An overview of Intrusion Detection Using Soft Computing

Archana Sapkota and Palden Lama

Abstract— **Many hacking tools and intrusive methods have appeared along with a widespread use of computer networks. Using an intrusion detection system (IDS) is one way of dealing with suspicious activities within a network. It monitors activities of a given environment and decides whether these activities are malicious (intrusive) or legitimate (normal). Intrusion detection in general is either anomaly based or misuse based. In both cases, an IDS needs to make an intelligent decision usually based on pattern recognition and probabilistic reasoning. With this motivation, a lot of research efforts have explored the use of soft computing in the intrusion detection field. Soft computing is an innovative approach to construct a computationally intelligent system which parallels the extraordinary ability of the human mind to reason and learn in an environment of uncertainty and imprecision. Typically, soft computing consists of several computing paradigms, including neural networks, fuzzy sets, approximate reasoning, genetic algorithms, simulated annealing, etc. This paper will present an overview of various soft computing techniques used for intrusion detection and elaborate on a few recently applied techniques.**

Index Terms— **Intrusion Detection, Soft computing, SVM, DT, Fuzzy, LGP, Classifier**

I. INTRODUCTION

The first level of protection techniques in computer and network security used in various organizations are user authentication, data encryption, avoiding programming errors and firewalls. As network attacks have increased in number and severity over the past few years, the security infrastructure mentioned above are not sufficient to provide the level of security needed. Therefore, intrusion detection is required as an additional wall for protecting systems despite the prevention techniques. Intrusion detection is the process of monitoring the events occurring in a computer system or network and analyzing them for signs of intrusions, defined as attempts to compromise the confidentiality, integrity, availability, or to bypass the security mechanisms of a computer or network.

Intrusion detection systems (IDSs) are software or hardware systems that automate the process of monitoring the events occurring in a computer system or network, analyzing them for signs of security problems. Many ISDs available in the literature contain mainly three functional components. A) information sources B) Analysis C) Response/Notification.

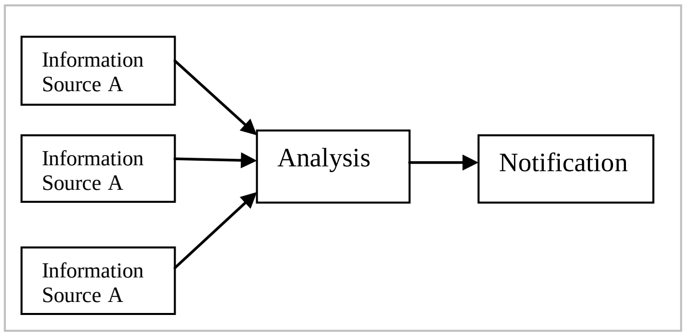


Figure 1: Components of IDS

Information Sources: The different sources of event information can be used to determine whether an intrusion has taken place. The information can be extracted from different levels of sources like network, host , application monitoring.

Analysis: This is the component of intrusion detection systems that actually organizes makes the decision based on the information source whether a particular event is an intrusion or not. The commonly used analysis approaches are misuse detection and anomaly detection.

Response: The set of actions that the system takes once the intrusion is detected. These are typically grouped into active and passive measures. Active measures involve some automated intervention on the part of the system, and passive measures involve reporting IDS findings to humans interacting with the systems.

Based on the information sources, there are mainly two kinds of IDS.
Host Based IDS(HIDS) : Host-based IDSs operate on information collected from within an individual computer system that is being monitored. Host-based IDSs normally utilize information sources of two types, operating system audit trails, and system logs. They periodically analyze logs, perform file system integrity check. Some examples of HIDS are: ISS RealSecure Server Sensor(Generic); Tripwire, AIDE(Check host file system); BlackICE, PortSentry (Check host network connections); LogSentry, Swatch (Check host's log files).

Network Based IDS(NIDS): NIDSs detect attacks by capturing and analyzing network traffic contents and patterns.

Some examples of NIDS are Snort, Cisco IDS4235. Based on the analysis techniques, there are mainly two kinds of IDSs: Misuse Detection and Anomaly Detection.

Misuse Detection: Uses well-defined patterns of the attack that exploit weaknesses in system and application software to identify the intrusions. These patterns are encoded in advance and used to match against the user behavior to detect intrusion.

Anomaly Detection: Uses the normal usage behavior patterns to identify the intrusion. The normal usage patterns are constructed from the statistical measures of the system features. The behavior of the user is observed and any deviation from the constructed normal behavior is detected as intrusion

With this background and different types of IDSs available in the literature, we are interested more in the 'Analysis' Component of the IDS where the main decision regarding whether a particular event is an intrusion or not. This component is very important because this determines the performance of IDS on the basis of false positives and false negatives. In out context the false positives mean something occurs that causes IDS to incorrectly identify an intrusion when none has occurred and the false positives mean something occurs that causes IDS to incorrectly fail to identify an intrusion when one has in fact occurred. Accuracy of IDS is determined by the number of false positives and completeness is determined by the number of false negatives.

Many techniques have been used for the analysis of intrusion detection. The main techniques used are statistical approaches, predictive pattern generation, expert systems, keystroke monitoring, model-based Intrusion detection, state transition analysis, pattern matching, and data mining techniques. Statistical approaches compare the recent behavior of a user of a computer system with observed behavior and any significant deviation is considered as intrusion. This approach requires construction of a model for normal user behavior. Any user behavior that deviates significantly from this normal behavior is flagged as an intrusion. Intrusion detection expert system (IDES) exploited the statistical approach for the detection of intruders. IDES maintains profiles, which is a description of a subject's normal behavior with respect to a set of intrusion detection measures. Profiles are updated periodically, thus allowing the system to learn new behavior as users alter their behavior. These profiles are used to compare the user behavior and informing significant deviation from them as the intrusion. IDES also uses the expert system concept to detect misuse intrusions. The advantage of this approach is that it adaptively learns the behavior of users, which is thus potentially more sensitive than human experts. This system has several disadvantages. The system can be trained for certain behavior gradually making the abnormal behavior as normal, which may make the intruders undetected. Determining the threshold above which an intrusion should be detected is a difficult task. Setting the threshold too low results in false positives and setting it too high results in false negatives.

Predictive pattern generation uses a rule base of user profiles defined as statistically weighted event sequences (Teng and Chen, 1990). This method of intrusion detection attempts to predict future events based on events that have already occurred. This system develops sequential rules of the form:

E1 - E2 - E3 -> (E4 = 94%, E5 = 6%)

This would mean that for the sequence of observed events E1 followed by E2 followed by E3, the probability of event E4 occurring is 94% and that of E5 is 6%. The rules are generated inductively with an information theoretic algorithm that measures the applicability of rules in terms of coverage and predictive power. An intrusion is detected if the observed sequence of events matches the left-hand side of the rule but the following events significantly deviate from the right-hand side of the rule. The main advantages of this approach include its ability to detect and respond quickly to anomalous behavior, easier to detect users who try to train the system during its learning period. The main problem with the system is its inability to detect some intrusions if that particular sequence of events have not been recognized and created into the rules.

The Keystroke monitoring technique utilizes a user's keystrokes to determine the intrusion attempt. The main approach is to pattern match the sequence of keystrokes to some predefined sequences to detect the intrusion. The main problems with this approach is a lack of support from the operating system to capture the keystroke sequences. Furthermore, there are also many ways of expressing the sequence of keystrokes for the same attack. Some shell programs like bash, ksh have the user definable aliases utility. These aliases make it difficult to detect the intrusion attempts using this technique unless some semantic analysis of the commands is used. Automated attacks by malicious executables cannot be detected by this technique as they only analyze keystrokes.

In literature, intrusion detection systems have been also been approached by various soft computing techniques. We are particularly interested in learning the role of soft computing in intrusion detection systems because unlike the traditional, hard computing, it is aimed at an accommodation with the pervasive imprecision of the real world. Thus, the guiding principle of soft computing is: '...exploit the tolerance for imprecision, uncertainty and partial truth to achieve tractability, robustness, low solution cost and better rapport with reality'. In the final analysis, the role model for soft computing is the human mind.

Section II describes some soft computing techniques that have been used in the literature for intrusion detection systems. Section III discusses various approaches of designing classifiers for intrusion detection. Section IV presents the database used for experiments. The experimental setup that have been used and results obtained in the different techniques are presented in Section V. The conclusion is given in Section VI.

II. Soft computing techniques for intrusion detection

Soft computing is not just one methodology but a consortium of various methodologies including machine learning, evolutionary computing, fuzzy logic, etc. These techniques can be applied separately or in combination for intrusion detection. In this section, we present a few selected techniques independently.

A. *Fuzzy Rule Based Systems*

Fuzzy logic has proved to be a powerful tool for decision making to handle and manipulate imprecise and noisy data. The notion central to fuzzy systems is that truth values (in fuzzy logic) or membership values (in fuzzy sets) are indicated by a value on the range [0.0, 1.0], with 0.0 representing absolute falseness and 1.0 representing absolute truth. A fuzzy system is characterized by a set of linguistic statements based on expert knowledge. The expert knowledge is usually in the form of if-then rules. It is commonly known as a paradigm for computing with words instead of numbers. It mimics the intuitive decision making capability of a human expert based on fuzzy information available. For instance, a human driver does not need to know the exact numeric values of the speed, direction and location of his car to park it successfully. His decisions and actions are based on fuzzy information such as “car is moving fairly slow, it is facing slightly right, it is near the parking spot”, etc. A simple rule structure of a fuzzy system is: “If antecedent then consequent” In the context of intrusion detection, a simple rule might be: “If variable1 is low and variable2 is high then output is benign else output is malignant”. In a fuzzy classification system, a case or an object can be classified by applying a set of fuzzy rules based on the linguistic values of its attributes. Every rule has a weight, which is a number between 0 and 1 and this is applied to the number given by the antecedent. It involves 2 distinct parts. First the antecedent is evaluated, which in turn involves fuzzifying the input and applying any necessary fuzzy operators and second applying that result to the consequent known as inference.

To build a fuzzy classification system, the most difficult task is to find a set of fuzzy rules pertaining to the specific classification problem. We present three fuzzy rule generation methods for intrusion detection systems. Let us assume that we have a n dimensional c-class pattern classification problem whose pattern space is an n-dimensional unit cube [0, 1]n. We also assume that m patterns $x_p = (x_{pl}, ..., x_{pn})$, p = 1,2,...,m, are given for generating fuzzy if-then rules where x_p ϵ [0,1] for p =1,2,..., m, i =1,2,...,n .

Rule Generation Based on the Histogram of Attribute Values (FR1):

In this method, use of histogram itself is an antecedent membership function. Each attribute is partitioned into 20 membership functions *fh(.)*, *h*=1,2,...,20. The smoothed histogram m_i^k (x) of class *k* patterns for the *ith* attribute is calculated using the 20 membership functions *fh (.)* as follows:

$$m_i^k(x_i) = \frac{1}{m^k} \sum_{x_p \in Class\ K} f_h\left(x_{pi}\right) \qquad (1)$$

The smoothed histogram in (1) is normalized so that its maximum value is 1. A single fuzzy *if-then* rule is generated for each class. The fuzzy if-then rule for the *kth* class can be written as:
If x_1 is A_k^1 and ... and *xn* is A_k^1 then class *k*,
where A_k^1 is an antecedent fuzzy set for the *ith* attribute. The membership function of A_k^1 is specified as

$$A_i^k(x_i) = e^{-(\frac{(x_i - \mu_i^k)^2}{2(\sigma_i^k)^2})}$$

Where μ_i^k is the mean of the i^{th} attribute values x_{pi} of class *k* patterns, and σ_i^k is the standard deviation. Fuzzy *if-then* rules for the two-dimensional two class pattern classification problem are written as follows:

If x_3 is A_3^1 and x_4 is A_4^1 *then* class 2
If x_3 is A_3^2 and x_4 is $\sqrt{a^2+b^2}$ *then* class 3

For a new pattern $x_p = (x_{p3}, x_{p4})$, the winner rule is determined as follows:

$$A_3^*(x_{p3}).A_2^*(x_{p4}) = \max\{A_1^k(x_{p3}).A_2^k(x_{p4}) \big| \, k = 1,2\}$$

Rule Generation Based on Partition of Overlapping Areas (FR2)

Figure 2 demonstrates a simple fuzzy partition, where the two dimensional pattern space is partitioned into 25 fuzzy subspaces by five fuzzy sets for each attribute (*S*: small, *MS*: medium small, *M*: medium, *ML*: medium large, *L*: large). A single fuzzy *if-then* rule is generated for each fuzzy subspace. Thus the number of possible fuzzy *if-then* rules in Figure 1 is 25.

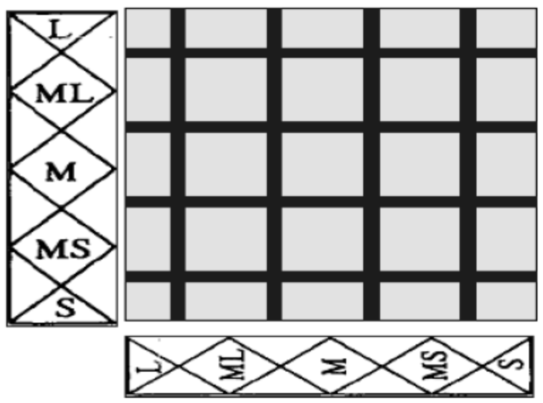


Figure 2: An example of fuzzy partition.

One disadvantage of this approach is that the number of possible fuzzy *if-then* rules exponentially increases with the dimensionality of the pattern space. Because the specification of each membership function does not depend on any information about training patterns, this approach uses fuzzy if-then rules with certainty grades. The FR2 approach generates fuzzy *if-then* rules in the same manner as the simple fuzzy grid approach except for the specification of each membership function. Because this approach utilizes the information about training patterns for specifying each membership function, the performance of generated fuzzy *if-then* rules is good even when we do not use the certainty grade of each rule in the classification phase. In this approach, the effect of introducing the certainty grade to each rule is not so important when compared to conventional grid partitioning.

Neural learning of fuzzy rules (FR3)

The derivation of *if-then* rules and corresponding membership functions depends heavily on the *a priori* knowledge about the system under consideration. However there is no systematic way to transform experiences of knowledge of human experts to the knowledge base of a Fuzzy Inference System (FIS). In a fused neuro-fuzzy architecture, neural network learning algorithms are used to determine the parameters of fuzzy inference system (membership functions and number of rules). Fused neuro-fuzzy systems share data structures and knowledge representations. A common way to apply a learning algorithm to a fuzzy system is to represent it in a special neural network-like architecture.

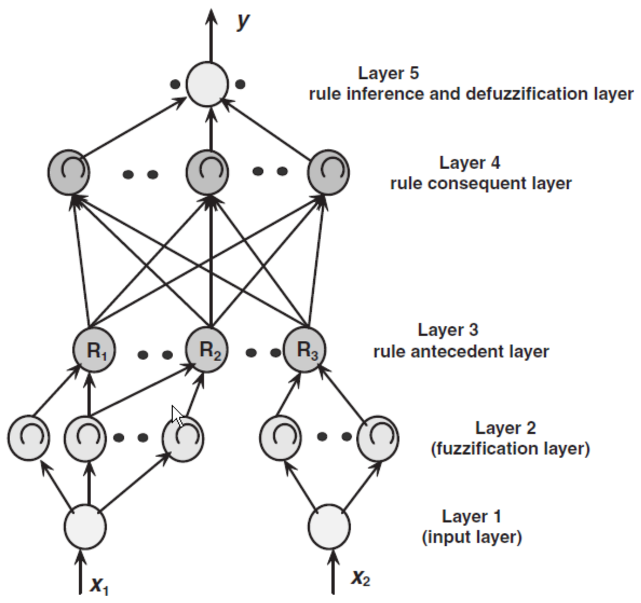


Figure 3: An integrated Neuro-Fuzzy system

Figure 3 shows a Mamdani type neuro-fuzzy system. It uses a supervised learning technique (backpropagation learning) to learn the parameters of the fuzzy membership functions.The detailed function of each layer is as follows:

Layer-1(input layer): No computation is done in this layer. Each node in this layer, which corresponds to one input variable, only transmits input values to the next layer directly. The link weight in layer 1 is unity.

Layer-2 (fuzzification layer): Each node in this layer corresponds to one linguistic label (excellent, good, etc.) to one of the input variables in layer 1. In other words, the output link represents the membership value, which specifies the degree to which an input value belongs to a fuzzy set, is calculated in layer 2. A clustering algorithm will decide the initial number and type of membership functions to be allocated to each of the input variable. The final shapes of the MFs will be fine tuned during network learning.

Layer-3 (rule antecedent layer): A node in this layer represents the antecedent part of a rule. Usually a T-norm operator is used in this node. The output of a layer 3 node represents the firing strength of the corresponding fuzzy rule.

Layer-4 (rule consequent layer): This node basically has two tasks. To combine the incoming rule antecedents and determine the degree to which they belong to the output linguistic label (high, medium, low, etc.). The number of nodes in this layer will be equal to the number of rules.

Layer-5 (combination and defuzzification layer): This node does the combination of all the rules consequents using a T-conorm operator and finally computes the crisp output after defuzzification.

B. Linear Genetic Programming

Linear genetic programming is a variant of the GP (Genetic Programming) technique that acts on linear genomes [3]. Its main characteristics in comparison to tree-based GP lies in that the evolvable units are not the expressions of a functional programming language (like LISP), but the programs of an imperative language (like C/C ++). An alternate approach is to evolve a computer program at the machine code level, using lower level representations for the individuals. This can tremendously hasten up the evolution process as, no matter how an individual is initially represented, finally it always has to be represented as a piece of machine code, as fitness evaluation requires physical execution of the individuals. The basic unit of evolution is a native machine code instruction that runs on the floating-point processor unit (FPU). Since different instructions may have different sizes, here instructions are clubbed up together to form instruction blocks of 32 bits each. The instruction blocks hold one or more native machine code instructions, depending on the sizes of the instructions. A crossover point can occur only between instructions and is prohibited from occurring within an instruction. However the mutation operation does not have any such restriction.

The settings of various LGP parameters are of utmost importance for successful performance of the system. Typically, the population space is subdivided into multiple subpopulation or demes. Migration of individuals among the subpopulations causes evolution of the entire population. It helps to maintain diversity in the population, as migration is restricted among the demes. Moreover, the tendency towards a bad local minimum in one deme can be countered by other demes with better search directions. The various LGP search parameters are the mutation frequency, crossover frequency and the reproduction frequency: The crossover operator acts by exchanging sequences of instructions between two tournament winners. A constant crossover rate of 90% is used for most experiments.

C. Decision Tree

A decision tree is made of decision nodes and leaf nodes. Each decision node corresponds to a test X over a single attribute of the input data and has a number of branches, each of which handles an outcome of the test X. Each leaf node represents a class that is the result of decision for a case. The process of constructing a decision tree is basically a divide and conquer process. Suppose a set T of training data consists of k classes (C1 , C2 ,…, Ck). If T only consists of cases of one single class, T will be a leaf. If T contains no case, T is a leaf and the associated class with this leaf will be assigned with the major class of its parent node. If T contains cases of mixed classes (i.e. more than one class), a test based on some attribute a_i of the training data will be carried and T will be split into n subsets (T1 , T2 , …, Tn), where n is the number of outcomes of the test over attribute a_i . The same process of constructing decision tree is recursively performed over each T j , where 1<= j <= n , until every subset belongs to a single class. The problem here is how to choose the best attribute for each decision node during construction of the decision tree. One possible criterion for choice is Gain Ratio Criterion. The basic idea of this criterion is to, at each splitting step, choose an attribute which provides the maximum information gain while reducing the bias in favor of tests with many outcomes by normalization. Once a decision tree is built, it can be used to classify testing data that has the same features as the training data. Starting from the root node of decision tree, the test is carried out on the same attribute of the testing case as the root node represents. The decision process takes the branch whose condition is satisfied by the value of tested attribute. This branch leads the decision process to a child of the root node. The same process is recursively executed until a leaf node is reached. The leaf node is associated with a class that is assigned to the test case.

Intrusion detection can be considered as classification problem where each connection or user is identified either as one of the attack types or normal based on some existing data. Decision trees work well with large data sets. This is important as large amounts of data flow across computer networks. The high performance of decision trees makes them useful in real-time intrusion detection. Decision trees construct easily interpretable models, which is useful for a security officer to inspect and edit. These models can also be used in the rule-based models with minimum processing. Generalization accuracy of decision trees is another useful property for intrusion detection model. There will always be new attacks on the system, which are small variations of known attacks after the intrusion detection models are built. The ability to detect these new intrusions is possible due to the generalization accuracy of decision trees.

D. Support Vector Machine

Support Vector Machines have been proposed as a novel technique for intrusion detection. SVM maps input (real-valued) feature vectors into a higher dimensional feature space through some nonlinear mapping. SVMs are powerful tools for providing solutions to classification, regression and density estimation problems. These are developed on the principle of structural risk minimization. Structural risk minimization seeks to find a hypothesis for which one can find the lowest probability of error. The structural risk minimization can be achieved by finding the hyper plane with maximum separable margin for the data. Computing the hyper plane to separate the data points, i.e. training a SVM, leads to a quadratic optimization problem. SVM uses a feature called a kernel to solve this problem. A kernel transforms linear algorithms into nonlinear ones via a map into feature spaces. SVMs classify data by using these support vectors, which are members of the set of training inputs that outline a hyper plane in feature space. Figure 4 shows a simple example of separable hyper plane between two data sets.

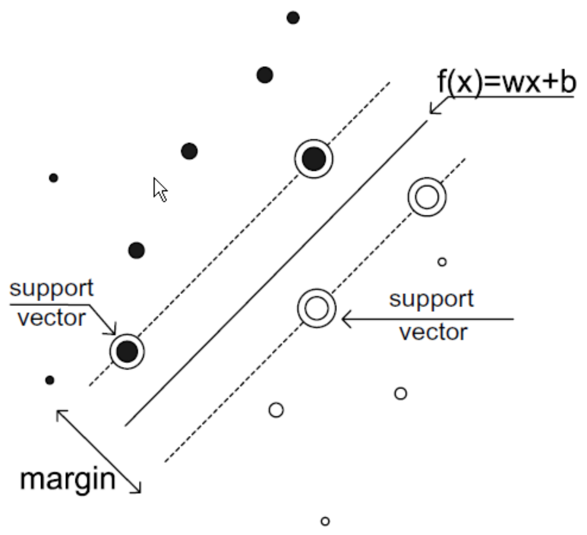


Figure 4. Separable hyper plane between two datasets.

III. CLASSIFIER DESIGN AND EVALUATION

A. Classifier Design

The classifiers for intrusion detection can be designed in various ways. In the literature people have used single classifier, hybrid classifiers and ensemble classifiers.

Single classifier: Single classifier system use one of the classification techniques like K – Nearest Neighbor, Artificial Neural Networks, Support Vector Machines, Self Organizing Map, Decision Tree, Bayes' Networks, Genetic Algorithms, Fuzzy Logic etc. Single classifiers are considered less robust than the ensemble or hybrid classifiers.

Hybrid Classifier: A hybrid intelligent system uses the approach of integrating different learning or decision-making models. Each learning model works in a different manner and exploits different set of features. Integrating different learning models gives better performance than the individual learning or decision-making models by reducing their individual limitations and exploiting their different mechanisms. In a hierarchical hybrid intelligent system each layer provides some new information to the higher level. The overall functioning of the system depends on the correct functionality of all the layers.

Ensemble Classifier: Empirical observations show that different classifiers provide complementary information about the patterns to be classified Although for a particular problem one classifier works better than the other, a set of misclassified patterns would not necessarily overlap. This different information combined together yields better performance than individual classifiers. The idea is not to rely on a single classifier for decision on an intrusion; instead information from different individual classifiers is combined to take the final decision, which is popularly known as the ensemble approach. The effectiveness of the ensemble approach depends on the accuracy of the base classifiers.

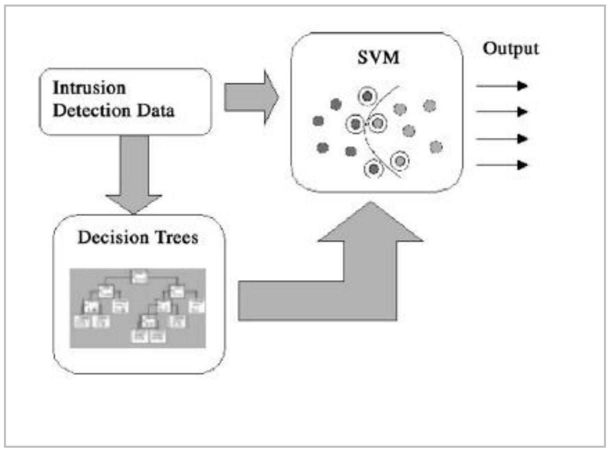


Figure 5: Example of hybrid classifier

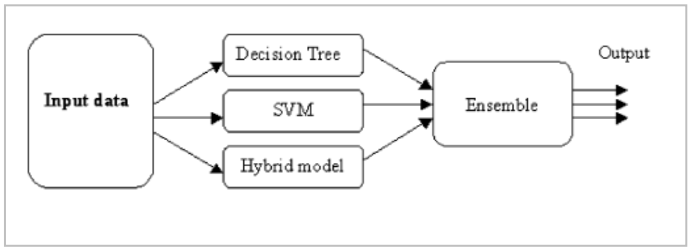


Figure 6: Example of Ensemble Classifier

B. *Evaluation strategies*

The three main steps involved in the classification and evaluation process using soft computing techniques are : Feature Selection, Training and Testing. Though there exist many techniques which do not require all three stages, but most of the literatures have used all three.

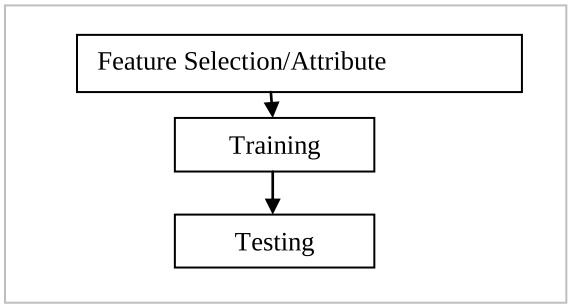


Figure 7: General Evaluation Strategy

Before training, the step of feature (or variable) selection may be considered. The process of feature selection identifies which features are more discriminative than the others. This has the benefit of generally improving system performance by eliminating irrelevant and redundant features. Feature selection is not very popular procedure in intrusion detection, however, few studies use different feature selection methods for their experiments and proved that that feature selection could improve some certain level of classification accuracy in intrusion detection. Generally while using the training and testing methods, a subset of the whole dataset is selected as a training set either randomly or using some criteria. The classifier is trained using this data and a subset of the whole data usually exclusive from the training set is selected as a test data. And the evaluation is based on its performance on test data give a trained classifier.

Table 1. Variables for intrusion detection data set

Variable No.	Variable name	Variable type	Variable label
1	duration	continuous	A
2	protocol_type	discrete	B
3	service	discrete	C
4	flag	discrete	D
5	src_bytes	continuous	E
6	dst_bytes	continuous	F
7	land	discrete	G
8	wrong_fragment	continuous	H
9	urgent	continuous	I
10	hot	continuous	J
11	num_failed_logins	continuous	K
12	logged_in	discrete	L
13	num_compromised	continuous	M
14	root_shell	continuous	N
15	su_attempted	continuous	O
16	num_root	continuous	P
17	num_file_creations	continuous	Q
18	num_shells	continuous	R
19	num_access_files	continuous	S
20	num_outbound_cmds	continuous	T
21	is_host_login	discrete	U

IV. DATABASE FOR EXPERIMENTS

A. 1998/1999 DARPA IDS dataset:

In 1998 DARPA recognized the need to be able to perform quantitative evaluations on intrusion detection systems. MIT's Lincoln Labs was contracted to work with the Air Force Research Laboratory in Rome, NY to build an evaluation dataset and perform an evaluation of the then current IDS research being funded by DARPA. They built a small network intending to simulate an Air Force base connected to the Internet, producing background activity with scripts, and injecting attacks at well defined points, and gathered tcpdump, Sun BSM, process and file system information. Lincoln Labs made the datasets available after the evaluations and many researchers used this data as the basis of evaluating their systems. However there have been many criticism against this dataset. The main criticism against this dataset was the failure to verify that the network in which data was collected realistically simulated a real-world network. However, researches say DARPA IDS evaluation dataset is still useful for testing intrusion detection systems for good performance but it is a necessary but not sufficient condition to demonstrate the capabilities of an advanced IDS[8].

B. KDD99

The 1999 KDD intrusion detection contest uses a version of DARPA IDS dataset. All the network traffic including the entire payload of each packet was recorded in tcpdump format and was provided for evaluation. The raw training data was about four gigabytes of compressed binary TCP dump data from seven weeks of network traffic. This was processed into about five million connection records. Similarly, the two weeks of test data yielded around two million connection records. A connection is a sequence of TCP packets starting and ending at some well defined times, between which data flows to and from a source IP address to a target IP address under some well defined protocol. Each connection is labeled as either normal, or as an attack, with exactly one specific attack type. Each connection record consists of about 100 bytes. Attacks fall into four main categories:

- DOS: denial-of-service, e.g. syn flood;
- R2L: unauthorized access from a remote machine, e.g. guessing password;
- U2R: unauthorized access to local superuser (root) privileges, e.g., various ``buffer overflow" attacks;
- probing: surveillance and other probing, e.g., port scanning.

But like DARPA 1999/1998 this database has also been criticized heavily.

```
0,tcp,http,SF,181,5450,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1
.00,0.00,0.00,9,9,1.00,0.00,0.11,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,239,486,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1.
00,0.00,0.00,19,19,1.00,0.00,0.05,0.00,0.00,0.00,0.00,0.00,normal
```

Positive Training Examples in KDD99

```
0,icmp,ecr_i,SF,1032,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,511,511,0.00,0.00,0.00,
0.00,1.00,0.00,0.00,255,255,1.00,0.00,1.00,0.00,0.00,0.00,0.00,0.00,smurf
0,icmp,ecr_i,SF,1032,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,511,511,0.00,0.00,0.00,
0.00,1.00,0.00,0.00,255,255,1.00,0.00,1.00,0.00,0.00,0.00,0.00,0.00,smurf.
```

Negative Training Examples in KDD99

C. Other datasets:

Few other datasets that have been used in the literature but are not very popular are: UNM, SSCNNJU, CUCS, RWND, PACCT, Windows system, network tcpdump data.

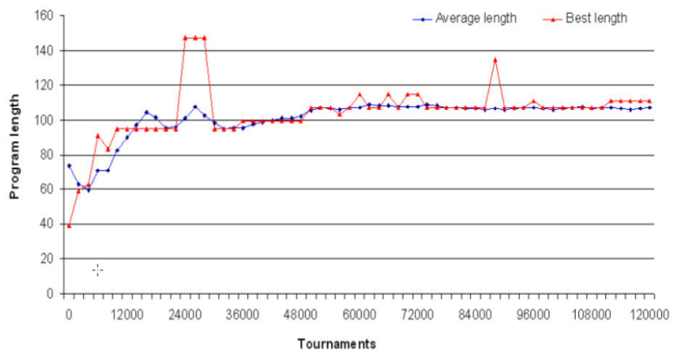
V. EXPERIMENTS AND RESULTS

We present the experimental results conducted in [1]. Using the original and reduced data sets described in the previous section, a 5-class classification was performed. The (training and testing) data set contains 11,982 randomly generated points from the data set representing the five classes, with the number of data from each class proportional to its size, except that the smallest class is completely included. The set of 5,092 training data and 6,890 testing data are divided in to five classes: normal, probe, denial of service attacks, user to super user and remote to local attacks. The datasets contain a total of 24 training attack types, with an additional 14 types in the test data only. Where the attack is a collection of different types of instances that belong to the four classes described earlier and the other is the normal data. The normal data belongs to class 1, probe belongs to class 2, denial of service belongs to class 3, user to super user belongs to class 4, remote to local belongs to class 5. All the IDS models are trained and tested with the same set of data.

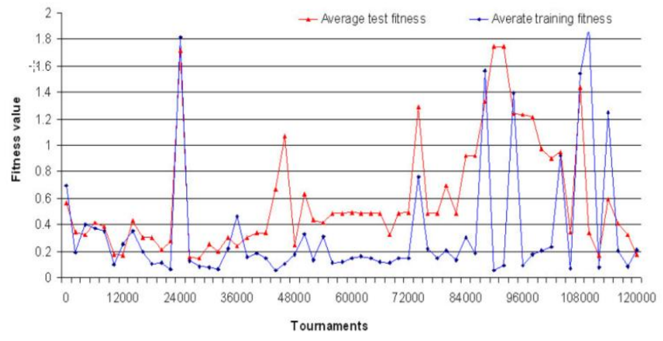
The performance of all three fuzzy rule based approaches (FR1, FR2 and FR3) mentioned in Section 3.1 were examined. When an attack is correctly classified the grade of certainty is increased and when an attack is misclassified the grade of certainty is decreased. A learning procedure is used to determine the grade of certainty. Triangular membership functions were used for all the fuzzy rule based classifiers. 4 triangular membership functions were used for each input variable for the neural network training of fuzzy inference system (FR3). A sensitivity threshold S_{thr} = 0.95 and error threshold Err_{thr} = 0.05 was used for all the classes.89 rule nodes were developed during the one pass learning. Table 1 depicts the parameter settings used for LGP experiments. The tournament size was set at 120,000 for all the 5 classes. Figure 7 demonstrates the growth in program length during 120,000 tournaments and the average fitness values for detecting normal patterns (class 1).

TABLE 1. PARAMETER SETTINGS OF LGP

Parameter	Normal	Probe	DoS	U2Su	R2L
Population size	2048	2048	2048	2048	2048
Tournament size	8	8	8	8	8
Mutation frequency (%)	85	82	75	86	85
Crossover frequency (%)	75	70	65	75	70
Number of demes	10	10	10	10	10
Maximum program size	256	256	256	256	256



(a)



(b)

Figure 7. LGP performance for the detection of normal patterns (a) growth in program length (b) average fitness

The trial experiments with SVM revealed that the polynomial kernel option often performs well on most of the datasets. The decision trees were constructed using the training data and then testing data was passed through the constructed classifier to classify the attacks.

A number of observations and conclusions are drawn from the results illustrated in Tables 2 and 3. Using 41 attributes, the FR2 method gave 100% accuracy for all the 5 classes, showing the importance of fuzzy inference systems. For the full data set, LGP outperformed decision trees and support vector machines in terms of detection accuracies (except for one class). The reduced dataset seems to work very well for most of the classifiers except the fuzzy classifier (FR2). For detecting U2R attacks FR2 gave the best accuracy.

TABLE 2. Performance comparison using full data set.

Attack type	Classification accuracy on test data set (%)					
	FR_1	FR_2	FR_3	DT	SVM	LGP
Normal	40.44	100.00	98.26	99.64	99.64	99.73
Probe	53.06	100.00	99.21	99.86	98.57	99.89
DOS	60.99	100.00	98.18	96.83	99.92	99.95
U2R	66.75	100.00	61.58	68.00	40.00	64.00
R2L	61.10	100.00	95.46	84.19	33.92	99.47

TABLE 3. Performance comparison using reduced data set.

Attack type	Classification accuracy on test data set (%)					
	FR_1	FR_2	FR_3	DT	SVM	LGP
Normal	74.82	79.68	99.56	100.00	99.75	99.97
Probe	45.36	89.84	99.88	97.71	98.20	99.93
DOS	60.99	60.99	98.99	85.34	98.89	99.96
U2R	94.11	99.64	65.00	64.00	59.00	68.26
R2L	91.83	91.83	97.26	95.56	56.00	99.98

Since a particular classifier could not provide accurate results for all the classes, an ensemble approach is demonstrated in Figure 5. The proposed ensemble model could detect all the attacks with high accuracy (lowest accuracy being 99.64%) with only 12 input variables. Ensemble performance is summarized in Table 4.

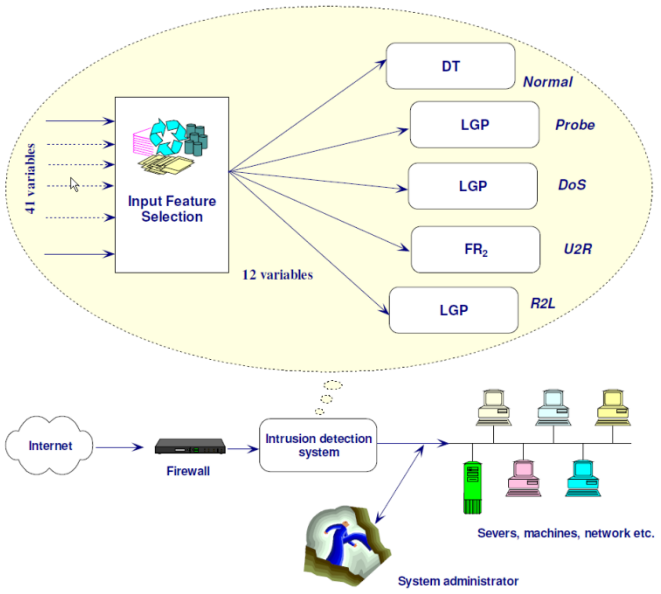


Figure 8. IDS architecture using an ensemble of intelligent paradigms.

For performance comparison of IDS based on soft computing approach with popular IDS available in market, we present the test results of SNORT on KDD data set in Table 5 [7].

TABLE 4. Performance of the ensemble model.

Attack type	Ensemble classification accuracy on test data (%)
Normal	100.00
Probe	99.93
DOS	99.96
U2R	99.64
R2L	99.98

TABLE 5. Performance of SNORT.

Attack type	Total attacks	Attacks detected	% detection
Probe	37	10	27%
DoS	63	30	48%
R2L	53	26	49%
U2R/Data	37	30	81%
Total	190	96	51%

VI. CONCLUSION

In this study of Intrusion Detection Systems, we analyzed the different kind of IDSs based on the three components, the data resources, the analyzer and the notifier. We were more interested in the analyzer and studied the different soft computing techniques available in the literature in the design of this component of IDS. We explored the evaluation strategy and experimental results presented by various authors on the performance of intrusion detection using soft computing techniques. Experimental results indicated that soft computing techniques performed well in detecting various types of attacks based on KDD99 dataset. A performance comparison with a popular IDS, SNORT, showed that IDS based on soft computing gave better performance in terms of intrusion detection rate. The reason behind its better performance is the self-learning and generalization capability inherent in soft computing techniques. However, recent studies claim that KDD99 dataset may not be good enough for evaluating intrusion detection systems. As a future work, it would be interesting to explore the performance evaluation of IDS based on soft computing for more realistic data sets.

REFERENCES

[1] A. Abraham and R. Jain, "Soft computing models for network intrusion detection systems", Studies in Computational Intelligence 16 (2005)

[2] Alshammari, R.; Sonamthiang, S.; Teimouri, M.; Riordan, D., "Using Neuro-Fuzzy Approach to Reduce False Positive Alerts," Fifth Annual Conference on Communication Networks and Services Research, 2007. CNSR '07. pp.345-349, 14-17 May 2007

[3] Brameier. M. and Banzhaf. W., "A comparison of linear genetic programming and neural networks in medical data mining," Evolutionary Computation," IEEE Transactions on, Volume: 5(1), pp. 17-26, 2001.

[4] K. Shah, N. Dave, S. Chavan, S. Mukherjee, A. Abraham, S. Sanyal, "Adaptive neuro-fuzzy intrusion detection system", in: Proceeding of IEEE International Conference on Information Technology: Coding and Computing

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

[6] M.S. Abadeh, J. Habibi and C. Lucas, "Intrusion detection using a fuzzy genetics-based learning algorithm", Journal of Network and Computer Applications (2005)

[7] Thomas, C.; Balakrishnan, N., "Performance enhancement of Intrusion Detection Systems using advances in sensor fusion," Information Fusion, 2008 11th International Conference on , vol., no., pp.1-7, June 30 2008-July 3 2008

[8] S.T. Brugger and J. Chow. An assessment of the DARPA IDS evaluation dataset using snort. Technical Report CSE-2007-1, University of California, Davis, Department of Computer Science, 2007.

[9] http://cs.uccs.edu/~chow/pub/ids/NISTsp800-31.pdf

[10] Chih-Fong Tsai, Yu-Feng Hsu, Chia-Ying Lin, Wei-Yang Lin,Intrusion detection by machine learning: A review, Expert Systems with Applications, Volume 36, Issue 10, December 2009, Pages 11994-12000, ISSN 0957-4174, DOI: 10.1016/j.eswa.2009.05.029.

[11] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[12] Peddabachigari, S., Abraham, A., Grosan, C., Thomas, J., 2007. Modeling intrusion detection system using hybrid intelligent systems.Journal of Network and Computer Applications 30 (1), 114–132.