Modeling Intrusion Detection Systems Using Linear Genetic Programming Approach

Srinivas Mukkamala, Andrew H. Sung, Ajith Abrham*
Department of Computer Science, New Mexico Tech, Socorro, NM 87801
*Department of Computer Science, Oklahoma State University, Tulsa, OK 74106
{srinivas,sung}@cs.nmt.edu, ajith.abraham@ieee.org

Abstract-This paper investigates the suitability of linear genetic programming (LGP) technique to model efficient intrusion detection systems, while comparing its performance with artificial neural networks and support vector machines. Due to increasing incidents of cyber attacks and, building effective intrusion detection systems (IDSs) are essential for protecting information systems security, and yet it remains an elusive goal and a great challenge. We also investigate key feature indentification for building efficient and effective IDSs. Through a variety of comparative experiments, it is found that, with appropriately chosen population size, program size, crossover rate and mutation rate, linear genetic programs could outperform support vector machines and neural networks in terms of detection accuracy. Using key features gives notable performance in terms of detection accuracies. However the difference in accuracy tends to be small in a few cases.

1 Introduction

Since most of the intrusions can be located by examining patterns of user activities and audit records, many IDSs have been built by utilizing the recognized attack and misuse patterns. IDSs are classified, based on their functionality, as misuse detectors and anomaly detectors. Misuse detection systems use well-known attack patterns as the basis for detection [1,2]. Anomaly detection systems make use user profiles as the basis for detection; any deviation from the normal user behavior is considered an intrusion [1,2,3,4]. One of the main problems with IDSs is the overhead, which can become unacceptably high. To analyze system logs, the operating system must keep information regarding all the actions performed, which invariably results in huge amounts of data, requiring disk space and CPU resource. Next, the logs must be processed to convert into a manageable format and then compared with the set of recognized misuse and attack patterns to identify possible security violations. Further, the stored patterns need be continually updated, which would normally involve human expertise. An intelligent, adaptable and cost-effective tool that is capable of (mostly) real-time intrusion detection is the goal of the researchers in IDSs. Various AI techniques have been utilized to automate the intrusion detection process to reduce human intervention; several such techniques include neural networks [3,4,5,6,7], and machine learning [8,11]. Several data mining techniques have been introduced to identify key features or parameters that define intrusions [8,9,10,11,12].

LGP has been already successfully implemented to a variety of machine learning problems [9]. This paper investigates the suitability of linear genetic programming technique, for modeling intrusion detection systems and indentifying the key features that help in deciding whether a connection is intrusive or normal activity. We also compare the performance of LGP with Support Vector Machines (SVM) and a Neural Network (NN) trained using resilient backpropagation learning. Performance metrics include a few critical aspects of intrusion detection like training and testing times, scalability and detection accuracy that help IDSs perform in real time or near real time. The data we used in our experiments originated from MIT's Lincoln Lab [14].

We perform experiments to classify the network traffic patterns according to a 5-class taxonomy. The five classes of patterns in the DARPA data are (normal, probe, denial of service, user to super-user, and remote to local). The experimental results of overall classification accuracy and class specific accuracies using linear genetic programs, support vector machines and resilient back propagation neural network are reported. Results obtained form feature ranking experiments are also reported with brief description of the most important features for each class.

In Section 2 a brief introduction to linear genetic programs, support vector machines and resilient back propagation algorithm is given. Section 3 briefly introduces the data we used to compare performance of different soft computing techniques. Section 4 presents the experimental results using LGP, SVM and NN. We briefly describe the feature selection/importance of ranking and the related results in section 5. The summary and conclusions of our work are given in section 6.

2 Linear Genetic Programming (LGP)

Linear genetic programming is a variant of the GP technique that acts on linear genomes [15,16]. 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.

2.1 Resilient Backpropagation (RBP)

The purpose of the resilient backpropagation training algorithm is to eliminate the harmful effects of the magnitudes of the partial derivatives. Only the sign of the derivative is used to determine the direction of the weight update; the magnitude of the derivative has no effect on the weight update. The size of the weight change is determined by a separate update value. The update value for each weight and bias is increased by a factor whenever the derivative of the performance function with respect to that weight has the same sign for two successive iterations. The update value is decreased by a factor whenever the derivative with respect that weight changes sign from the previous iteration. If the derivative is zero, then the update value remains the same. Whenever the weights are oscillating the weight change will be reduced. If the weight continues to change in the same direction for several iterations, then the magnitude of the weight change will be increased [17].

2.2 Support Vector Machines (SVMs)

The SVM approach transforms data into a feature space *F* that usually has a huge dimension. It is interesting to note that SVM generalization depends on the geometrical characteristics of the training data, not on the dimensions of the input space [18,19]. Training a support vector machine (SVM) leads to a quadratic optimization problem with bound constraints and one linear equality constraint. Vapnik shows how training a SVM for the pattern recognition problem leads to the following quadratic optimization problem [20].

Minimize: $$W(\alpha) = -\sum_{i=1}^{l}\alpha_i + \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j k(x_i, x_j) \qquad (1)$$

Subject to $$\sum_{i=1}^{l} y_i \alpha_i \qquad (2)$$
$$\forall i : 0 \le \alpha_i \le C$$

Where l is the number of training examples α is a vector of l variables and each component α_i corresponds to a training example (x_i, y_i). The solution of (1) is the vector α^* for which (1) is minimized and (2) is fulfilled.

3 Intrusion Detection Data

In the 1998 DARPA intrusion detection evaluation program, an environment was set up to acquire raw TCP/IP dump data for a network by simulating a typical U.S. Air Force LAN. The LAN was operated like a real environment, but being blasted with multiple attacks [21,22]. For each TCP/IP connection, 41 various quantitative and qualitative features were extracted [11,23]. Of this database a subset of 494021 data were used, of which 20% represent normal patterns.

Attack types fall into four main categories namely (1) Probing: surveillance and other probing (2) DoS: denial of service (3) U2Su: unauthorized access to local super user (root) privileges and (4) R2L: unauthorized access from a remote machine

3.1 Probing

Probing is a class of attacks where an attacker scans a network to gather information or find known vulnerabilities. An attacker with a map of machines and services that are available on a network can use the information to look for exploits. There are different types of probes: some of them abuse the computer's legitimate features; some of them use social engineering techniques. This class of attacks is the most commonly heard and requires very little technical expertise.

3.2 Denial of Service Attacks

Denial of Service (DoS) is a class of attacks where an attacker makes some computing or memory resource too busy or too full to handle legitimate requests, thus denying legitimate users access to a machine. There are different ways to launch DoS attacks: by abusing the computers legitimate features; by targeting the implementations bugs; or by exploiting the system's misconfigurations. DoS attacks are classified based on the services that an attacker renders unavailable to legitimate users.

3.3 User to Root Attacks

User to root (U2Su) exploits are a class of attacks where an attacker starts out with access to a normal user account on the system and is able to exploit vulnerability to gain root access to the system. Most common exploits in this class of attacks are regular buffer overflows, which are caused by regular programming mistakes and environment assumptions.

3.4 Remote to User Attacks

A remote to user (R2L) attack is a class of attacks where an attacker sends packets to a machine over a network, then exploits machine's vulnerability to illegally gain local access as a user. There are different types of R2U attacks; the most common attack in this class is done using social engineering.

4 Experiments

In our experiments, we perform 5-class classification. The (training and testing) data set contains 11982 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 5092 training data and 6890 testing data are divided in to five classes: normal, probe, denial of service attacks, user to super user and remote to local attacks. Where the attack is a collection of 22 different types of instances that belong to the four classes described in section 3, and the other is the normal, and the other is the normal data. The normal data belongs to class1, 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. Note two randomly generated separate data sets of sizes 5092 and 6890 are used for training and testing the LGPs, SVMs, and RBP respectively. Same training and test datasets were used for all the experiments. Tables 1,2 and 3 summarize the overall classification accuracy of LGPs, RBP and SVMs.

4.1 Linear Genetic Programming Training

LGP manipulates and evolves program at the machine code level [10]. The settings of various LGP parameters are of utmost importance for successful performance of the system. This section discusses the different parameter settings used for the experiment, justification of the choices and the significances of these parameters. The population space has been 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% has been used for all the simulations. After a trial and error approach, the parameter settings in Table 1 are used to develop IDS. Five LGP models are employed to perform five class classifications (normal, probe, denial of service, user to root and remote to local). We partition the data into the two classes of “Normal” and “Rest” (Probe, DoS, U2Su, R2L) patterns, where the Rest is the collection of four classes of attack instances in the data set. The objective is to separate normal and attack patterns. We repeat this process for all classes. Table 1 summarizes the results of the experiments using LGPs.

Table 1. Performance of LGPs

Class	Population Size	Program Size	Crossover Rate	Mutation Rate	Testing Accuracy (%)
Normal	1024	256	72.24	90.51	99.89
Probe	2048	512	51.96	83.30	99.85
DoS	2048	512	71.15	78.14	99.91
U2Su	2048	256	72.24	90.51	99.80
R2L	2048	512	71.15	78.14	99.84

4.2 RBP Experiments

In our study we used two hidden layers with 20 and 30 neurons each and the network is trained using resilient backpropogation neural network. As multi-layer feed forward networks are capable of multi-class classifications, we partition the data into 5 classes (Normal, Probe, Denial of Service, and User to Root and Remote to Local).We used testing data set (6890), network architecture [41 20 30 1] to measure the peformace of the RBP. The top-left entry of Table 2 shows that 1394 of the actual “normal” test set were detected to be normal; the last column indicates that 99.6 % of the actual “normal” data points were detected correctly. In the same way, for “Probe” 649 of the actual “attack” test set were correctly detected; the last column indicates that 92.7% of the actual “Probe” data points were detected correctly. The bottom row shows that 96.4% of the test set said to be “normal” indeed were “normal” and 85.7% of the test set classified, as “probe” indeed belongs to Probe. The overall accuracy of the classification is 97.04 with a false positive rate of 2.76% and false negative rate of 0.20.

Table 2. Performance of resilient back propagation neural network

	Normal	Probe	DoS	U2Su	R2L	%
Normal	1394	5	1	0	0	99.6
Probe	49	649	2	0	0	92.7
DoS	3	101	4096	2	0	97.5
U2Su	0	1	8	12	4	48.0
R2L	0	1	6	21	535	95.0
%	96.4	85.7	99.6	34.3	99.3	

4.3 SVM Experiments

Because SVMs are only capable of binary classifications, we employed five SVMs, for the 5-class classification problem.. We partition the data into the two classes of “Normal” and “Rest” (Probe, DoS, U2Su, R2L) patterns, where the rest is the collection of four classes of attack instances in the data set. The objective is to separate normal and attack patterns. We repeat this process for all classes. Training is done using the RBF (radial bias function) kernel option; an important point of the kernel function is that it defines the feature space in which the training set examples will be classified. Table 3 summarizes the results of the experiments using SVMs.

Table 3. Performance of SVMs

Class	Training time (sec)	Testing time (sec)	Accuracy (%)
Normal	7.66	1.26	99.55
Probe	49.13	2.10	99.70
DoS	22.87	1.92	99.25
U2Su	3.38	1.05	99.87
R2L	11.54	1.02	99.78

5. Ranking The Significance of Features

Feature selection and ranking [16,17] is an important issue in intrusion detection. Of the large number of features that can be monitored for intrusion detection purpose, which are truly useful, which are less significant, and which may be useless? The question is relevant because the elimination of useless features (the so-called audit trail reduction) enhances the accuracy of detection while speeding up the computation, thus improving the overall performance of an IDS. The feature ranking and selection problem for intrusion detection is similar in nature to various engineering problems that are characterized by:

a. Having a large number of input variables **x** = (x_1, x_2, …, x_n) of varying degrees of importance to the output **y**; i.e., some elements of **x** are essential, some are less important, some of them may not be mutually independent, and some may be useless or irrelevant (in determining the value of **y**)

b. Lacking an analytical model that provides the basis for a mathematical formula that precisely describes the input-output relationship, $\mathbf{y} = \boldsymbol{F}(\mathbf{x})$
c. Having available a finite set of experimental data, based on which a model (e.g. neural networks) can be built for simulation and prediction purposes

Due to the lack of an analytical model, one can only seek to determine the relative importance of the input variables through empirical methods. A complete analysis would require examination of all possibilities, e.g., taking two variables at a time to analyze their dependence or correlation, then taking three at a time, etc. This, however, is both infeasible (requiring 2^n experiments!) and not infallible (since the available data may be of poor quality in sampling the whole input space). We applied the technique of deleting one feature at a time (*16*) to rank the input features and identifying the most important ones for intrusion detection. Figures 1 and 2 gives the rank and importance towards calssification of each class respectively. Table 4 gives a brief description of the most important five features for each class. Feature ranking performance results for the individual classes are given in figures 3,4,5,6, and 7.

Fig. 1. Rank of each features for each class

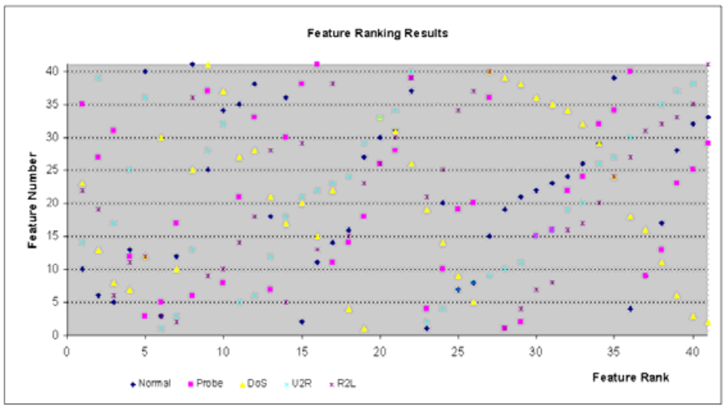


Fig. 2. Feature importance towards classification for each class

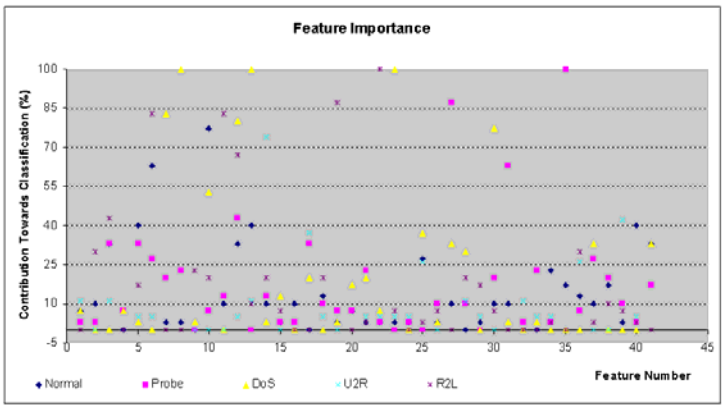


Fig. 3. Feature reduction results for normal

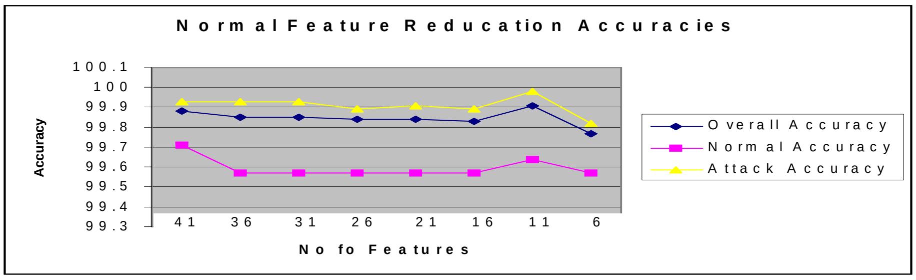


Table 4. Description of 5 most important features

Class	Feature Discription
Normal	▪ hot indicators: Number of "hot" indicators ▪ destination bytes: Number of bytes sent from the destination system to the host system ▪ source bytes: Number of bytes sent from the host system to the destination system ▪ compromised conditions: Number of compromised conditions ▪ dst_host_rerror_rate: % of connections that have REJ errors from a destination host
Probe	▪ dst_host_diff_srv_rate: % of connections to different services from a destination host ▪ rerror_rate: % of connections that have REJ errors ▪ srv_diff_host_rate: % of connections that have same service to different hosts ▪ logged in: binary decision ▪ service: type of service
DoS	▪ count: Number of connections made to the same host system in a given interval of time ▪ compromised conditions: Number of compromised conditions ▪ wrong_fragments: no of wrong fragments ▪ land: 1 if connection is from/to the same host/port; 0 otherwise ▪ logged in: 1 if successfully logged in; 0 otherwise
U2Su	▪ root shell: 1 if root shell is obtained; 0 otherwise ▪ dst_host_srv_serror_rate: % of connections to the same service that have SYN errors from a destination host ▪ no of file creations: no of file creation operations ▪ serror_rate: % of connections that have SYN errors ▪ dst_host_same_src_port_rate: % of connections to same service ports from a destination host
R2L	▪ guest login: 1 if the login is a "guest' login; 0 otherwise ▪ no of file access: no of operations on access control files ▪ destination bytes: Number of bytes sent from the destination system to the host system ▪ faile logins: no of failed login attempts ▪ logged in: binary decision

Fig. 4. Feature reduction results for probe

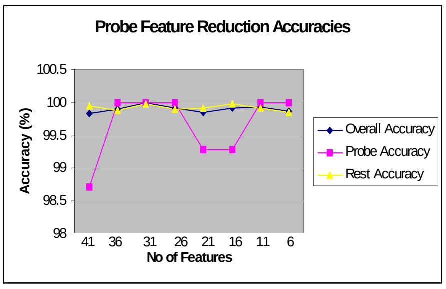


Fig. 5. Feature reduction results for DoS

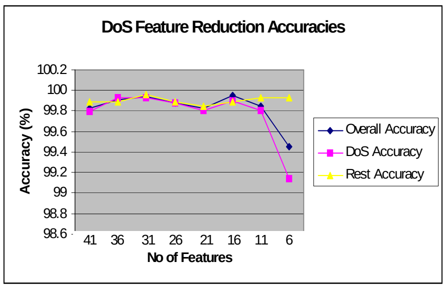


Fig. 6. Feature reduction results for U2Su

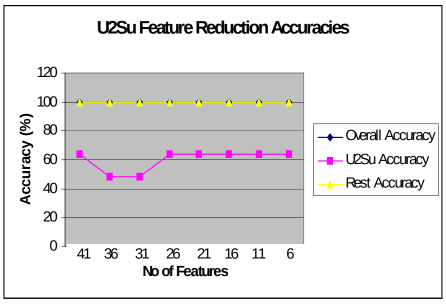
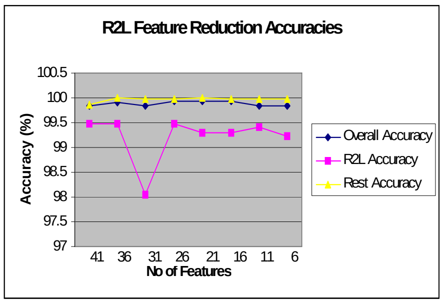


Fig. 7. Feature reduction results for R2L



6 Conclusions

Table 5 summarizes the overall performance of the three soft computing paradigms considered. LGPs outperform SVMs and RBP in terms of detection accuracies with the expense of time. SVMs outperform RBP in the important respects of scalability (SVMs can train with a larger number of patterns, while would ANNs take a long time to train or fail to converge); training time and prediction accuracy. Resilient back propagation achieved the best performance among the several other neural network learning algorithms we considered in terms of accuracy (97.04 %). The performances of using the important features for each class, give comparable performance with no significant differences, to that of using all 41 features.

Table 5. Performance comparison of testing for class specific classification

Class	SVMs Accuracy (%)	RBP Accuracy (%)	LGP Accuracy (%)
Normal	98.42	99.57	99.64
Probe	98.57	92.71	99.86
DoS	99.11	97.47	99.90
U2Su	64	48	64
R2L	97.33	95.02	99.47

We note, however, that the difference in accuracy figures tend to be very small and may not be statistically significant, especially in view of the fact that the 5 classes of patterns differ in their sizes tremendously. More definitive conclusions can only be made after analyzing more comprehensive sets of network traffic data.

References

1. Denning D. (1987) "An Intrusion-Detection Model," *IEEE Transactions on Software Engineering*, Vol. SE-13, No. 2, pp.222-232.
2. Kumar S., Spafford E. H. (1994) "An Application of Pattern Matching in Intrusion Detection," Technical Report CSD-TR-94-013. Purdue University.
3. Cannady J. (1998) "Applying Neural Networks for Misuse Detection," *Proceedings of 21st National Information Systems Security Conference*, pp.368-381.
4. Ryan J., Lin M-J., Miikkulainen R. (1998) "Intrusion Detection with Neural Networks," *Advances in Neural Information Processing Systems*, Vol. 10, Cambridge, MA: MIT Press.
5. Mukkamala S., Janoski G., Sung A. H. (2002) "Intrusion Detection Using Neural Networks and Support Vector Machines," *Proceedings of IEEE International Joint Conference on Neural Networks*, pp.1702-1707.
6. Stolfo J., Wei F., Lee W., Prodromidis A., and Chan P. K. (1999) "Cost-based Modeling and Evaluation for Data Mining with Application to Fraud and Intrusion Detection," *Results from the JAM Project by Salvatore.*
7. Mukkamala S., Sung A. H. (2002) "Identifying Key Features for Intrusion Detection Using Neural Networks," *Proceedings of ICCC International Conference on Computer Communications*, pp. 1132-1138.
8. http://www.ll.mit.edu/IST/ideval/data/data_index.html
9. Banzhaf. W., Nordin. P., Keller. E. R., Francone F. D. (1998) "Genetic Programming : An Introduction on The Automatic Evolution of Computer Programs and its Applications," Morgan Kaufmann Publishers, Inc.
10. AIMLearning Technology, http://www.aimlearning.com.
11. Brameier. M., Banzhaf. W. (2001) "A comparison of linear genetic programming and neural networks in medical data mining, Evolutionary Computation," IEEE Transactions on, Volume: 5(1), pp. 17-26.
12. Riedmiller M., and Braun H. (1993) "A direct adaptive method for faster back propagation learning: The RPROP algorithm", *Proceedings of the IEEE International Conference on Neural Networks*.
13. Joachims T. (1998) "Making Large-Scale SVM Learning Practical," LS8-Report, University of Dortmund, LS VIII-Report.
14. Joachims T. (2000) "SVMlight is an Implementation of Support Vector Machines (SVMs) in C," http://ais.gmd.de/~thorsten/svm_light. University of Dortmund. Collaborative *Research Center on Complexity Reduction in Multivariate Data (SFB475*).
15. Vladimir V. N. (1995) "The Nature of Statistical Learning Theory," *Springer*.
16. Kendall K. (1998) "A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems", *Master's Thesis, Massachusetts Institute of Technology*.
17. Webster S. E. (1998) "The Development and Analysis of Intrusion Detection Algorithms," *M.S. Thesis, Massachusetts Institute of Technology*.