Estimation of Distribution Algorithm for Optimization of Neural networks for Intrusion Detection System

Yuehui Chen[1] and Ajith Abraham[1,2]

[1]School of Information Science and Engineering
Jinan University, Jinan 250022, P.R.China
Email: yhchen@ujn.edu.cn
[2]School of Computer Science and Engineering
Chung-Ang University, Seoul, Republic of Korea
Email: ajith.abraham@ieee.org

Abstract. An Intrusion Detection System (IDS) is a program that analyzes what happens or has happened during an execution and tries to find indications that the computer has been misused. An IDS does not eliminate the use of preventive mechanism but it works as the last defensive mechanism in securing the system. This paper evaluates the performances of Estimation of Distribution Algorithm (EDA) to train a feedforward neural network classifier for detecting intrusions in a network. Results are then compared with Particle Swarm Optimization (PSO) based neural classifier and Decision Trees (DT). Empirical results clearly show that evolutionary computing techniques could play an important role in designing real time intrusion detection systems.

1 Introduction

Attacks on the nation's computer infrastructures are becoming an increasingly serious problem. Computer security is defined as the protection of computing systems against threats to confidentiality, integrity, and availability [1]. Confidentiality (or secrecy) means that information is disclosed only according to policy, integrity means that information is not destroyed or corrupted and that the system performs correctly, availability means that system services are available when they are needed. Computing system refers to computers, computer networks, and the information they handle. Security threats come from different sources such as natural forces (such as flood), accidents (such as fire), failure of services (such as power) and people known as intruders. There are two types of intruders: the external intruders who are unauthorized users of the machines they attack, and internal intruders, who have permission to access the system with some restrictions. The traditional prevention techniques such as user authentication, data encryption, avoiding programming errors and firewalls are used as the first line of defense for computer security. If a password is weak and is compromised, user authentication cannot prevent unauthorized use, firewalls are vulnerable to errors in configuration and ambiguous or undefined security policies. They are generally unable to protect against malicious mobile code, insider attacks and unsecured modems. Programming errors cannot be avoided as the complexity of the system and application software is changing rapidly leaving behind some exploitable weaknesses. Intrusion detection is therefore required as an additional wall for protecting systems. Intrusion detection is useful not only in detecting successful intrusions, but also provides important information for timely countermeasures. Intrusion detection is classified into two types: misuse intrusion detection and anomaly intrusion detection. Misuse intrusion detection uses well-defined patterns of the attack that exploit weaknesses in system and application software to identify the intrusions. Anomaly intrusion detection identifies deviations from the normal usage behavior patterns to identify the intrusion.

We have two options to secure the system completely, either prevent the threats and vulnerabilities which come from flaws in the operating system as well as in the application programs or detect them and take some action to prevent them in future and also repair the damage. It is impossible in practice, and even if possible, extremely difficult and expensive, to write a completely secure system. Transition to such a system for use in the entire world would be an equally difficult task. Cryptographic methods can be compromised if the passwords and keys are stolen. No matter how secure a system is, it is vulnerable to insiders who abuse their privileges. There is an inverse relationship between the level of access control and efficiency. More access controls make a system less user-friendly and more likely of not being used. An Intrusion Detection system is a program (or set of programs) that analyzes what happens or has happened during an execution and tries to find indications that the computer has been misused. An Intrusion detection system does not eliminate the use of preventive mechanism but it works as the last defensive mechanism in securing the system. Data mining approaches are a relatively new technique for intrusion detection. There are a wide variety of data mining algorithms drawn from the fields of statistics, pattern recognition, machine learning, and databases. Previous research of data mining approaches for intrusion detection model identified several types of algorithms as useful techniques. Classification is one of the data mining algorithms, which have been investigated as a useful technique for intrusion detection models.

Various intelligent paradigms namely Neural Networks [2], Support Vector Machine [3], Neuro-Fuzzy systems [4], Linear Genetic Programming [5], Flexible Neural Tree [6][7] and Decision Trees [9] have been used for intrusion detection. Various data mining techniques have been applied to intrusion detection because it has the advantage of discovering useful knowledge that describes a user's or program's behavior from large audit data sets.

This paper proposes an EDA based evolutionary neural network classifier for detecting intrusions. The weights, bias and flexible activation function parameters are optimized by EDA algorithm. Results are then compared with Particle Swarm Optimization (PSO) based neural classifier and Decision Trees (DT).

Table 1. The flexible activation functions

Gaussian Function	$f(x,a,b) = exp(-\frac{(x-a)^2}{b^2})$
Unipolar sigmoid function	$f(x,a) = \frac{2\lvert a\rvert}{1+e^{-2\lvert a\rvert x}}$
Bipolar sigmoid function	$f(x,a) = \frac{1-e^{-2xa}}{a(1+e^{-2xa})}$
Nonlocal radial coordinates	$f(x,a,b) = (b^2 + \lVert x-a\rVert^2)^{-\alpha}(\alpha > 0)$
General multi quadratics	$f(x,a,b) = (b^2 + \lVert x-a\rVert^2)^{\beta}(0 < \beta < 1)$
Thin-plate s-spline function	$f(x,a,b) = (b\lVert x-a\rVert)^2 \ln(b\lVert x-a\rVert)$

2 Neural Networks

A typical three-layer feedforward neural network consists of an input layer, a hidden layer and an output layer. The nodes are connected by weights and output signals, which are a function of the sum of the inputs to the node modified by a simple nonlinear activation function. The usually used activation function is the sigmoid function with threshold defined as

$$f(\sum_{i=1}^{n} w_i x_i - \theta) = \frac{1}{1 + exp(-(\sum_{i=1}^{n} w_i x_i - \theta))} \tag{1}$$

where x_i is the input to the node and w_i is the corresponding input weight, θ is a value which is usually called the threshold, n is the number of the inputs to the node. In this study, an flexible activation functions at hidden and output layers is selected. Some flexible activation functions shown in Table 1.

The output of a node is scaled by the connecting weight and is fed forward as an input to the nodes in the next layer of the network. The input layer plays no computational role but merely serves to pass the input vector to the network. The input layer and the hidden layer are connected by weights and likewise the hidden layer and output layer also have connection weights. The network has the ability to learn through training. The training requires a set of training data, i.e., a series of input and associated output vectors. During the training, the network is repeatedly presented with the training data and the weights and thresholds in the network are adjusted from time to time till the desired inputCoutput mapping occurs.

3 Neural Network Training by PSO and EDA

3.1 Estimation of Distribution Algorithm (EDA)

(EDA) [15][16][17][18][19][20] is a new class of EAs. EDA directly extracts the global statistical information about the search space from the search so far and builds a probabilistic model of promising solutions. New solutions are sampled from the model thus built. Several EDAs [17][18][19][20] have been proposed for the global continuous optimization problem. These algorithms are very promising, but much work needs to be done to improve their performances. An efficient evolutionary algorithm should make use of both the local information of solutions found so far and the global information about the search space. The local information of solutions found so far can be helpful for exploitation, while the global information can guide the search for exploring promising areas. The search in EDAs is mainly based on the global information, but DE on the distance and direction information which is a kind of local information. Therefore, it is worthwhile investigating whether combining DE with EDA could improve the performance of the DE algorithm and EDA.

One of the major issues in EDAs is how to select parents. A widely-used selection method in EDA is the truncation selection. In the truncation selection, individuals are sorted according to their objective function values. Only the best individuals are selected as parents.

Another major issue in EDAs is how to build a probability distribution model p(x). In EDAs for the global continuous optimization problem, the probabilistic model p(x) can be a Gaussian distribution [11], a Gaussian mixture [12][13], a histogram [14], or a Gaussian model with diagonal covariance matrix (GM/DCM) [12].

GM/DCM is used in our algorithm. In GM/DCM, the joint density function of the k-th generation is written as follows:

$$p_k(x) = \prod_{i=1}^{n} N(x_i; \mu_i^k, \sigma_i^k) \tag{2}$$

where

$$N(x_i; \mu_i^k, \sigma_i^k) = \frac{1}{\sqrt{(2\pi\sigma_i)}} exp(-\frac{1}{2}(\frac{x_i - \mu_i}{\sigma_i})^2) \tag{3}$$

In (2), the n-dimensional joint probability distribution is factorized as a product of n univariate and independent normal distributions. There are two parameters for each variable required to be estimated in the k-th generation: the mean, μ_i^k, and the standard deviation, σ_i^k. They can be estimated as follows:

$$\hat{\mu}_i^k = \bar{x}_i^k = \frac{1}{M}\sum_{j=1}^{M} x_{ji}^k \tag{4}$$

$$\hat{\sigma}_i^k = \sqrt{\frac{1}{M}\sum_{j=1}^{M}(x_{ji}^k - \bar{x}_i^k)^2} \tag{5}$$

Implementation of EDA for NN classifier. Before describing the details of EDA for training NN classifier, the issue of coding is presented. Coding concerns the way the weights, bias and the flexible activation function parameters of NN are represented by individuals. A float point coding scheme is adopted here. For NN coding, suppose there are M nodes in hidden layer and one node in output layer and n input variables, then the number of total weights is $n * M + M * 1$, the number of thresholds is $M + 1$ and the number of flexible activation function parameters is $M + 1$, therefore the total number of free parameters in a NN to be coded is $n * M + M + 2(M + 1)$. These parameters are coded into an individual or particle orderly.

Let $Pop(t)$ be the population of solutions at generation t. EDAs work in the following iterative way.

S1 Selection. Select M promising solutions from Pop(t) to form the parent set Q(t) by truncation selection method;
S2 Modeling. Build a probabilistic model p(x) based on the statistical information extracted from the solutions in Q(t);
S3 Sampling. Sample new solutions according to the constructed probabilistic model p(x);
S4 Replacement. Partly replace solutions in Pop(t) by the sampled new solutions to form a new population Pop(t + 1).

3.2 Parameter Optimization with PSO

The Particle Swarm Optimization (PSO) conducts searches using a population of particles which correspond to individuals in evolutionary algorithm (EA). A population of particles is randomly generated initially. Each particle represents a potential solution and has a position represented by a position vector $\mathbf{x_i}$. A swarm of particles moves through the problem space, with the moving velocity of each particle represented by a velocity vector $\mathbf{v_i}$. At each time step, a function f_i representing a quality measure is calculated by using $\mathbf{x_i}$ as input. Each particle keeps track of its own best position, which is associated with the best fitness it has achieved so far in a vector $\mathbf{p_i}$. Furthermore, the best position among all the particles obtained so far in the population is kept track of as $\mathbf{p_g}$. In addition to this global version, another version of PSO keeps track of the best position among all the topological neighbors of a particle. At each time step t, by using the individual best position, $\mathbf{p_i}$, and the global best position, $\mathbf{p_g(t)}$, a new velocity for particle i is updated by

$$\mathbf{v_i(t+1)} = \mathbf{v_i(t)} + c_1\phi_1(\mathbf{p_i(t)} - \mathbf{x_i(t)}) + c_2\phi_2(\mathbf{p_g(t)} - \mathbf{x_i(t)}) \tag{6}$$

where c_1 and c_2 are positive constant and ϕ_1 and ϕ_2 are uniformly distributed random number in [0,1]. The term $\mathbf{v_i}$ is limited to the range of $\pm\mathbf{v_{max}}$. If the velocity violates this limit, it is set to its proper limit. Changing velocity this way enables the particle i to search around its individual best position, $\mathbf{p_i}$, and global best position, $\mathbf{p_g}$. Based on the updated velocities, each particle changes its position according to the following equation:

$$\mathbf{x_i(t+1)} = \mathbf{x_i(t)} + \mathbf{v_i(t+1)}. \tag{7}$$

A neural network classifier trained by PSO algorithm with flexible bipolar sigmoid activation functions at hidden layer were constructed for the breast-cancer data set. The issue of coding is similar with the one used in EDA-NN discussed above.

The simple loop of the proposed training algorithm for neural network is as follows.

S1 Initialization. Initial population is generated randomly. The learning parameters c_1 and c_2 in PSO should be assigned in advance.

S2 Evaluation. The objective function value is calculated for each particle.

S3 Modification of search point. The current search point of each particle is changed using Eqn.(2) and Eqn.(1).

S4 if maximum number of generations is reached or no better parameter vector is found for a significantly long time (100 steps), then stop, otherwise goto step **S2**;

3.3 Decision Tree Classification

For comparison purpose, a Decision tree induction is one of the classification algorithms in data mining. The Classification algorithm is inductively learned to construct a model from the pre-classified data set. Each data item is defined by values of the attributes. Classification may be viewed as mapping from a set of attributes to a particular class. The Decision tree classifies the given data item using the values of its attributes. The decision tree is initially constructed from a set of pre-classified data. The main approach is to select the attributes, which best divides the data items into their classes. According to the values of these attributes the data items are partitioned. This process is recursively applied to each partitioned subset of the data items. The process terminates when all the data items in current subset belongs to the same class. A node of a decision tree specifies an attribute by which the data is to be partitioned. Each node has a number of edges, which are labeled according to a possible value of the attribute in the parent node. An edge connects either two nodes or a node and a leaf. Leaves are labeled with a decision value for categorization of the data.

Induction of the decision tree uses the training data, which is described in terms of the attributes. The main problem here is deciding the attribute, which will best partition the data into various classes. The ID3 algorithm [11] uses the information theoretic approach to solve this problem. Information theory uses the concept of entropy, which measures the impurity of a data items. The value of entropy is small when the class distribution is uneven, that is when all the data items belong to one class. The entropy value is higher when the class distribution is more even, that is when the data items have more classes. Information gain is a measure on the utility of each attribute in classifying the data items. It is measured using the entropy value. Information gain measures the decrease of the weighted average impurity (entropy) of the attributes compared with the impurity of the complete set of data items. Therefore, the attributes with the largest information gain are considered as the most useful for classifying the data items.

To classify an unknown object, one starts at the root of the decision tree and follows the branch indicated by the outcome of each test until a leaf node is reached. The name of the class at the leaf node is the resulting classification. Decision tree induction has been implemented with several algorithms. Some of them are ID3 [11] and later on it was extended into C4.5 [12] and C5.0. Another algorithm for decision trees is CART [13]. Of particular interest to this work is the C4.5 decision tree algorithm. C4.5 avoids over fitting the data by determining a decision tree, it handles continuous attributes, is able to choose an appropriate attribute selection measure, handles training data with missing attribute values and improves computation efficiency. C4.5 builds the tree from a set of data items using the best attribute to test in order to divide the data item into subsets and then it uses the same procedure on each sub set recursively. The best attribute to divide the subset at each stage is selected using the information gain of the attributes.

4 Simulation Studies

4.1 The Data Set

The data for our experiments was prepared by the 1998 DARPA intrusion detection evaluation program by MIT Lincoln Lab. The data set contains 24 attack types that could be classified into four main categories namely *Denial of Service (DOS)*, *Remote to User (R2L)*, *User to Root (U2R)* and *Probing.* The original data contains 744 MB data with 4,940,000 records. The data set has 41 attributes for each connection connection record plus one class label. Some features are derived features, which are useful in distinguishing normal from attacks. These features are either nominal or numeric. Some features examine only the connection in the past two seconds that have the same destination host as the current connection, and calculate statistics related to protocol behavior, service, etc. These called same host features. Some features examine only the connections in the past two seconds that have same service as the current connection and called same service features. Some other connection records were also stored by destination host, and features were constructed using a window of 100 connections to the same host instead of a time window. These called host-based traffic features. R2L and U2R attacks don't have any sequential patterns like DOS and Probe because the former attacks have the attacks embedded in the data packets whereas the later attacks have many connections in a short amount of time. So some features that look for suspicious behavior in the data packets like number of failed logins are constructed and these are called contents features. The data for our experiments contains randomly generated 11982 records having 41 features [8].

This data set has five different classes namely *Normal DOS*, *R2L*, *U2R* and *Probe.* The training and test comprises of 5092 and 6890 records respectively. All the IDS models were trained and tested with the same set of data. As the data set has five different classes we performed a 5-class binary classification.

The *normal* data belongs to class 1, *Probe* belongs to class 2, *DOS* belongs to class 3, *U2R* belongs to class 4 and *R2L* belongs to class 5.

4.2 Intrusion Detection by EDA-NN

A neural network classifier with structure {41-8-1} trained by EDA with flexible bipolar sigmoid activation functions were constructed using the training data sets and then the neural network classifier was used on the test data set to detect the different types of attacks. All the input variables were used for the experiments. Table 2 depicts the detection performance of EDA-NN for test data set.

4.3 Intrusion Detection by PSO-NN

For comparison purpose, a neural network classifier with structure {41-8-1} trained by PSO and with flexible bipolar sigmoid activation functions were also constructed using the same training data sets and then the neural network classifier was used on the test data set to detect the different types of attacks. All the input variables were used for the experiments. Table 2 depicts the detection performance of PSO-NN for test data set.

4.4 Intrusion Detection by DT

The important variables were also decided by their contribution to the construction of the decision tree. Variable rankings were generated in terms of percentages. We eliminated the variables that had 0.00% rankings and considered only the primary splitters or surrogates. This resulted in a reduced 12 variable data set with x_2, x_4, x_5, x_{11}, x_{22}, x_{23}, x_{24}, x_{27}, x_{30}, x_{31}, x_{32}, x_{34} as variables. The detection performance of the DT by using the original 41 variable data set is shown in Table 4.

Table 2. Detection performance using EDA-NN, PSO-NN and DT classification models for test data set

Attack Class	EDA-NN	PSO-NN	DT
Normal	**97.58%**	95.69%	82.32%
Probe	**95.57%**	95.53%	94.83%
DOS	**97.76%**	90.41%	77.10%
U2R	99.90%	**100%**	99.83%
R2L	**98.90%**	98.10%	94.33%

The achieved true positive and false positive rates using 41 input variables by the EDA-NN, PSO-NN and DT algorithms are depicted in Table 3.

Table 3. Comparison of false positive rate (fp) and true positive rate (tp) for EDA-NN, PSO-NN and DT classifiers for test data set

Attack	EDA-NN		PSO-NN		DT	
Class	fp(%)	tp(%)	fp(%)	tp(%)	fp(%)	tp(%)
Normal	0.29	99.64	4.3	88.70	29.66	99.60
Probe	0.02	56.57	0.40	37.15	0.24	31.00
DOS	3.94	98.86	3.68	89.38	72.10	97.63
U2R	0.01	52.00	0.05	55.81	0.022	59.26
R2L	0.08	87.39	0.17	86.63	0.022	30.73

5 Conclusions

In this paper, we have illustrated the importance of evolutionary algorithm and neural networks based techniques for modeling intrusion detection systems. EDA-NN classification accuracy is grater than 95% for all the considered attack types (classes) and achieved good true positive and false positive rates. It is to be noted that for real time intrusion detection systems EDA and neural networks would be the ideal candidates because of its simplified implementation.

References

1. R.C. Summers, Secure Computing: Threats and Safeguards. McGraw Hill, New York, 1997.
2. M. Debar, D. Becke, and A. Siboni. "A Neural Network Component for an Intrusion Detection System". *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, 1992.
3. S. Mukkamala, A.H. Sung and A. Abraham, "Intrusion Detection Using Ensemble of Soft Computing Paradigms", *Advances in Soft Computing, Springer Verlag, Germany*, pp. 239-248, 2003.
4. K. Shah, N. Dave, S. Chavan, S. Mukherjee, A. Abraham and S. Sanyal, "Adaptive Neuro-Fuzzy Intrusion Detection System", *IEEE International Conference on ITCC'04*, Vol. 1, pp. 70-74, 2004.
5. A. Abraham, Evolutionary Computation in Intelligent Web Management, Evolutionary Computing in Data Mining, A. Ghosh and L. Jain (Eds.), Studies in Fuzziness and Soft Computing, Springer Verlag Germany, Chapter 8, pp. 189-210, 2004.
6. Y. Chen, B. Yang, J. Dong, and A. Abraham, "Time-series Forecasting using Flexible Neural Tree Model", *Information Science*, 174(3-4): 219-235, 2005.
7. Y. Chen, A. Abraham, "Feature Selection and Intrusion Detection using Hybrid Flexible Neural Tree", ISNN-05, LNCS 3498, pp. 439-444, 2005.
8. KDD cup 99, http://kdd.ics.uci.edu/database/kddcup99/kddcup.data_10_percent.gz
9. S. Chebrolu, A. Abraham, J. P. Thomas, Feature Detection and Ensemble Design of Intrusion Detection Systems. *Computers and security*, Vol. 24/4, pp. 295-307, 2005.

10. Barbara D., Couto J., Jajodia S. and Wu N., ADAM: A Testbed for Exploring the Use of Data Mining in Intrusion Detection. *SIGMOD Record*, 30(4), pp. 15-24, 2001.
11. J. R. Quinlan. Induction of Decision Trees. Machine Learning, 1:81-106, 1986.
12. J. R. Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann, 1993.
13. Brieman L., Friedman J., Olshen R., and Stone C., Classification of Regression Trees. Wadsworth Inc., 1984.
14. D. Joo, T. Hong, I. Han, The neural network models for IDS based on the asymmetric costs of false negative errors and false positive errors, *Expert Systems with Applications*, Vol. 25, pp. 69-75, 2003.
15. A. Ochoa, H. Muhlenbein, M. Soto, A Factorized Distribution Algorithm Using Single Connected Bayesian Networks, *PPSN*, 787-796, 2000.
16. M. Pelikan, D. E. Goldberg and E. Cantu-Paz, BOA: The Bayesian Optimization Algorithm, *In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO99)*, I, 525-532, 1999.
17. S. Rudlof and M. Koppen, Stochastic Hill-Climbing with Learning by Vectors of Normal Distributions, Nagoya, Japan, 1996.
18. P. Larranaga and J. A. Lozano, Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation, Kluwer Academic Publishers, 2001.
19. P. A. N. Bosman and D. Thierens, Expanding from Discrete to Continuous EDAs: The IDEA, *In Proceedings of Parallel Problem Solving from Nature*, PPSN-VI, 767-776, 2000.
20. S. Tsutsui, M. Pelikan, and D. E. Goldberg, Evolutionary Algorithm Using Marginal Histogram Models in Continuous Domain, *In Proceedings of the 2001 Genetic and Evolutionary Computation Conference Workshop*, 230-233, San Francisco, CA, 2001.