9)$$

Which $P(C_j)$ is calculated by formula (7). $P(a_i = L_q \mid C_j)$ represents the probability of a set of instances of U, which have L_q in their i`th column and their consequent class is C_j. This can be written as:

$$P(a_i = L_q \mid C_j) = \frac{\left|\{x'_p \in U \mid x'_p \in C_j \bigcap L_q \in x'_p(a_i)\}\right|}{\left|\{x'_p \in U \mid x'_p \in C_j\}\right|} \quad (10)$$

The statement $L_q \in x'_p$ represents that $U(p,i) = x'_p(a_i)$, consists a set of objects, not just one object.

Although the formula (10) seems convincing at the first glance, but it has a major drawback. The impact of *values* of attributes is neglected here. To solve this, we compute the sum of values of instances in i`th column of matrix T which can be covered by $L_q$ and belong to class $C_j$, for each $L_q$ in the i`th column of U and divide the computed value to the number of instances of class C_j. By this, an average value can be determined. Then we can compute the membership grade of this average with respect to L_q and multiply that by the value obtained from (10). Thus the formula (10) can be rewritten as follows:

$$P(a_i = L_q \mid C_j) = \frac{\left|\{x'_p \in U \mid x'_p \in C_j \bigcap L_q \in x'_p(a_i)\}\right|}{\left|\{x'_p \in U \mid x'_p \in C_j\}\right|} \times \mu_{L_q}\left(\frac{\sum_{\{x_p \in T \mid \mu_{L_q}(x_p(a_i))>0\}} x_p(a_i)}{\left|\{x_p \in T \mid \mu_{L_q}(x_p(a_i)) > 0\}\right|}\right) \quad (11)$$

Instance name	a1	a2	C
O_1	0.2	0.5	1
O_2	0.9	0.4	1
O_3	0.1	0.2	2
O_4	0.3	0.1	1

Fig. 2. The dataset used in example

Instance name	a1	a2	C
O_1	$\{L_1,L_2,L_3,L_4\}$	$\{L_1,L_2,L_4\}$	1
O_2	$\{L_1,L_2,L_4,L_5\}$	$\{L_1,L_2,L_3,L_4\}$	1
O_3	$\{L_1,L_2,L_3,L_4\}$	$\{L_1,L_2,L_3,L_4\}$	2
O_4	$\{L_1,L_2,L_3,L_4\}$	$\{L_1,L_2,L_3,L_4\}$	1

Fig. 3. The matrix U, constructed by membership functions L_1 to L_5 from Fig. 1. and dataset Fig. 2.

Let`s see an example:

We construct the matrix $U_{4\times3}$, with respect to membership functions $\{L_1,L_2,L_3,L_4,L_5\}$ obtained from Fig. 1. and dataset represented in Fig. 2. To find the value of U(1,1), we should find all membership functions which can cover the T(1,1) = 0.2. This is equal to find membership functions which their membership grade for T(1,1) = 0.2 is bigger than 0. It is obvious that all membership functions can cover 0.2 except L_5. Thus $U(1,1) = \{L_1,L_2,L_3,L_4\}$. The column C of matrix U, is the equal copy of column C of matrix T. The values of cells of matrix U are illustrated in Fig. 3.

In this section C_1 is used to represent the class 1 and C_2 is used to represent the class 2. Using formula (7) we can write:

$$P(C_1) = \frac{3}{4}$$

$$P(C_2) = \frac{1}{4}$$

Since the dataset has two classes and the column a_1 of matrix U, consists of vales $\{L_1,L_2,L_3,L_4,L_5\}$, then we should compute 10 probabilities. For example we compute $P(a_1 = L_3 \mid C_1)$. Based on formula (11), we should find data instances which belong to class C_1 and have L_3 value in their a_1 feature. The answer is the set $\{O_1, O_4\}$ which it`s cardinality is equal 2. The values of a_1 column of these instances in the matrix T, is 0.2 and 0.3. The average of these two values is equal to 0.25. Now based on formula (11), we can write:

$$p(a_1 = L_3 \mid C_1) = \tfrac{2}{3} \times \mu_{L_3}(0.25) = 0.3333$$

The rest of calculated probabilities are as follows:

$$p(a_1 = L_1 \mid C_1) = \tfrac{3}{3} \times \mu_{L_1}(\tfrac{0.2+0.9+0.3}{3}) = 0.5333$$
$$p(a_1 = L_2 \mid C_1) = \tfrac{3}{3} \times \mu_{L_2}(\tfrac{0.2+0.9+0.3}{3}) = 0.4667$$
$$p(a_1 = L_4 \mid C_1) = \tfrac{3}{3} \times \mu_{L_4}(\tfrac{0.2+0.9+0.3}{3}) = 0.9333$$
$$p(a_1 = L_5 \mid C_1) = \tfrac{1}{3} \times \mu_{L_5}(\tfrac{0.9}{1}) = 0.2667$$
$$p(a_1 = L_1 \mid C_2) = \tfrac{1}{1} \times \mu_{L_1}(\tfrac{0.1}{1}) = 0.9$$
$$p(a_1 = L_2 \mid C_2) = \tfrac{1}{1} \times \mu_{L_2}(\tfrac{0.1}{1}) = 0.1$$
$$p(a_1 = L_3 \mid C_2) = \tfrac{1}{1} \times \mu_{L_3}(\tfrac{0.1}{1}) = 0.8$$
$$p(a_1 = L_4 \mid C_2) = \tfrac{1}{1} \times \mu_{L_4}(\tfrac{0.1}{1}) = 0.2$$
$$p(a_1 = L_5 \mid C_2) = \tfrac{0}{1} \times \mu_{L_5}(\tfrac{0}{1}) = 0$$

The values of a_2 in matrix U, has 4 values $\{L_1,L_2,L_3,L_4\}$. Considering 2 classes of dataset, we need to calculate 8 probabilities:

$$p(a_2 = L_1 \mid C_1) = \frac{3}{3} \times \mu_{L_1}(\frac{0.5+0.4+0.1}{3}) = 0.6667$$
$$p(a_2 = L_2 \mid C_1) = \frac{3}{3} \times \mu_{L_2}(\frac{0.5+0.4+0.1}{3}) = 0.3333$$
$$p(a_2 = L_3 \mid C_1) = \frac{2}{3} \times \mu_{L_3}(\frac{0.4+0.1}{2}) = 0.3333$$
$$p(a_2 = L_4 \mid C_1) = \frac{3}{3} \times \mu_{L_4}(\frac{0.5+0.4+0.1}{3}) = 0.6667$$
$$p(a_2 = L_1 \mid C_2) = \frac{1}{1} \times \mu_{L_1}(\frac{0.2}{1}) = 0.8$$
$$p(a_2 = L_2 \mid C_2) = \frac{1}{1} \times \mu_{L_2}(\frac{0.2}{1}) = 0.2$$
$$p(a_2 = L_3 \mid C_2) = \frac{1}{1} \times \mu_{L_3}(\frac{0.2}{1}) = 0.6$$
$$p(a_2 = L_4 \mid C_2) = \frac{1}{1} \times \mu_{L_4}(\frac{0.2}{1}) = 0.4$$

Now we want to determine the class of rule R_{test} which can be produced by any rule generation process.

R_{test} : if a_1 is L_5 and a_2 is L_1

We simply need to compute two probabilities:

$$P(C_1) \times P(a_1 = L_5 \mid C_1) \times P(a_2 = L_1 \mid C_1) = \frac{3}{4} \times 0.2667 \times 0.6667 = 0.1334$$

$$P(C_2) \times P(a_1 = L_5 \mid C_2) \times P(a_2 = L_1 \mid C_2) = 0.25 \times 0 \times 0.8 = 0$$

The probability of class C_1 is bigger than the probability of class C_2. Thus the class of R_{test} is determined C_1.

V. TIME COMPLEXITY ANALYSIS

There are some bottlenecks which could be assumed as a measure for analyzing algorithms. We consider the number of product/division operations for complexity analysis in this section because the product/division operation is one of the operations which impose a heavy overhead on system. We assume here that the overhead of division operator is equal to the multiplication operator. As mentioned before, the traditional approach used to determine the consequent class of generated rules, in a fuzzy rule-based intrusion detection system is a brute force-like approach with one pass over all data instances. Although it doesn`t seem bad at first glance, but in the systems with lots of rules, it could be very time consuming. For example assume the number of *N* rules, are generated in an intermediate step of generation phase of a fuzzy rule-based intrusion detection system. The mentioned process to determine the consequent class of the rules, should be repeated *N* times. The process should compute the confidence of each rule to all classes of the dataset. This step could be completed in O(m), which *m* is the number of dataset instances. Calculating confidence measure needs to compute compatibility grade of each data instance with the rule R_j. The time complexity of calculating compatibility grade for each rule with average number of n/2 active antecedents is O(n/2) = O(n). Active antecedents in rules are those features that are not equal to L_0. The total number of product operations is m × (n/2). Thus computing the consequent class of N rules having the average n/2 active antecedents out of n, using the traditional approach requires N × m × (n/2) product operations which can be written as O(Nmn).

In proposed method, we don't have such a huge number of product operations. Computing probabilities using (11) needs only 2 division operations means O(1). By this way the impact of calculating compatibility grade is eliminated. This calculation should be accomplished for all *M* classes of dataset for membership functions covering $x_p(a_i)$ value. Notice that the number of membership functions which can cover x_{pi} is always less than the number of all membership functions used to partition the pattern space. We assume the average number of membership functions that can cover $x_p(a_i)$ is s/2 which s is number of all membership functions used to partition pattern space. Thus this operation can be done in 2 × M × (s/2) = O(Ms). Calculating all probabilities needs doing this process for n-1 columns. This means we need O(Msn) = 2 × M × (s/2) × (n-1) critical operations. Determining the consequent class of *N* rules with average active antecedents equal to n/2 for a dataset with *M* class needs (n/2) × M × N product operations which is O(MNn). Therefore the time complexity of NBAFRBS is O(Msn) + O(MNn) = max{O(Msn) + O(MNn)}. It is obvious that in lots of datasets the number of data instances *(m)* is much more than the number of classes *(M).* The number of membership functions used to partition the feature space rarely exceeds 14 [8], [11], but at the other hand using big values for *N* to obtain good accuracies is popular. There is no need to mention that calculating the probabilities needed for Naïve Bayes offline, can reduce time complexity to O(MNn) which is definitely better than O(Nmn).

VI. EXPERIMENTAL RESULTS

In this section we are going to lunch some experiments to compare new method with the traditional method. For this purpose we need a rule generation method to generate fuzzy rule antecedents. We use SGERD method (a Steady state Genetic algorithm for Extracting fuzzy classification Rules from Data) to generate rule antecedents [8]. The experiments are accomplished on a 3.00 GHz Intel Pentium 4 CPU (one processing core) with 512 MB of RAM on the Windows platform using MATLAB. The KDD99-10% [32] train set is used to accomplish experiments. The test set of KDD99 is used for test the classifiers. To eliminate the unrealistic results, we have run the experiments 8 times. In this paper we use *confidence* which has been proposed in section 2 as the

TABLE. 1. RESULTS OF COMPARING METHODS OF DETERMINING CLASSES OF GENERATED RULES ON KDD99-10%

Exp #	Time of New method	Time of traditional method	Accuracy of New method	Accuracy of traditional method
1	292.77	423.12	0.72	0.72
2	327.46	433.54	0.78	0.80
3	340.85	452.38	0.80	0.80
4	313.44	395.69	0.80	0.80
5	273.58	188.59	0.72	0.58
6	302.39	418.19	0.75	0.77
7	284.27	425.62	0.72	0.72
8	297.62	386.71	0.78	0.80
Ave#1	304.05	390.48	0.7687	0.7587
Ave#2	308.40	419.32	0.764286	0.7729

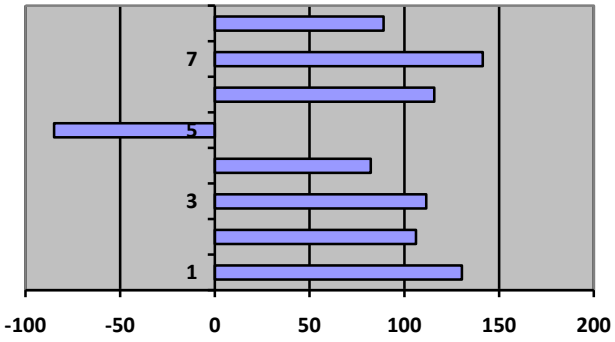

Fig. 4. Time improvement of proposed method for constructed intrusion detection systems based on Kdd99-10%. The horizontal axis shows improvement in time achieved by proposed method (Minutes), and the vertical axis represents experiment number (Exp#).

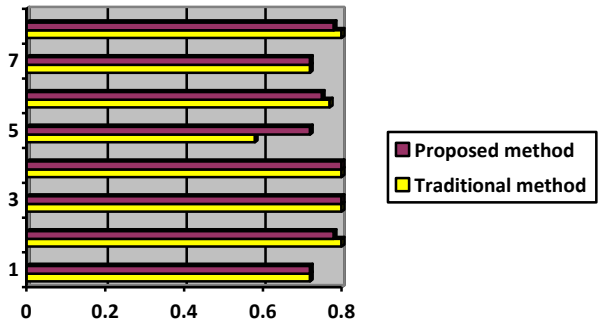

Fig. 5. Accuracies achieved by proposed and traditional methods for each constructed intrusion detection system based on Kdd99-10%. The horizontal axis shows accuracy, and the vertical axis represents experiment number (Exp#).

rule evaluation criteria. The relevant number of rules for each class can be determined through the heuristic method proposed in [8]. That is equal to 20. The obtained results can be seen in Table. 1, Fig. 4 and Fig. 5. All computed times are in minutes. In all experiments, the time for calculating the probabilities needed for Naïve Bayes, is approximately 2 seconds.

The results show that the time needed to construct the Fuzzy intrusion detection systems with the proposed method is much less than time needed for constructing such systems with the traditional method. The maximum of time improvement is 141.35 minutes and is related to experiment number 7. The minimum time improvement is related to experiment number 4 and it's about 82.25 minutes. Though the time improvement is very significant in most cases, but the time achieved by proposed method is worse than the time achieved by traditional method in experiment number 5. When we consider the accuracy of classifiers in experiment number 5, we notice that the training phase of classifier is not accomplished correctly. This might occur by evolutionary nature of SGERD [8]. The stopping criterion is satisfied, when the individuals are not good enough, probably. This results of this experiment for traditional method does not seem to be accurate. Since the experiment number 5 can worsen the overall results, we have calculated the averages of time and accuracy twice. First, with considering results of experiment number 5 (Ave#1) and second without considering the results of experiment number 5 (Ave#2). It is obviously clear that the intrusion detection systems which are constructed with proposed method need less time to be constructed. See Fig. 4. This figure shows time improvement of proposed method for constructed intrusion detection systems based on Kdd99-10%. The horizontal axis shows improvement in time achieved by proposed method (Minutes), and the vertical axis represents experiment number (Exp#). At the other hand, the accuracy of proposed method is very similar to accuracy of traditional method. See Fig. 5. In this figure, we have compared accuracies achieved by traditional and proposed methods. It is clear that achieved accuracies by proposed method are very similar to accuracies achieved by traditional method. The obtained accuracies are equal for proposed and traditional methods in 50 percent of experiments. The reason of obtaining different accuracy values by experiment number 5 have been negotiated before. Thus we can conclude that the average accuracy of proposed method is very similar to average accuracy of traditional method. The time improvement of proposed method is very significant. It should be stated that the proposed method can be integrated with any rule generation process and can accelerate it.

VII. CONCLUSION

In this paper we proposed a novel method which can accelerate the construction phase of Fuzzy intrusion detection systems. Experimental results on KDD99 datasets showed that our method can reduce time complexity of building fuzzy intrusion detection systems while preserving overall accuracy. Utilizing other successful methods in the field of text mining in constructing fuzzy rule-based classification systems is our aim in further works. Studying methods and processes which are used in other fields of data mining can be very promising.

REFERENCES

1. El-Semary, A., Edmonds, J., Gonzalez, J., Papa, M., "A framework for hybrid fuzzy logic intrusion detection systems", in: The 14th IEEE International Conference on Fuzzy Systems (FUZZ'05), Reno, NV, USA, 25–25 May 2005, IEEE Press, 2005, pp. 325–330.
2. Cordón, O., Herrera, F., Peregrín, A. "Applicability of the fuzzy operators in the design of fuzzy logic controllers", fuzzy Sets and Systems, 1997, pp. 15–41.
3. Glorennec, P. Y., "Application of fuzzy control for building energy management. In: Building Simulation", International Building Performance Simulation Association 1. Antipolis, A. France, 1991, pp. 197–201.
4. Eftekhari, M., Zolghadri, M. J., Katebi, S. D. "new criteria for rule selection in fuzzy learning classifier systems", Iranian Journal of Fuzzy Systems, 2006, Vol. 3, No. 1, pp. 77-89.
5. Ishibuchi, H., Yamamoto, T., "Rule Weight Specification in Fuzzy Rule-Based Classification Systems", IEEE transactions on fuzzy systems, August 2005, Vol. 13, No. 4. pp. 428-435.
6. Abadeh, M. S., Habibi, J., Lucas, C., "Intrusion detection using a fuzzy genetics-based learning algorithm", Journal of Network and Computer Applications, 2007, Vol. 30, pp. 414–428.
7. Abadeh, M. S., Mohamadi, H., Habibi, J., "Design and analysis of genetic fuzzy systems for intrusion detection in computer networks", Expert Systems with Applications, 2011, Vol. 38, pp. 7067–7075.
8. Mansoori, E. G., Zolghadri, M. J., S. D. Katebi, "SGERD: A steady state genetic algorithm for extracting fuzzy classification rules from data", IEEE transactions on fuzzy systems, August 2008, Vol. 16, No. 4, pp. 1061-1071.
9. Cordón, O., Del Jesus, M. J., Herrera, F., "A proposal on reasoning methods in fuzzy rule-based classification systems," International Journal of Approximate Reasoning., January 1999,Vol. 20, No. 1, pp. 21–45.

10. Ishibuchi, H., Nakashima, T., "Effect of rule weights in fuzzy rule-based classification systems," IEEE transactions on fuzzy systems, August 2001, Vol. 9, No. 4, pp. 506–515.
11. Mansoori, E. G., Zolghadri, M. J., S. D. Katebi, "A weighting function for improving fuzzy classification systems performance", Fuzzy sets and systems, 2007, Vol. 158, pp. 583-591.
12. Zolghadri, M. J., Mansoori, E. G., "Weighting function classification rules using receiver operating characteristics (ROC) analysis", Information science, 2007, Vol. 177, pp. 2296-2307.
13. Beynon, M., Curry, B., Morgan, P., "Classification and rule induction using rough set theory", Expert systems, July 2000, Vol. 17, No. 3. pp. 136-148.
14. Pawlak, Z. Skowron, A., "Rough sets: Some extensions", Information Science, 2007, Vol. 177, pp. 28-40.
15. Ziarko, W., "Variable precision rough set model", journal of computer and system science, 1993, Vol. 43, pp. 39-59.
16. Shen, Q., Chouchoulas, A., "A rough-fuzzy approach for generating classification rules", Pattern Recognition, 2002, Vol. 35, pp. 2425–2438.
17. Jensen, R., Cornelis, C., Shen, Q., "Hybrid Fuzzy-Rough Rule Induction and Feature Selection", Fuzzy IEEE conference, Korea, August 20-24, 2009, pp. 1151-1156.
18. Cornelis, C., De Cock, M., Radzikowska, A. M., "Vaguely Quantified Rough Sets", in Proceedings of 11`th international conference on Rough sets, Fuzzy sets, Data mining and granular computing (RSFDGrC2007), 2007, pp. 87-94.
19. Nauck, D., Kruse, R. "How the learning of rule weights affects the interpretability of fuzzy systems," in Proceedings of 7th IEEE International Conference on Fuzzy Systems, Anchorage, AK, May 1998, pp. 1235–1240.
20. Berg, J. V. D., Kaymak, U., Bergh, W. D., "Fuzzy classification using probability based rule weighting," in Proceedings of 11th IEEE International Conference on Fuzzy Systems, Honolulu, HI, May 2002, pp. 991–996.
21. Ishibuchi, H., Nakashima, T., Morisawa, T., "Voting in fuzzy rule-based systems for pattern classification problems," Fuzzy Sets and Systems., April 1999, Vol. 103, No. 2, pp. 223–238.
22. Ishibuchi, H., Nozaki, K., Tanaka, H., "Distributed representation of fuzzy rules and its application to pattern classification," Fuzzy Sets and Systems., 1992, Vol. 52, No. 1, pp. 21–32.
23. Ishibuchi, H., Yamamoto, T., "Comparison of heuristic criteria for fuzzy rule selection in classification problems," Fuzzy Optimal Decision Making, 2004, Vol. 3, No. 2, pp. 119–139.
24. Ishibuchi, H., Yamamoto, T., "Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining", Fuzzy Sets and Systems, 2004, Vol. 141, No. 1, pp. 59–88.
25. Cordón, O., Gomide, F., Herrera, F., Hoffmann, F., Magdalena, L., "Ten years of genetic fuzzy systems: Current framework and newtrends", Fuzzy Sets and Systems, 2004, Vol. 41, pp. 5–31.
26. Hong, T. P., Kuo, C. S., Chi, S. C., "Trade-off between computation time and number of rules for fuzzy mining from quantitative data", International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, October 2001, Vol. 9, No. 5, pp. 587–604.
27. Ishibuchi, H., Yamamoto, T., Nakashima, T., "Fuzzy data mining: Effect of fuzzy discretization," in Proceedings of 1st IEEE International Conference on Data Mining, San Jose, CA, November 2001, pp. 241–248.
28. Sebastiani, F., "Machine learning in automated text categorization", ACM Computing Surveys, 2002, pp. 1–47.
29. Mitchell, T., "Machine Learning", McGraw-Hill, 1997, pp. 156-199.
30. Zhang, H., "The optimality of naive bayes", Proceedings of the 17th International FLAIRS Conference 2004.
31. Liu, F., Lin, L., "Unsupervised anomaly detection based on an evolutionary artificial immune network", in: F. R., et al. (Eds.), Applications on Evolutionary Computing-EvoWorkshops 2005: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoMUSART, and EvoSTOC, Lausanne, Switzerland, 30 March 30–1 April 2005, volume 3449 of Lecture Notes in Computer Science, Springer, Berlin/ Heidelberg, 2005 , pp. 166–174.
32. KDD data set, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, [accessed on July 2011].
33. Denning, D. E., "An intrusion detection model", IEEE Transactions on Software Engineering, 1987, Vol. 13, No. 2, pp. 222–232.
34. Debar, H., Dacier, M., Wespi, A., "Towards a taxonomy of intrusion-detection systems", Computer Networks, 1999, Vol. 31, No. 8, pp. 805–822.
35. Dickerson, J.E., Dickerson, J.A., "Fuzzy network profiling for intrusion detection", in: Proceedings of the 19th International Conference of the North American Fuzzy Information Society (NAFIPS'00), Atlanta, GA, USA, 13–15 July 2000, IEEE Press, 2000, pp. 301–306.
36. Abraham, A., Jain, R., Thomas, J., Han, S. Y., "D-SCIDS: distributed soft computing intrusion detection system", Journal of Network and Computer Applications, 2007, Vol. 30, No. 1, pp. 81–98.
37. Zheng, C., Chen, L., "FCBI-an efficient user-friendly classifier using fuzzy implication table", in: L. Kalinichenko, R. Manthey, B. Thalheim, U. Wloka (Eds.), Advances in Databases and Information Systems, volume 2798 of Lecture Notes in Computer Science, Springer, Berlin/Heidelberg, 2003, pp. 266–277.