Intrusion Detection and Attack Classifier Based on Three Techniques: A Comparative Study

Dr. Adnan Mohsin Abdulazeez Brifcani **Adel Sabry Issa**

Received on: 29/6/2010
Accepted on: 5/1/2011

Abstract

Different soft-computing based methods have been proposed in recent years for the development of intrusion detection systems. The purpose of this work is to development, implement and evaluate an anomaly off-line based intrusion detection system using three techniques; data mining association rules, decision trees, and artificial neural network, then comparing among them to decide which technique is better in its performance for intrusion detection system. Several methods have been proposed to modify these techniques to improve the classification process. For association rules, the majority vote classifier was modified to build a new classifier that can recognize anomalies. With decision trees, ID3 algorithm was modified to deal not only with discreet values, but also to deal with numerical values. For neural networks, a back-propagation algorithm has been used as the learning algorithm with different number of input patterns (118, 51, and 41) to introduce the important knowledge about the intruder to the neural networks. Different types of normalization methods were applied on the input patterns to speed up the learning process. The full 10% KDD Cup 99 train dataset and the full correct test dataset are used in this work. The results of the proposed techniques show that there is an improvement in the performance comparing to the standard techniques, furthermore the Percentage of Successful Prediction (PSP) and Cost Per Test (CPT) of neural networks and decision trees are better than association rules. On the other hand, the training time for neural network takes longer time than the decision trees.

Keywords: Association Rules, Back-propagation, Decision Trees, Intrusion Detection, Neural Networks, Classification, Normalization.

كشف التطفل و تصنيف الهجمات بالاعتماد على ثلاثة تقنيات : دراسة مقارنة

الخلاصة

في السنوات الاخيرة تم تطوير انظمة كشف التطفل المبنية على مختلف نظريات الحسابات البرمجية. الهدف من هذا العمل هو تصميم وانشاء وتقييم انظمة كشف التطفل باستخدام ثلاثة تقنيات وهي تنجييم البيانات وقوانيين الاشتراك واشجار القرار والشبكات العصبية الاصطناعية ومن ثم المقارنة بينهم لتحديد الاكثر كفاءة من بين هذه الانظمة. تم استخدام وتحوير عدة تقنيات لتحسين عملية التصنيف. في حالة استخدام قوانين الاشتراك تم تحوير خوارزمية (the majority vote classifier) لبناء مصنف جديد قادر على كشف التطفل بكفاءة أكثر. اما في حالة استخدام اشجار القرار تم تحوير خوارزمية (ID3) للتعامل ليس فقة مع القيم المتقطعة وانما مع جميع القيم الرقمية. اما في حالة استخدام الشبكات العصبية الاصطناعية فقد تم استعمال خوارزمية (back-propagation) لتعليم الشبكة ذات ادخالات مختلفة (118, 51, and 41) لاعطاء الشبكة العصبية المعلومات المهمة حول المتطفل، وقد تم استخدام نظريات مختلفة لتعديل قيم الادخلات لزيادة سرعة العليم. في هذا العمل تم استعمال قاعدة البيانات 10% KDD Cup 99 في عملية التعليم. نتائج التقنيات المقترحة اظهرت تحسين في كفاءة انظمة كشف التطفل مقارنة بالتقنيات القياسية وكذلك تحسين في نسبة نجاح التخمين وكلفة كل اختبار خاصة عند استعمال الشبكات العصبية الاصطناعية واشجار القرار كما اظهرت النتائج بأن اشجار القرار ذات سرعة اكبر من الشبكات العصبية الاصطناعية في عملية التصنيف.

* Education College , University of Duhok / Duhok

https://doi.org/10.30684/etj.29.2.17
2412-0758/University of Technology-Iraq, Baghdad, Iraq

1. Introduction

Reliance on Internet and world wide connectivity has increased the potential damage that can be inflicted by attacks launched over Internet against remote systems. Successful attacks inevitably occur despite the best security precautions. Therefore, Intrusion Detection System (IDS) has become an essential component of computer security to detect *these* attacks with the aim of preserving systems from widespread damages and identifying vulnerabilities of the intruded system [1].

IDS can be categorized into anomaly detection and misuse detection [2]. Anomaly detection systems, flag observed activities that deviate significantly from the established normal usage patterns as anomalies (i.e. possible intrusion).

While misuse detection systems, use patterns of well-known attacks or weak spots of the system to match and identify known intrusion patterns or signatures.

Several soft-computing have been proposed in recent years for the development of IDS including Data Mining Association Rules (DM ARs), Decision Trees (DTs), and Artificial Neural Networks (ANNs).

Arman Tajbakhsh, et al., 2009 [2] proposed a new intrusion detection framework based on classification algorithm using fuzzy association rules for building classifiers. The fuzzy association rule sets are exploited as descriptive models of different classes. The method proposed to speed up the rule induction algorithm.

Victor H. et al., 2006 [3] proposed the use of ID3 to Web attack detection. The DT was made to classify a number of not previously considered Web application queries. The results show that the ID3 is an effective means for detecting and classifying web application attack queries.

Bouzida and F. Cuppens 2006 [4] proposed two different techniques for anomaly intrusion namely NN and DT in order to detect new attacks that are not present in the training data set. They improve them for anomaly intrusion detection and test them over the KDD Cup 99 data sets and over real network traffic in real time.

Mehdi Moradi and Mohammad Zulkernine 2004 [5], present a NN approach to intrusion detection. A multi-layer perceptron is used for intrusion detection based on an off-line analysis approach and applying the early stopping validation method on the proposed NN.

RachidBeghdad 2008 [6] aimed to determine which of the NN classifies well the attacks and leads to a higher detection rate of each attack. The paper focused on two classification types of records: a single class (normal, or attack), and a multiclass, where the category of attack is also detected by the NN. Five different types of NNs were tested: Multi-Layer Perceptron (MLP), Generalized Feed Forward (GFF), Radial Basis Function (RBF), Self-Organizing Feature Map (SOFM), and Principal Component Analysis (PCA) NN.

Yuehui Chen et al., 2007 [7] proposed an IDS model based on a general and enhanced Flexible Neural Tree (FNT). Based on the predefined instruction/operator sets, the framework allows input variables selection. Over layer connections and different activation functions for the various nodes involved.

2. Data Mining Association Rules

2.1 Association Rules (ARs)

An Association model [8] is often used for market basket analysis, which attempts to discover relationships or correlations in a set of items. Market basket analysis is widely used in data analysis for direct marketing, catalog design, and other business decision-making. An AR is a rule of the form:

(milk and bread) =>butter

Where (milk and bread) is called the rule body and butter the head of the rule. It associates the rule body with its head.

2.2 Apriori Algorithm

The most commonly known, and the first presented ARs mining algorithm is the Apriori algorithm which is introduced by Agrawal [9].The Apriori algorithm has proved to be an efficient algorithm in mining ARs. Apriori follows a two-step process to generate rules. The first step is to find all frequent itemsets. The algorithm counts item occurrences to determine large one-item sets. The other passes consist of two steps [10]. First, the large itemsetsLk–1 found in the (k–1) pass are used to generate the candidate itemsets Ck. Next, all those itemsets which have some k – 1 subset that is not in Lk–1is deleted, yielding Ck. Once the large itemsets are obtained, rules of the form a→ (l – a) are computed which is the second step, where a⊂l and l is a large itemset. There are two important basic measures for ARs [11] support and confidence:

Support: It shows the frequency of the patterns in the rule; it is the percentage of transactions that contain, both A and *B*.

$$Support = \frac{(\#\ of\ transactions\ involving\ A\ and\ B)}{(total\ number\ of\ transactions)} \quad (1)$$

Confidence: is the strength of implication of a rule; it is the percentage of transactions that contain B if they contain A.

$$Confidence = \frac{(\#\ of\ transactions\ involving\ A\ and\ B)}{(total\ number\ of\ transactions\ that\ have\ A)} \quad (2)$$

2.3 Classification using ARs

Classification using ARs [10] can be divided into three fundamental parts:

- AR mining.
- Pruning and.
- Classification.

The mining of ARs is a typical Data Mining (DM) task [8] that works in an unsupervised manner. A major advantage of ARs is that they are theoretically capable of revealing all interesting relationships in a database. But for practical applications the number of mined rules is usually too large to be exploited entirely. This is why the pruning phase is stringent in order to build accurate and compact classifiers. The smaller the number of rules a classifier needs to approximate the target concept satisfactorily, the more human-interpretable is the result. To build a Classifier, there are three ways [8][10] to use Classification rules:

Majority Vote (MV): The basic decision involves whether to consider every rule that covers an instance to a certain extent or to consider only a single rule; the simplest way to consider each rule equally. An unseen instance is classified using just the class label that is favored by the majority of rules in the set that are covering this instance. This kind of classification algorithm is called majority vote.

Weighting Scheme: Another scheme is called weighting scheme. The use of weighted scheme is easily accommodated; because AR mining produces a sorted set of rules according to their interestingness measure the MV scheme does not take any information out of the sort order. Hence in order to reveal the differences between associations rule mining algorithms weighted scheme is preferred.

Decision List: The other scheme is to look only at a single rule therefore the sorted set of class AR is used as a sorted list and the first rule that covers the instance to be classified is used for prediction. This type of approach is called decision list algorithm.

These three schemes are the basic schemes used in classification based on the AR set.

2.4 Data Preprocessing

In order to convert data from the original data to suitable data as an input to Apriori algorithm, there are several operations required. Fig. 1 describes the block diagram of data processing phase.

First we just change the last field of connection record which denotes the type of attack to class label. Because, when we partition the data into five subsets, each belongs to one of the five classes considered in dataset for generating the rules subsets, each subset belongs to a specific class; we will depend on this class label. For example: if attack is (pod) which is located in last field, then the connection record will be assigned to DoS category by changing (pod) to (DoS). Second, In order to deal with continuous values, the partition method was used to partition values to three intervals. The function Partition () was exploited for this reason. The algorithm works as follows:

First: we determine the maximum and minimum values for each continuous item.

Second: partition item domain to three parts depending on maximum and minimum value as shown below:

Part 1:

$Part\_1 \leq [min + (max - min) / 3]$

Part 2:

$[min + (max - min)]/ 3] < Part\_2 \leq [max - (max - min) / 3]$

Part 3:

$Part\_3 > [max - (max - min)/3]$

Where max and min represent a maximum and minimum value extracted from attributes domain. Finally after partition algorithm, each value of the attributes will be converted to Boolean values (0 or 1). For example: the attribute protocol_type contain three values (tcp, udp, icmp). If protocol_type of current connection record is (tcp), then protocol_type values will be converted to Boolean values such as protocol_ type = {1, 0, 0}. If it is (udp) then the protocol_type = {0, 1, 0}, and so on.

2.5 The Proposed Classifier

The MV scheme [10] depending only on majority of rules in the ruleset that are covering the instances, from this point we aim to modify this approach to build our classifier, depending not only on majority of rules but also depending on sum of confidence, and sum of support we call this approach Majority based on Rules, Confidences, and Support (MRCS).

Definition: Let T be the set of all training samples in which each sample corresponds to one of L possible classes. We partition T to L disjoint subsets (T_1, T_2 ..., T_L) such that T_l ($1 \leq l \leq L$) contains all samples of class l. These subsets are independently used to induce L rule sets (R_1, R_2 . . ., R_L) such that for each l ($1 \leq l \leq L$), $R_l = \{r_{l1}, r_{l2} ..., r_{lp}\}$ contains rules describing the patterns observed in class l.The next step (the classification phase) is to assign a label to a new sample say t. The flowchart of MRCS algorithm is illustrated in Fig. 2.

The proposed classifier works as follows: If the number of rules in R_l that covers the sample t, more than the number of rules in other subsets of rules, then the sample t will be classified as class l. If the number of rules that coverings the sample t in different subsets is equal, in this case the sample t will be classified as class l if sum of confidence of rules in R_l bigger than in other subsets. If two summation of confidence in different subsets of rules are equal, this time the sum of support is used to assign the sample t to class l using the same manner. Otherwise we use the default class to assign the sample t. The default class is majority class extracted from training phase in first scan. In this way every connection record will be assigned to one of the five classes considered in KDD Cup 99 data.

3. Decision Trees

3.1 Decision Trees (DTs)

DTs classifier by Quinlan [12] falls under the subfield of machine learning within the larger field of artificial intelligence. The DT is a classifier expressed as a recursive partition of the instance space, consists of nodes that form a rooted tree, meaning it is a directed tree with a node called a root that has no incoming edges referred to as an internal or test node. All other nodes are called leaves (also known as terminal or decision nodes). In the DT, each internal node splits the instance space into two or more sub-spaces according to a certain discrete function of the input attribute values. In the simplest and most frequent case, each test considers a single attribute, such that the instance space is partitioned according to the attributes value.

3.2 The Interactive Dichotomizer3 (ID3) Algorithm

The classical methods of attribute selection, implemented in well-known algorithms ID3 [12], is based on minimizing the entropy or information gain. The ID3 algorithm is used to construct a DT based on a given database. The tree is constructed top-down in a recursive fashion. At the root, each attribute is tested to determine how well it alone classifies the transactions. The best attribute is then chosen and the remaining transactions are partitioned by it.

3.3 Proposed Method

The initial definition of ID3 is restricted to attributes that take on a discrete set of values. First, the target attribute whose value is predicted by the learned tree must be discrete valued. Second, the attributes tested in the decision nodes of the tree must also be discrete valued. This second restriction can easily be removed so that continuous-valued decision attributes can be incorporated into the learned tree. This can be accomplished by dynamically defining new discrete-valued attributes that partition the continuous attribute values into a discrete set of intervals. For this reason K_mean algorithm is used here to partition each continuous attribute to three groups (group A, group B, group C).

Functionality of the proposed system is divided into four phases:

- Input Data and Partition process.
- Labeling continuous values.
- Training.
- Classification.

In the first phase, the continuous attributes will be partitioned to 3 groups (A, B, C) by applying K_mean algorithm on input data. In the second phase, the data is converted into suitable input data by assigning each continuous value to one of three groups (A, B, C) so that the input is given to ID3 algorithm. In the training phase, the system gathers knowledge about the normal and attacks from the preprocessed input data, and store the acquired knowledge. In classification phase, the system detects normal behavior or specific attack based on the knowledge, which is achieved during the training phase. Fig. 3 describes the block diagram of the proposed system.

4. Artificial Neural Networks

4.1 Introduction

An NNs [1][13] is an information processing system that is inspired by the way biological nervous systems, such as the brain, process information. It is composed of a large number of highly interconnected processing elements working with each other to solve specific problems. Each processing element is basically a summing element followed by an activation function. The output of each neuron (after applying the weight parameter associated with the connection) is fed as the input to all neurons in the next layer. The learning process is essentially an optimization process in which the parameters of the best set of connection coefficients (weighs) for solving a problem are found and include the following basic steps [14][15]:

- Present the NN with a number of inputs (vectors each representing a pattern)
- Check how closely the actual output generated for a specific input matches the desired output.

Normalization Methods: Data transformation such as normalization may improve the accuracy and efficiency of mining algorithms involving NNs. Such methods provide better results if the data to be analyzed has been normalized, that is, scaled to specific ranges such as (0, 1).

- Min-Max Normalization: The min-max normalization [23, 24] performs a linear transformation on the original data values. The new$_v$ will be found using the following formula:

$$new_v = \frac{v - minX}{maxX - minX} \qquad (3)$$

- Normalization by Decimal Scaling (DS): The decimal scaling [24] normalizes by moving the decimal point of values of feature X. The number of decimal points moved depends on the maximum absolute value of X. A modified value new$_v$ corresponding to *v* is obtained by using:

$$new_v = \frac{v}{10^n} \qquad (4)$$

Where n is the smallest integer such that max (|*v*'|) < 1.

- Logarithmic Normalization (Log): In this method the new value is calculated by the following formula.

$$new_v = \frac{Log(v - min + 1)}{Log(max - min + 1)} \qquad (5)$$

Back-propagation Algorithm (BP):

In this work, the standard BP algorithm was used for training the NN. The basic idea behind BP learning is to gradually adjust the weights of an ANN so as to reduce the error between the actual (y) and desired outputs (t) on a series of training cases. Each case is typically a pair, (ti, yi), indicating an element of the mapping between a domain and a range for some function [16][17].

4.2 The Proposed Architecture

We will exploit ANN to solve a multi-class problem of intrusion detection using a classic multi-layer feed-forward NN trained with the BP algorithm to predict intrusions. In this work, we are going to develop three different architecture of NN with a different number of neurons in input layer each architecture contains three layers (input layer, one hidden layer, and output layer). In the first NN, the number of neurons in the input layer will be (118) neurons. In the second NN we proposed (51) neurons, while ere for the third NN the number of neurons in the input layer will be (41) neurons. Also we aim to apply three different normalization methods (min-max question [3], logarithmic (log) question [5], and Decimal Scaling (DS) question [4] on input patterns to speed up the learning algorithm. Fig. 4 describes the block diagram of the proposed approach.

To construct input vector with 118 real values, two steps required. First, for each different string value of an attribute is assigned a neuron on the input layer. For example, for the protocol_type (tcp, udp, icmp), there are 3 inputs, say i_0, i_1, i_2 assigned to this attribute, each unit is initialized to 0. If the protocol_ type of the current connection record is "tcp" then i_0 is set to 1, if it is "udp" then i_1 is set to 1, and so on. For the service type, there are 66 inputs and

11 inputs for flag type and 38 inputs for the remaining attributes, the final input neurons on the input layer will be 118 neurons. The next step, the normalization method will be used to scale these inputs to (0, 1) range.
The 51 inputs for NN with 51 input patterns are calculated as follow: For protocol_ type (tcp, udp, icmp), we assigned 2 neurons on the input layer. This because, we need only 2 digits to represent these three values. For service attribute, there are 66 different values which need 7 digits to represent them, which are assigned to 7 neurons. While for flag attribute we assigned 4 neurons because, there are 11 different values. For the remaining attributes we assigned 38 neurons. The final number of the input layer will be 51 input neurons. Next, the normalization method will be applied to range the values between (0, 1).

The 41 real inputs for NN with 41 inputs are determined using the following manner: Each attribute from 41 attributes in connection record is assigned to 1 neuron on the input layer, for example: for the protocol_type (tcp, udp, icmp), there are only one input say i assigned to this attribute. If the protocol_type of the current connection record is "tcp" then i is set to 1, if it is "udp" then i is set to 2, otherwise i will be set to 3 and so on. Using this method, each attribute with different string values will be represented by the sequence of integer values. After this transformation, the normalization method will be used to scale these values to (0, 1) range. . Fig. 5 describes one normal connection record before the conversion process, while Fig. 6, 7, and 8 describe the same connection record after converted to vector of size (118, 51, 41 respectively).
While Fig. 9, 10, and 11 describe the converted record of size 41value in Fig. 5, after it normalized by max-min, SD, and Log methods respectively.

The number of neurons on the output layer is equal to the number of the total classes corresponding to the five classes considered in the KDD Cup 99 contest (Normal, Probing, DoS, U2R and R2L respectively). So the (output set) will be converted to vector of Boolean values such that, [1, 0, 0, 0, 0] for normal connection, [0, 1, 0, 0, 0] for class Probing, [0, 0, 1, 0, 0] for class DoS and so on.

5. KDD Cup 99 Data Sets

The data set used in the experiments is "KDD Cup 1999 Data" [21], which is a subversion of DARPA (Defense Advanced Research Projects Agency) 1998 dataset. The KDD cup 99 dataset includes a set of 41 features [18][19] derived from each connection and a label which specifies the status of connection records as either normal or specific attack type. These features had all forms of continuous, discrete, and symbolic, with significantly varying ranges falling in four categories. Intrinsic features of a connection, the content features, the same host features and the similar same service features. Likewise, attacks fall into four main categories DoS (Denial of Service), R2L (Remote to Local), U2R (User to Root) and Probe.

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% was employed here, distribution of normal and attack types of connection records in 10%KDD train dataset and test data respectively have been summarized in Table 1 [4].

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.

6. Evaluation Criteria

To rank the different results, there are standard metrics that have been developed for evaluating network intrusion detections. Detection Rate (DR) and false alarm rate are the two most famous metrics that have already been used. DR 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 [87]. In the KDD Cup 99, the criteria used for evaluation of the participant entries is the Cost Per Test (CPT) computed using the confusion matrix and a given cost matrix [21] .

A Confusion Matrix (CM) 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, 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, 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 [21] employed for the KDD Cup 99 classifier learning contest are shown in Table 2.

A Cost Per Test (CPT) is calculated by using the following formula:

$$PT = \frac{1}{N}\sum_{i=1}^{m}\sum_{j=1}^{m} CM(i,j) * C(i,j) \qquad (6)$$

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.

The accuracy is based on the Percentage of Successful Prediction (PSP) on the test data set.

$$PSP = \frac{\text{number of successful instance classification}}{\text{number of instances in the test set}} \qquad (7)$$

Higher values of PSP and Lower of CPT show better classification for the intrusion detection system. In this work, we used PSP and CPT measures to rank the different results.

7. Experiment Methodology and Result

The experiments go through two steps: first we performed preliminary experiments using the proposed algorithm then the standard algorithm. Second, after approving the eligibility of the proposed algorithms, then we train and test them on full 10% KDD Cup 99 data. For the experiments, we used computer Pentium 4, CPU 3.06 GHz, and 2 GB of RAM.

7.1 Preliminary Experiments

For preliminary experiments, we used subsets of KDD Cup 99 train and test datasets in Table 3 that describes different attack types and their corresponding occurrence number in the training and test data respectively. The number of training data is equal to 4947 and test data is equal to 3117, which are selected randomly from KDD Cup 99 dataset. From Table 3 Probing (41; 42) means that the number of records in train dataset of attack Probe is equal to (41 connection records), while the number of records in test dataset of this attack is equal to (42 connection records).

7.1.1 Preliminary Experiments for MRCS and Standard MV Classifiers

In this experiment, we compared MRCS classifier with standard MV scheme. For this reason, we wrote two programs in VC#, one for MV and another for MRCS algorithm. In the following for both the algorithms, the min-sup and min-conf are fixed to (0.20) and (0.7) respectively, after many experiments where these parameters varied over the interval (0.1, 0.8), the same train and test dataset are used for above algorithms. The confusion Tables 4 and 5 summarize the results of MV and MRCS classifiers respectively obtained throughout experiments. They indicate the DR for each classification type considering class of attacks (Normal, Probing, Dos, U2R, and R2L), PSP and CPT.

7.1.2 Preliminary Experiments for Proposed and Standard ID3 Algorithms

We performed preliminary experiments and compared our algorithm to standard algorithm ID3. We implemented our algorithm in VC# language. ID3 algorithm is borrowed and run from the WEKA3-4 (Waikato Environment for Knowledge Analysis). WEKA [22] is open source java code created by researchers at the University of Waikato in New Zealand. It provides many different machine learning algorithms. The WEKA program deals with “arff” file format, for this reason we wrote another program to convert proposed dataset, both train and test dataset to “arff” format. The confusion Tables 6 and 7 summarize the results obtained throughout the experiments by using the standard and proposed ID3 algorithms respectively.

7.1.3 Preliminary Experiments for Different NN with Different Number of Input Neurons Using Different Normalization Methods

In the experiments, we proposed three different architectures of NN, each with different number of neurons in the input layer (118, 51, and 41) neurons are used here, and applying different normalization methods to perform a BP algorithm.

First we applied max-min, next we applied normalization by DS, and finally we applied Log normalization.

The number of hidden layers considered in proposed NNs architectures is limited to only one hidden layer. The weights values of the different connections in the whole network are randomly initialized in the interval (−0.5, 0.5). Since each neuron on the output layer corresponds to one class, the neuron with the highest value defines the predicted class. Using this technique, every sample will be assigned to a class among the five classes defined apriori.

In first experiment, we used NN with 118 neurons in input layer, 240 neurons in hidden layer and 5 neurons in output layer. The number of epochs, momentum and learning rate are established during the training phase for each experiment. Fig. 12, 13, and 14 show the SSE for 500 epochs after we applied different normalization methods (min-max, DS, and Log) respectively.

We see from Fig. 14 that the SSE graph is better than when using min-max and DS methods.

In next experiment, we used NN with 51 neurons in the input layer, 100 neurons in hidden layer and 5 neurons in the output layer. Fig. 15, 16, and 17 show the SSE per 500 epochs when (min-max, DS, and Log) are applied on input patterns respectively.

As we see from Fig. 17 that the SSE graph with log normalization is better than when using min-max and decimal scaling normalization.

In the Last NN we use NN with 41 neurons in the input layer, 82 neurons in the hidden layer and 5 neurons in the output layer are used for these experiments. Fig. 18 describes the SSE per 500 epochs when min-max normalization was applied on input patterns. Where Fig. 19, describe the SSE per 2000 epochs after DS normalization. While the SSE per 1005 epochs with Log normalization is described in Fig. 20.

As we see from Fig. 20, the graph of SSE with Log normalization is better than when we used both min-max and DS.

Generally speaking that all figures of SSE, shows that when using Log normalization, is better than when using min-max and DS. Table 8 describes number of epochs, training time, PSP, and CPT obtained when Log normalization method is used with different NNs.

7.2 Experiments on Full 10% KDD Cup 99 Dataset Using DM ARs, DT, and NNs Techniques

All the results of NNs with Log normalization obtained from previews experiments; outperform all results when we used min-max, and DS, especially with the network that has 41 neurons in input layer. For this reason, we chose NN-41 and Log normalization method to train and test this network on full train and test 10%KDD Cup 99 dataset.

Also, after we approved the eligibility of both MRCS classifier and proposed ID3 algorithm, then we can use them with full 10% KDD Cup 99 dataset.

7.2.1 Experiment using ARs

In this experiment, the Apriori algorithm is used for generating the frequent itemsets, the min-sup is fixed to (0.2) after many experiments where this parameter varied over the interval (0.2, 0.5). The min-conf is fixed to (0.6) after varying it over the interval (0.4, 0.8). Table 9 presents the confusion matrix related to the DR, PSP, and CPT obtained using MRCS classifier and full test dataset.

7.2.2 Experiments Using DT

In this experiment the proposed ID3 algorithm is used to construct the DT. The full train and test dataset are used. Table 10 presents the confusion matrix related to the DR, PSP, and CPT obtained using the full test dataset.

7.2.3 Experiments Using NN

The NN used in this experiment construct with 41 neurons in input layer, 82 neurons in the hidden layer and 5 neurons in the output layer. The momentum is fixed to (0.8) after many experiments where this parameter varied over the interval (0.2, 0.9). The learning rate is fixed to (0.2) after varying it over the interval (0.1, 0.5). However, the weights values of the different connections in the whole network are randomly initialized in the interval [-0.5, 0.5]. Table 11 shows the DR for each classification type, PSP and CPT obtained from this experiment using the full test dataset.

8. Discussion

8.1 Discussion of Preliminary Experiments

In this discussion, we will compare the results obtained from the proposed algorithm with results obtained with standard algorithms.

8.1.1 Comparison Between MRCS and Standard MV Algorithm

From Tables 4 and 5, the experiments show that the MRCS algorithm gives better accuracy for Normal class compared to the standard algorithm. For Probing, U2R, and R2L classes, both methods give the same performance. For DoS class, there is only a small difference in the accuracy for MRCS and the standard Majority vote. For PSP, the two tables show that the MRCS is better than the standard algorithm. On the other hand, there is a small difference in the CPT for MV and MRCS.

8.1.2 Comparison between Proposed ID3 and Standard ID3 Algorithm

From Table 6 and 7, the experiments show that the proposed ID3 algorithm gives better accuracy for Probing, U2R and R2L classes compared to standard ID3 algorithm. For Normal and Dos class, there is only a small difference in the accuracy between these two techniques. For PSP and CPT, the two tables show our ID3 is better performance than when using standard ID3.

8.1.3 Comparison among Different NN Architectures

As we see from Fig.12, 13, 14, 15, 16, 17, 18, 19, and 20, the graphs of SSE with Log normalization for different NNs, are better than when we used both min-max and DS. That means the Log normalization improved the BP algorithm and the learning process.

While from Table 8, we see that the PSP is equal to (92.26) and CPT is equal to (0.2393) for NN-41 is better than both NN-51 and NN-118. While the NN-118 gives the worst PSP and CPT, this means whenever the number of neurons in the input layer becomes less, it gives a better PSP and CPT. Also less neurons in input layer reduce the training time.

8.2 Comparison among ARs, DT, and NN Techniques

From Tables 9, 10, and 11, the experiments show that the DT technique gives better accuracy for Normal, Probing, DoS and U2R class compared to ARs and NN techniques. For R2L classes, the NN gives better accuracy than both ARs and DT while the ARs gives the worst accuracy for detecting DoS. Table 12 summarizes the results of PSP, CPT and training time. As we see from Table 12, the ARs give the worst results of PSP and CPT, because the frequent item sets were only 2-itemsets. The best results could be obtained in case that the itemsets ware more than 2-itemsets. The time of generating frequent itemsets it took about 2.50 Hours, while there is significant difference in the PSP for DT compared with NN. For CPT, there is only a small deference for NN compared with DT, on the other hand it takes a very long time for training the network about 23.5 Days, While the DT takes only 2 Minutes for generating DT rules.

9. Conclusion

Here, a three techniques Data mining Association rules (DM ARs), Decision trees (DT), and Artificial Neural Network (ANN) based IDS, intended to classify the normal and attack patterns and the type of the attack, has been presented in this work. In the present work, we have presented different techniques used for modeling IDS and a comparison among them. Also, these techniques have been modified to improve them.

For ARs, we modified the Majority vote (MV) scheme considering not only majority of rules but also the confidence and support. The new algorithm outperforms the standard MV scheme.

For DT, we modified the standard ID3 algorithm to deal not only with discreet values but also to deal with continuous values. The new algorithm also outperforms the standard ID3.

For ANN, we built three different architectures, each with a different number of neurons in the input layer (118, 51, and 41 neurons) have been presented in this work. Also we applied different normalization method (min-max, Logarithmic (Log), and normalization by Decimal Scaling (DS)) on input patterns to improve the Back propagation algorithm. The results showed that the neural network (NN) with 41 neurons in the input layer and using the Log method for scaling patterns to (0, 1) range, gives better performance than other architecture.

For comparisons among above techniques (ARs, DT, and NN), the experimental results in Table 12 showed that ARs gives the worst results of PSP and CPT, for NN and DT significant difference in PSP about (0.42) and small deferent in CPT about (0.02). On the other hand, NN take a very long time about (23.5

Days) compared to DT which takes only (2 Minutes).

References

[1] Arman Tajbakhsh, Mohammad Rahmati, and Abdolreza Mirzaei, "Intrusion detection using fuzzy association rules", Applied Soft Computing ASOC-509, Elsevier B.V, 2008.

[2] M. Gnana Prasad, "Modeling Intrusion Detection System by Optimized Selection of ANN Training Algorithms", Master of engineering, Computer science and engineering, Thesis, Thapar University, Patiala, July 2008.

[3] Victor H. Garcia, Raul Monroy and Maricela Quintana, "Web Attack Detection Using ID3" . In, John Debenham, editor, 2nd IFIP International Symposium on Professional Practice in AI, WCC 2006, IFIP, Volume 219, pages 323-332, Santiago, Chile, 2006.

[4] Yacine Bouzida, and Frederic Cuppens, "Neural networks vs. decision trees for intrusion detection", IEEE, IST Workshop on Monitoring, Attack Detection and Mitigation MonAM2006 Tuebingen, Germany, September 2006.

[5] Mehdi Moradi, and Mohammad Zulkernine, "A Neural Network Based System for Intrusion Detection and Classification of Attacks", International Conference on Advances in Intelligent Systems, Theory and Applications, Luxembourg, Kirchberg, Luxembourg, IEEE, November 2004.

[6] Rachid Beghdad, "Critical Study of Neural Networks in Detection Intursions", Press, Computer and Security, Elsevier, June 2008.

[7] Yuehui Chen, Ajith Abraham, and Bo Yang, "Hybrid Flexible Neural-Tree-Based Intrusion Detection Systems", International Journal of Intelligent Systems, Volume 22, pp 337-352, 2007.

[8] Stefan Mutter, "Classification using Association Rules, Diploma of Computer Science", Thesis, Freiburg, Freiburg imBreisgau, Germany, 2004.

[9] Rakesh Agrawal, and Ramakrishnan Srikant, "Fast Algorithms for Mining Association Rules", VLDB Conference, Santiago, Shile, 1994.

[10] Rajanish Dass, "Classification Using Association Rules", W.P. No. 2008-01-05, January 2003.

[11] Tihomir Trifonov and Tsvetanka Georgieva, "Application for Discovering the Constraint-Based Association Rules in an Archive for Unique Bulgarian Bells", European Journal of Scientific Research, ISSN 1450-216X, Volume 31, No. 3, pp 366-371, 2009.

[12] Quinlan, J. R. (1993). "C4.5, Programs for Machine Learning", Morgan Kaufmann San Mateo Ca, 1993.

[13] Srinivasa Kumar Devireddy, and Settipalli Appa Rao, "Hand Written Character Recognition Using Back Propagation Network", Journal of Theoretical and Applied Information Technology, Volume.5, No. 3, 2009.

[14] Abdi, H., "Neural Networks In

M. Lewis-Beck", A. Bryman, T. Futing Eds, Encyclopedia for research methods for the social sciences, Thousand Oaks CA, Sage, pp 792-795, 2003.

[15] Niti Guru, Anil Dahiya, and NavinRajpal, "Decision Support System for Heart Disease Diagnosis using Neural Network", Delhi Business Review, Volume 8, No. 1, 2007.

[16] JIANGLukan, "On line monitoring of crystallization process using FBRM and artificial neural network", Master of SIDS, Thesis, University of Claude Bernard Lyon 1, 69622 Villeurbanne cedex, France, 2006.

[17] Choy Meiling, "Computational Protein Expression Prediction Using Artificial Neural Network system", School of Science and Technology, SIM University, May 2009.

[18] Ray I Chang, Liang Bin Lai, Wen De Su, Jen ChiehWang, and Jen ShiangKouh, "Intrusion Detection by Backpropagation Neural Networks with Sample-Query and Attribute-Query", International Journal of Computational Intelligence Research, Volume 3, No. 1, pp 6-10, 2007.

[19] V. Venkatachalam, and S. Selvan, "An Approach for Reducing the Computational Complexity of LAMSTAR Intrusion Detection System using Principal Component Analysis", International Journal of Computer Science IJCS, Volume 2, No. 1, November 2006.

[20] Carolina Fortuna, Blaz Fortuna, and MihaelMohorcic, "Anomaly Detection in Couputer Networks Using Linear SVMs", Slovenian Research Agency and the IST Programme of the EC UnderNeOn IST-4-027595-IP and PASCAL IST-2002-506778, September 2007.

[21] Charles Elkan, "Results of the KDD’99 Classifier Learning, SIGKDD Explorations", ACM SIGKDD, Issue 2, Volume 1, Page 67, January 2000.

[22] WEKA website, http://www.cs.waikato.ac.nz/ml/weka/.

[23] Andreea Vescan, and Horia F. Pop, "Constraint Optimization-Based Component Selection Problem", Studia Univ, Babes-Bolyai, Informatica, Volume LIII, No. 2, 2008.

[24] Luai Al Shalabi, ZyadShaaban, and Basel Kasabeh, "Data Mining: A Preprocessing Engine", Journal of Computer Science 2 9: 735-739, ISSN 1549-3636, 2006.

Table (1) the deferent attack types and their occurrence number respectively in the Training and test dataset

Normal(97,277; 60,593)	
Probing (4, 107; 4, 166)	DoS(391, 458; 229, 853)
ipsweep(1, 247; 306), mscan(0; 1, 053), nmap(231; 84), portsweep(1, 040; 364), saint(0; 736), satan(1, 589; 1, 633).	apache2(0; 794), back(2, 203; 1.098), land(21; 9), mailbomb(0; 5, 000), neptune(107, 201; 58, 001), pod(264; 87), processtable(0; 759), smurf(280, 790; 164, 091), teardrop(979; 12), udpstorm(0; 2).
U2R(52; 228)	R2L(1, 126; 16, 189)
buffer overflow(30, 22), httptunnel(0; 158), guess passwd(53; 4, 367), loadmodule(9; 2), perl(3; 2), perl(3; 2), ps(0; 16), rootkit(10; 13), sqlattack(0; 2), xterm(0; 13).	ftp write(8; 3), imap(12; 1), multihop(7; 18), named(0; 17), phf(4; 2), sendmail(0; 17), snmpgetattack(0; 7, 741), snmpguess(0; 2, 406), spy(2; 0), warezclient(1, 020; 0), warezmaster(20; 1, 602), worm(0; 2), xlock(0; 9), xsnoop(0; 4).

Table (2) the cost matrix

	Normal	Probing	DoS	U2R	R2L
Normal	0	1	2	2	2
Probing	1	0	2	2	2
DoS	2	1	0	2	2
U2R	3	2	2	0	2
R2L	4	2	2	2	0

Table (3) the different attack type and their corresponding occurrence number respectively in the training and test dataset

ormal(973 ; 606)	
Probing (41 ; 42)	DoS(3915 ; 2299)
Ipsweep (12 ; 3), Mscan (0 ; 11), Nmap (2 ;1), Portsweep (11 ; 4), Saint (0 ; 7), Satan (16 ; 16).	apache2 (0 ; 8), back (22 ; 11), land (0 ; 0), mailbomb (0 ; 50), Neptune (1072 ; 580) processtable (0;8), Pod (3 ; 1), udpstorm(0; 0). Smurf (2808;1641), Teardrop (10 ; 0),
U2R(5 ; 10)	R2L(13 ; 160)
buffer overflow (3 ;1), httptunnel (0 ; 3), loadmodule (0 ; 0), perl (0 ; 0) rootkit (2 ; 2), xterm (0 ; 2). Ps (0 ; 2), sqlattack (0 ; 0),	ftp write (0;0),imap (0; 0), guess passwd (2; 44), named (0; 0), multihop(0; 0),phf (0; 0), sendmail (0;0),snmpgetattack (0;77), snmpguess(0; 24),spy(0; 0), warezclient (10 ; 0),worm(0 ;0), warezmaster (1;15), xsnoop(0;0). xlock(0;0),

Table (4) The DR for each classification type, PSP, and CPT using MV algorithm

Predicted Actual	Normal	Probing	DoS	U2R	R2L	%DR
Normal(606)	**547**	5	34	1	19	90.26
Probing (42)	0	**31**	10	1	0	73.81
DoS (2299)	19	596	**1676**	8	0	72.9
U2R (10)	1	3	3	**2**	1	20
R2L (160)	106	6	39	0	**9**	5.62
***PSP* = 72.66%**			***CPT* = 0.4222**			

Table (5) the DR for each classification type, PSP, and CPT using MRCS algorithm

Predicted Actual	Normal	Probing	DoS	U2R	R2L	%DR
Normal(606)	**565**	5	16	1	19	93.23
Probing (42)	1	**31**	9	1	0	73.80
DoS (2299)	20	598	**1673**	8	0	72.77
U2R (10)	4	3	0	**2**	1	20
R2L (160)	140	6	5	0	**9**	5.62
***PSP* = 73.147%**			***CPT* = 0.4343**			

Table (6) the DR for each classification type, PSP, and CPT using standard ID3 algorithm

Predicted / Actual	Normal	Probing	DoS	U2R	R2L	%DR
Normal(606)	**494**	8	76	0	19	81.51
Probing (42)	6	**24**	6	1	0	57.14
DoS (2299)	23	530	**1710**	7	0	74.38
U2R (10)	4	1	0	**2**	0	20
R2L (160)	42	0	103	2	**10**	6.25
***PSP* = 71.887%**				***CPT* = 0.375**		

Table (7) the DR for each classification type, PSP, and CPT using proposed ID3 algorithm

Predicted / Actual	Normal	Probing	DoS	U2R	R2L	%DR
Normal(606)	**434**	59	90	0	19	71.61
Probing (42)	6	**28**	4	1	0	66.66
DoS (2299)	28	499	**1771**	1	10	77.03
U2R (10)	6	0	0	**1**	0	10
R2L (160)	43	0	103	5	**9**	5.62
***PSP* = 71.96%**				***CPT* = 0.409**		

Table (8) the PSP, CPT, and training time using Log Normalization with different NNs

NNs with Log Normalization	PSP	CPT	Training Time for 500 epochs
NN-118	83.5%	0.325	2.14 Hours
NN-51	91.82%	0.2537	15 Minutes
NN-41	92.268%	0.2393	13 Minutes

Table (9) the DR for each classification type, PSP and CPT using MRCS classifier

Predicted / Actual	Normal	Probing	DoS	U2R	R2L	%DR
Normal(60591)	**57332**	108	1181	77	1893	94.62
Probing (4166)	41	**2781**	818	85	441	66.75
DoS (229853)	7204	40564	**181780**	300	5	79.08
U2R (228)	20	136	2	**32**	38	14.03
R2L (16189)	14432	11	18	205	**1523**	9.4
***PSP* = 78.27%**				***CPT* = 0.3965**		

Table (10) the DR for each classification type, PSP and CPT using proposed ID3 algorithm

Predicted Actual	Normal	Probing	DoS	U2R	R2L	%DR
Normal(60,591)	**59560**	921	68	4	6	98.3
Probing (4,166)	367	**3259**	379	1	160	78.23
DoS (229,853)	6071	842	**222940**	0	2	96.99
U2R (228)	59	7	17	**143**	2	62.72
R2L (16,189)	14995	242	3	8	**941**	5.81
PSP **= 92.224%**				***CPT*** **= 0.2451**		

Table (11) the DR for each classification type, PSP and CPT using NN

Predicted Actual	Normal	Probing	DoS	U2R	R2L	%DR
Normal(60591)	**53343**	1360	3486	2	4255	88.04
Probing (4166)	373	**3384**	396	0	13	81.23
DoS (229853)	5450	487	**223545**	0	371	97.25
U2R (228)	61	69	46	**0**	52	0.0
R2L (16189)	10586	337	5	0	**5261**	32.50
PSP **= 91.80%**				***CPT*** **= 0.2241**		

Table (12) the PSP, CPT, and training time of the three techniques

The Techniques	PSP	CPT	Training Time
ARs	78.27%	0.3965	2.50 Hours
DT	92.22%	0.2451	2 Minutes
NN	91.80%	0.2241	23.5 Days

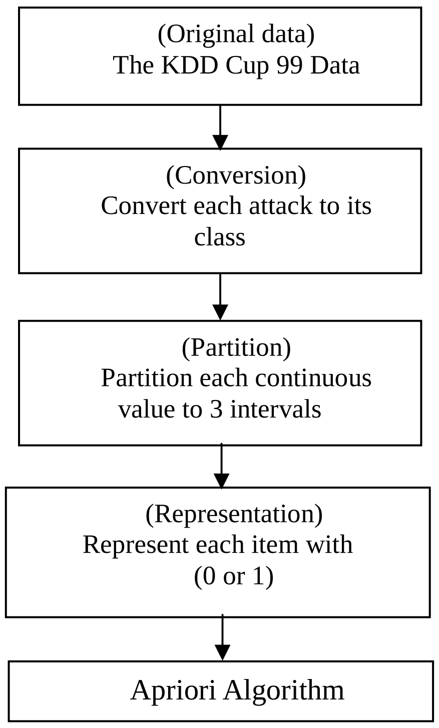
(Original data)
The KDD Cup 99 Data
(Conversion)
Convert each attack to its class
(Partition)
Partition each continuous value to 3 intervals
(Representation)
Represent each item with (0 or 1)
Apriori Algorithm

Figure (1) Data processing block diagram

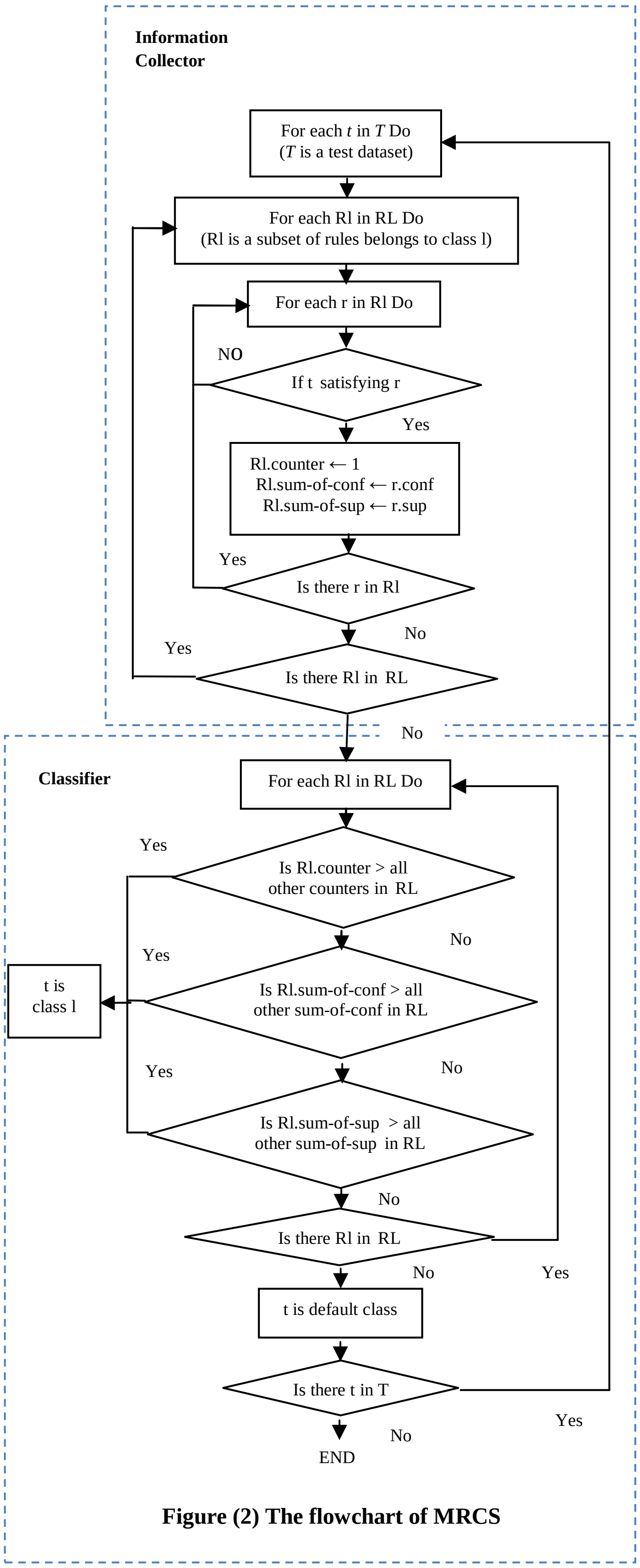

Figure (2) The flowchart of MRCS

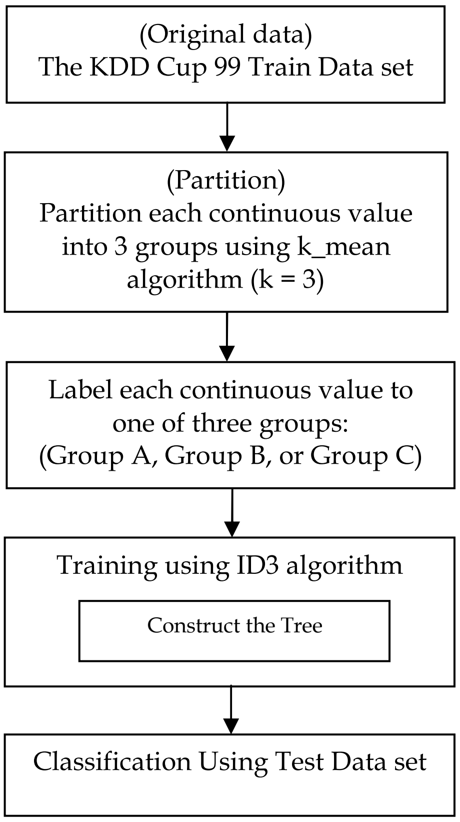
(Original data)
The KDD Cup 99 Train Data set
(Partition)
Partition each continuous value into 3 groups using k_mean algorithm (k = 3)
Label each continuous value to one of three groups:
(Group A, Group B, or Group C)
Training using ID3 algorithm
Construct the Tree
Classification Using Test Data set

Figure (3) The block diagram of the proposed system

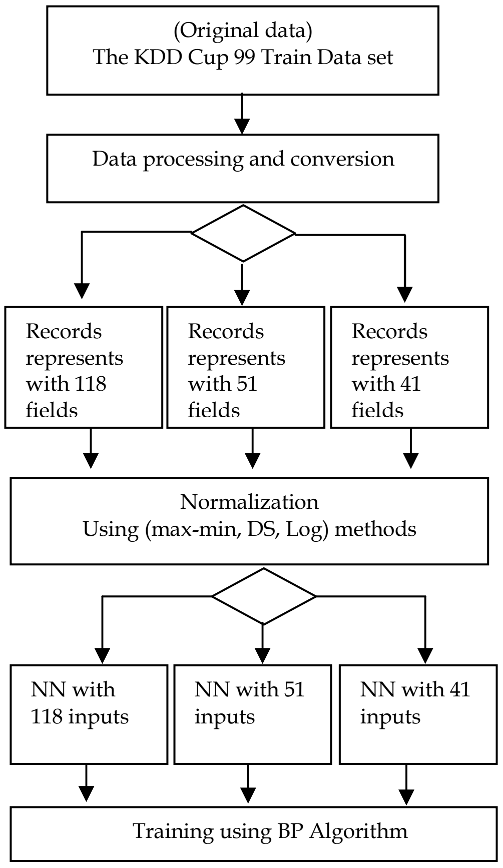
(Original data)
The KDD Cup 99 Train Data set
Data processing and conversion
Records represents with 118 fields
Records represents with 51 fields
Records represents with 41 fields
Normalization
Using (max-min, DS, Log) methods
NN with 118 inputs
NN with 51 inputs
NN with 41 inputs
Training using BP Algorithm

Figure (4) The block diagram of the proposed approach

```
0,tcp,smtp,SF,523,277,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,5
5,108,0.55,0.09,0.02,0.02,0.00,0.00,0.00,0.00,normal.
```

Figure (5) The normal connection record before conversion process

```
0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,523,277,0,0,0,0,0,1
,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,1,0,0,55,108,0.55,0.09,0.02,0.02,0,0,0,0,
```

Figure (6) The normal connection record after converted to vector of size 118

0,1,0,0,1,0,0,0,0,0,1,0,0,0,523,277,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,1,0,0,55,
108,0.55,0.09,0.02,0.02,0,0,0,0.

Figure (7) The normal connection record after converted to vector of size 51

0,1,2,1,523,277,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,1,0,0,55,108,0.55,0.09,0.02,
0.02,0,0,0,0,

Figure (8) The normal connection record after converted to vector of size 41

0,0.00191204588910134,0.00382409177820268,0.00191204588910134,1,0.52963671
1281071,0,0,0,0,0,0.00191204588910134,0,0,0,0,0,0,0,0,0,0.00191204588910134,0.
00191204588910134,0,0,0,0,0.00191204588910134,0,0,0.105162523900574,0.206500
956022945,0.00105162523900574,0.00017208413001912,3.82409177820268E-
05,3.82409177820268E-05,0,0,0,0,

Figure (9) The connection record in figure (5) after it had been normalized by Max-min method

0,0.001,0.002,0.001,0.523,0.277,0,0,0,0,0,0.001,0,0,0,0,0,0,0,0,0,0.001,0.001,0,0,0,0,0
.001,0,0,0.055,0.108,0.00055,9E-05,2E-05,2E-05,0,0,0,0,

Figure (10) The connection record in figure (5) after it had been normalized by DS method

0,0.11070000816189,0.175455361766121,0.11070000816189,1,0.898766843016396,0,0,0
,0,0,0.11070000816189,0,0,0,0,0,0,0,0,0,0.11070000816189,0.11070000816189,0,0,0,0,
0.11070000816189,0,0,0.642874237270769,0.749238059993981,0.0699920966162997,0.
0137631255594963,0.00316260538136082,0.00316260538136082,0,0,0,0.

Figure (11) The connection record in figure (5) after it had been normalized by Log method

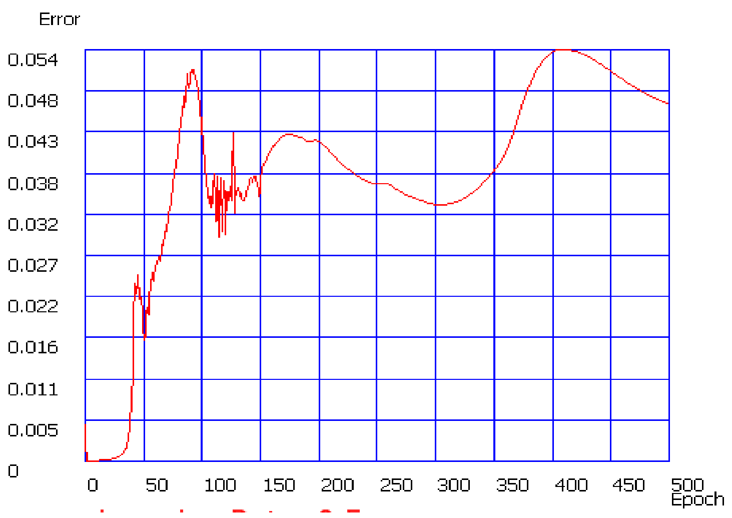

Figure (12) The SSE for 500 epochs with NN-118 and min-max normalization

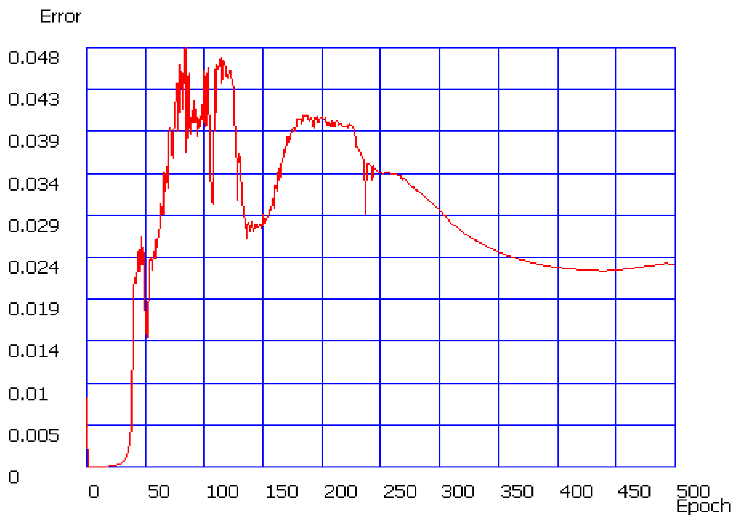

Figure (13) The SSE for 500 epochs with NN-118 and DS normalization

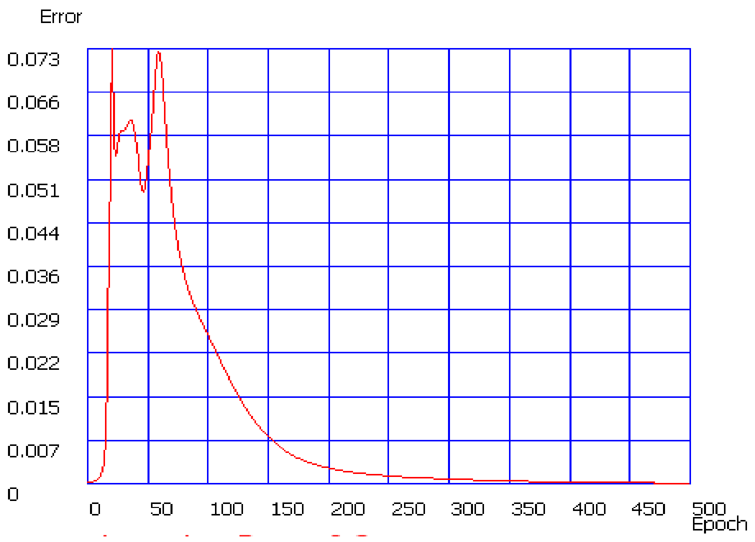

Figure (14) The SSE for 500 epochs with NN-118 and Log normalization

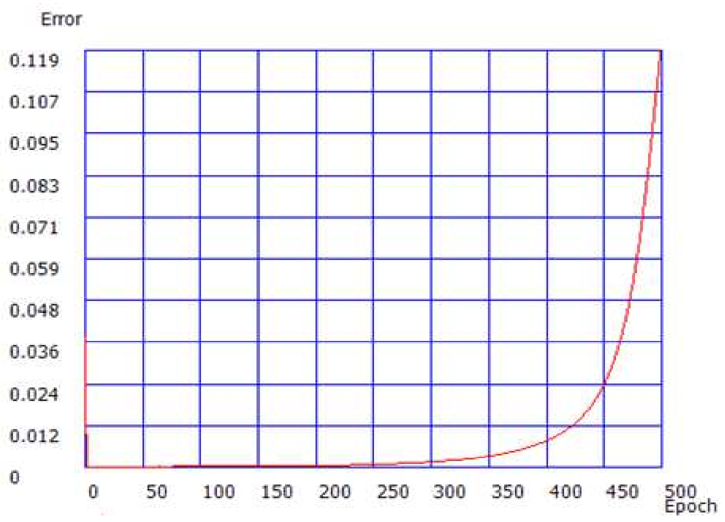

Figure (15) The SSE for 500 epochs using NN-51 and min-max normalization method

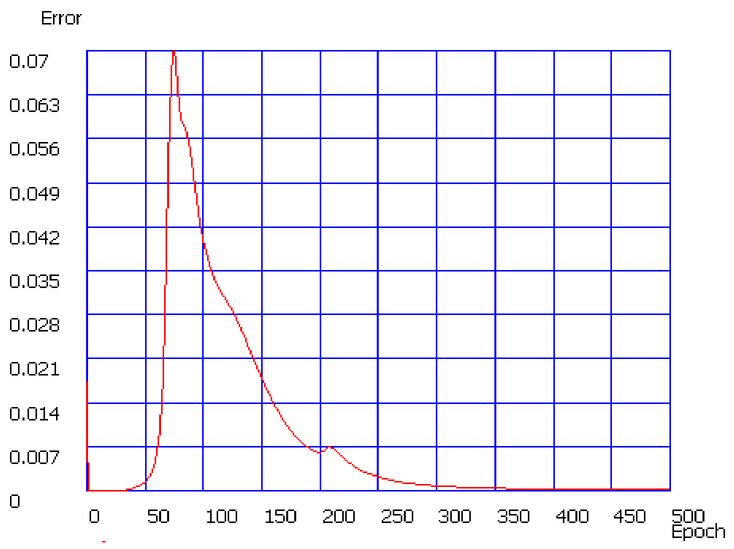

Figure (16)The SSE for 500 epochs using NN-51 and DS normalization

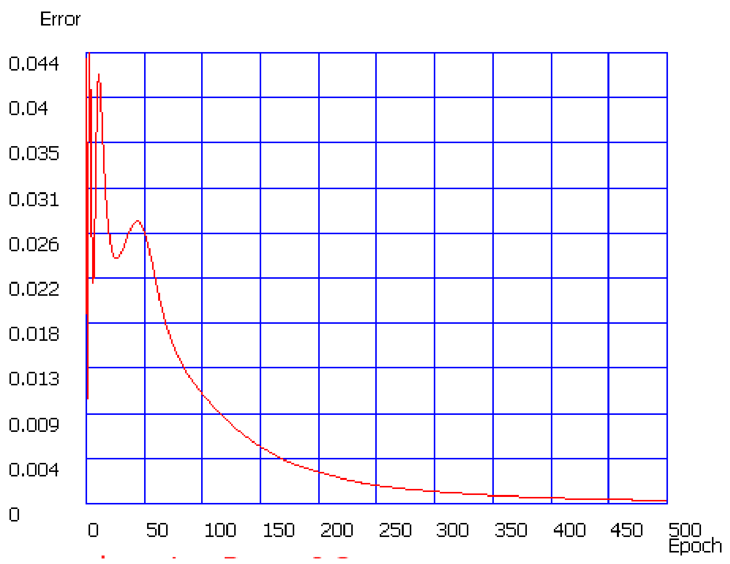

Figure (17) The SSE for 500 epochs using NN-51 and Log normalization method

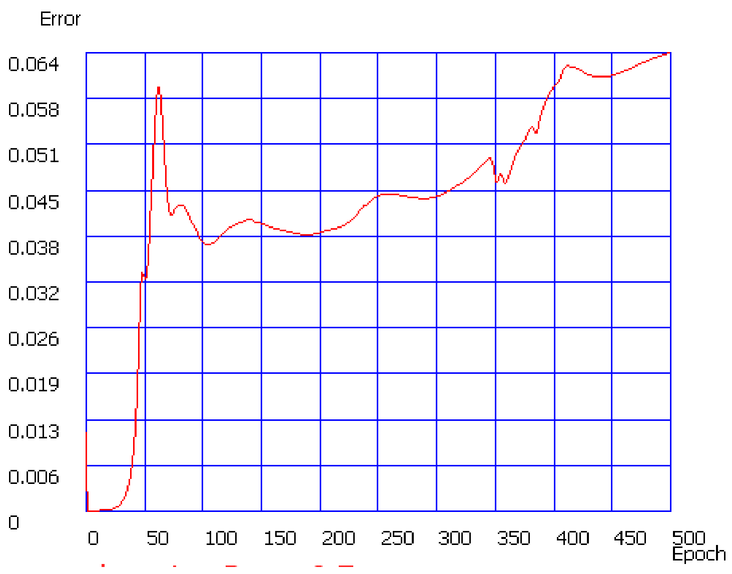

Figure (18) The SSE for 500 epochs using NN-41 and min-max normalization

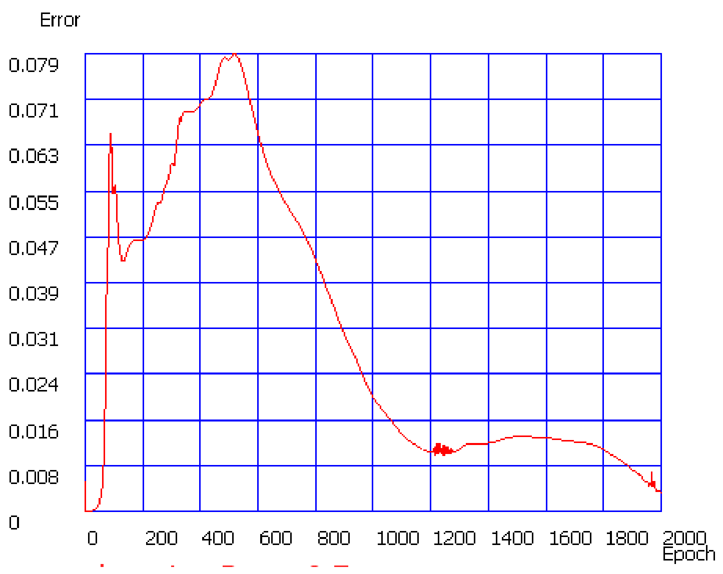

Figure (19) The SSE for 500 epochs using NN-41 and DS normalization

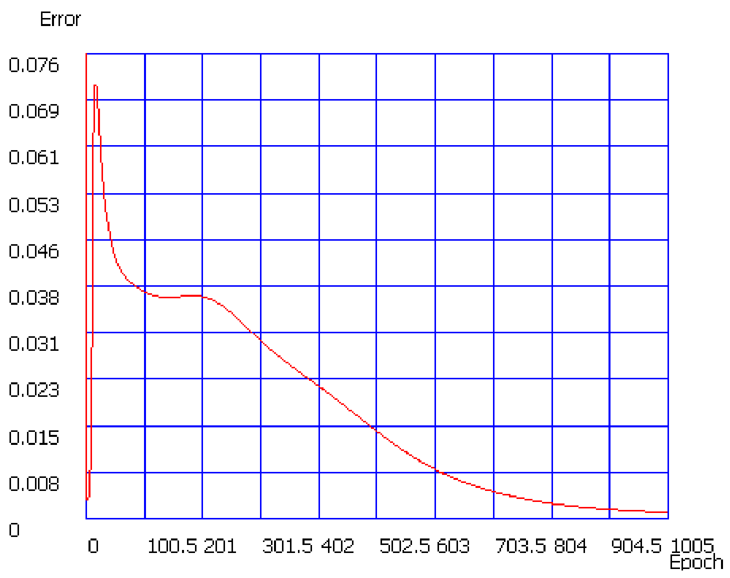

Figure (20) The SSE for 500 epochs using NN-41 and Log normalization