# Feature Selection and Intrusion Detection using Hybrid Flexible Neural Tree

Yuehui Chen and Ajith Abraham

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

**Abstract.** Current Intrusion Detection Systems (IDS) examine all data features to detect intrusion or misuse patterns. Some of the features may be redundant or contribute little (if anything) to the detection process. The purpose of this study is to identify important input features in building an IDS that is computationally efficient and effective. This paper proposes an IDS model based on general and enhanced Flexible Neural Tree (FNT). Based on the pre-defined instruction/operator sets, a flexible neural tree model can be created and evolved. This framework allows input variables selection, over-layer connections and different activation functions for the various nodes involved. The FNT structure is developed using an evolutionary algorithm and the parameters are optimized by particle swarm optimization algorithm. Empirical results indicate that the proposed method is efficient.

## 1 Introduction

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.

Various intelligent paradigms namely Neural Networks [1], Support Vector Machine [2], Neuro-Fuzzy systems [3], Linear genetic programming [4] and Decision Trees [7] 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 papers proposes a Flexible Neural Tree (FNT) [5] for selecting the input variables and detection of network intrusions. Based on the pre-defined instruction/operator sets, a flexible neural tree model can be created and evolved. FNT allows input variables selection, over-layer connections and different activation functions for different nodes. In our previous work, the hierarchical structure was evolved using Probabilistic Incremental Program

Evolution algorithm (PIPE) with specific instructions. In this research, the hierarchical structure is evolved using tree-structure based evolutionary algorithm. The fine tuning of the parameters encoded in the structure is accomplished using particle swarm optimization (PSO). The proposed method interleaves both optimizations. Starting with random structures and corresponding parameters, it first tries to improve the structure and then as soon as an improved structure is found, it fine tunes its parameters. It then goes back to improving the structure again and, fine tunes the structure and rules' parameters. This loop continues until a satisfactory solution is found or a time limit is reached. The novelty of this paper is in the usage of flexible neural tree model for selecting the important features and for detecting intrusions.

## 2 The Flexible Neural Tree Model

The function set $F$ and terminal instruction set $T$ used for generating a FNT model are described as $S = F \bigcup T = \{+_2, +_3, \ldots, +_N\} \bigcup \{x_1, \ldots, x_n\}$, where $+_i (i = 2, 3, \ldots, N)$ denote non-leaf nodes' instructions and taking $i$ arguments. $x_1$,$x_2$,. . .,$x_n$ are leaf nodes' instructions and taking no other arguments. The output of a non-leaf node is calculated as a flexible neuron model (see Fig.1). From this point of view, the instruction $+_i$ is also called a flexible neuron operator with $i$ inputs.

In the creation process of neural tree, if a nonterminal instruction, i.e., $+_i (i = 2, 3, 4, \ldots, N)$ is selected, $i$ real values are randomly generated and used for representing the connection strength between the node $+_i$ and its children. In addition, two adjustable parameters $a_i$ and $b_i$ are randomly created as flexible activation function parameters. For developing the IDS, the flexible activation function $f(a_i, b_i, x) = e^{-(\frac{x-a_i}{b_i})^2}$ is used. The total excitation of $+_n$ is $net_n = \sum_{j=1}^{n} w_j * x_j$, where $x_j (j = 1, 2, \ldots, n)$ are the inputs to node $+_n$. The output of the node $+_n$ is then calculated by $out_n = f(a_n, b_n, net_n) = e^{-(\frac{net_n - a_n}{b_n})^2}$. The overall output of flexible neural tree can be computed from left to right by depth-first method, recursively.

**Tree Structure Optimization.** Finding an optimal or near-optimal neural tree is formulated as a product of evolution. In this study, the crossover and selection operators used are same as those of standard GP. A number of neural tree mutation operators are developed as follows:

(1) Changing one terminal node: randomly select one terminal node in the neural tree and replace it with another terminal node;
(2) Changing all the terminal nodes: select each and every terminal node in the neural tree and replace it with another terminal node;
(3) Growing: select a random leaf in hidden layer of the neural tree and replace it with a newly generated subtree.
(4) Pruing: randomly select a function node in the neural tree and replace it with a terminal node.

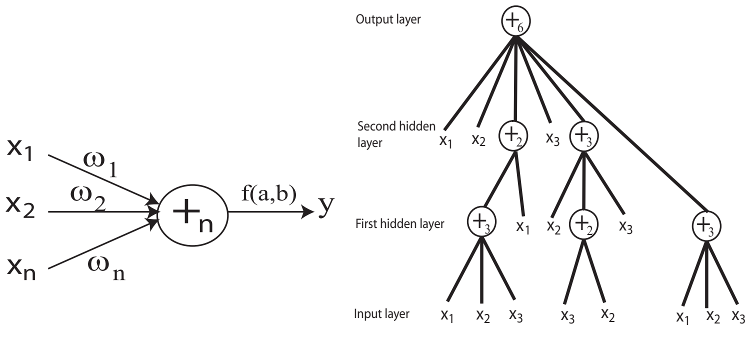

**Fig. 1.** A flexible neuron operator (left), and a typical representation of the FNT with function instruction set $F = \{+_2, +_3, +_4, +_5, +_6\}$, and terminal instruction set $T = \{x_1, x_2, x_3\}$ (right)

**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

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

where $c_1$ and $c_2$ are positive constant and $\phi_1$ and $\phi_2$ 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:

$$\mathbf{x_i(t+1)} = \mathbf{x_i(t)} + \mathbf{v_i(t+1)}. \tag{2}$$

**Procedure of the general learning algorithm.** The general learning procedure for constructing the FNT model can be described as follows.

1) Create an initial population randomly (FNT trees and its corresponding parameters);

2) Structure optimization is achieved by the neural tree variation operators as described in subsection 2.
3) If a better structure is found, then go to step 4), otherwise go to step 2);
4) Parameter optimization is achieved by the PSO algorithm as described in subsection 2. In this stage, the architecture of FNT model is fixed, and it is the best tree developed during the end of run of the structure search. The parameters (weights and flexible activation function parameters) encoded in the best tree formulate a particle.
5) If the maximum number of local search is reached, or no better parameter vector is found for a significantly long time then go to step 6); otherwise go to step 4);
6) If satisfactory solution is found, then the algorithm is stopped; otherwise go to step 2).

## 3 Feature Selection and Classification Using FNT Paradigms

**The Data Set.** The data for our experiments was prepared by the 1998 DARPA intrusion detection evaluation program by MIT Lincoln Lab and 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 data for our experiments contains randomly generated 11982 records having 41 features [6]. The training and test data comprises of 5092 and 6890 records respectively. All the IDS models were trained and tested with the same set of data. Since the data set has five different attack types 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.
**Feature/Input Selection with FNT.** It is often a difficult task to select important variables for any problem, especially when the feature space is large. A fully connected NN classifier usually cannot do this. In the perspective of FNT framework, the nature of model construction procedure allows the FNT to identify important input features in building an IDS that is computationally efficient and effective. The mechanisms of input selection in the FNT constructing procedure are as follows. (1) Initially the input variables are selected to formulate the FNT model with same probabilities; (2) The variables which have more contribution to the objective function will be enhanced and have high opportunity to survive in the next generation by a evolutionary procedure; (3) The evolutionary operators i.e., crossover and mutation, provide a input selection method by which the FNT should select appropriate variables automatically.
**Modeling IDS using FNT with 41 Input-Variables.** For this simulation, the original 41 input variables are used for constructing a FNT model. A FNT classifier was constructed using the training data and then the classifier was used on the test data set to classify the data as an attack or normal data. The instruction sets used to create an optimal FNT classifier is $S = F \bigcup T = \{+_5, \ldots, +_{20}\} \bigcup \{x_1, x_2, \ldots, x_{41}\}$. Where $x_i (i = 1, 2, \ldots, 41)$ denotes the 41 features.

**Table 1.** The important features selected by the FNT algorithm

| Class | Important variables |
|---|---|
| Class 1 | $x_3$, $x_{11}$, $x_{21}$, $x_{40}$ |
| Class 2 | $x_1$, $x_3$, $x_{12}$, $x_{18}$, $x_{20}$, $x_{21}$, $x_{23}$, $x_{26}$, $x_{27}$, $x_{31}$, $x_{37}$, $x_{41}$ |
| Class 3 | $x_1$, $x_8$, $x_{10}$, $x_{11}$, $x_{16}$, $x_{17}$, $x_{20}$, $x_{12}$, $x_{23}$, $x_{28}$, $x_{29}$, $x_{31}$ |
| Class 4 | $x_{11}$, $x_{14}$, $x_{17}$, $x_{28}$, $x_{29}$, $x_{32}$, $x_{36}$, $x_{38}$ |
| Class 5 | $x_1$, $x_3$, $x_{11}$, $x_{12}$, $x_{13}$, $x_{18}$, $x_{20}$, $x_{22}$, $x_{25}$, $x_{38}$ |

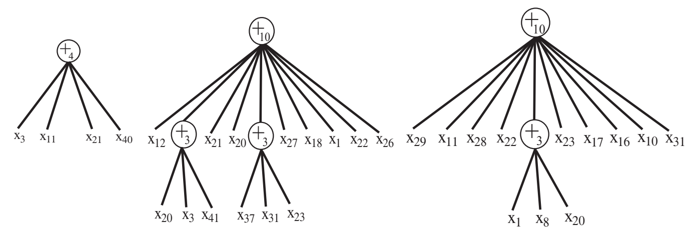
+4
x3 x11 x21 x40
+10
x12 +3 x21 x20 +3 x27 x18 x1 x22 x26
x20 x3 x41 x37 x31 x23
+10
x29 x11 x28 x22 +3 x23 x17 x16 x10 x31
x1 x8 x20

**Fig. 2.** The evolved FNT for class 1, 2 and 3 with 41 input variables.

The optimal FNTs for classes 1-5 are shown in Figures 2-3. It should be noted that the important features for constructing the FNT model were formulated in accordance with the procedure mentioned in the previous section. These important variables are shown in Table 1. Table 2 depicts the detection performance of the FNT by using the original 41 variable data set.

For comparison purpose, a neural network classifier trained by PSO algorithm were 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 and the results are shown in Table 2.

**Table 2.** Detection performance using FNT and NN classification models

| Attack Class | FNT | NN |
|---|---|---|
| Normal | **99.19%** | 95.69% |
| Probe | **98.39%** | 95.53% |
| DOS | **98.75%** | 90.41% |
| U2R | 99.70% | **100%** |
| R2L | **99.09%** | 98.10% |

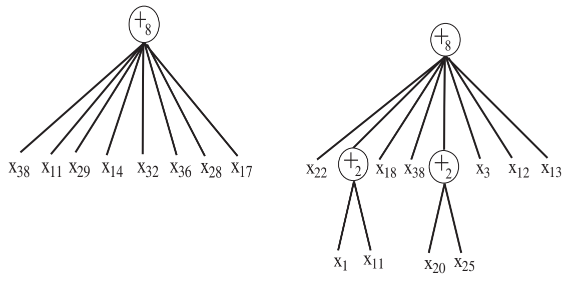
$+_8$
$x_{38}$ $x_{11}$ $x_{29}$ $x_{14}$ $x_{32}$ $x_{36}$ $x_{28}$ $x_{17}$
$+_8$
$x_{22}$ $+_2$ $x_{18}$ $x_{38}$ $+_2$ $x_3$ $x_{12}$ $x_{13}$
$x_1$ $x_{11}$ $x_{20}$ $x_{25}$

**Fig. 3.** The evolved FNT for class 4 and 5 with 41 input variables.

## 4 Conclusions

In this paper we presented a Flexible Neural Tree (FNT) model for Intrusion Detection Systems (IDS) with a focus on improving the intrusion detection performance by reducing the input features. We have also demonstrated the performance using different reduced data sets. As evident from Tables 1 and 2, the proposed flexible neural tree approach seems to be very promising. The FNT model was able to reduce the number of variables to 4, 12, 12, 8 and 10 (using 41 input variables) for classes 1-5 respectively. Using 41 variables, FNT model gave the best accuracy for the detection of most of the classes (except U2R). The direct NN classifier outperformed the FNT approach for U2R attack only.

## References

1. 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.
2. 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.
3. 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.
4. 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.
5. Y. Chen, B. Yang, J. Dong, and A. Abraham, "Time-series Forcasting using Flexible Neural Tree Model", *Information Science*, In press.
6. KDD cup 99, http://kdd.ics.uci.edu/database/kddcup99/kddcup.data_10_percent.gz
7. S. Chebrolu, A., Abraham, J. P., Thomas, Feature Detection and Ensemble Design of Intrusion Detection Systems. *Computers and security*, (http://dx.doi.org/10.1016/j.cose.2004.09.008) In press.
8. 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.

9. 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.