Performance Comparison of Neuro-Fuzzy Cloud Intrusion Detection Systems

Sivakami Raja[*] and Saravanan Ramaiah[#]
[*] Department of Information Technology, PSNA College of Engineering and Technology, Dindigul, Tamilnadu, India.
[#] Department of Computer Science and Engineering, RVS Educational Trust's Group of Institutions, Dindigul, Tamilnadu, India.

Abstract: *Cloud computing is a subscription-based service where we can obtain networked storage space and computer resources. Since access to cloud is through internet, data stored in clouds are vulnerable to attacks from external as well as internal intruders. In order to preserve privacy of the data in cloud, several intrusion detection techniques, authentication methods and access control policies are being used. The common intrusion detection systems are predominantly incompetent to be deployed in cloud environments due to their openness and specific essence. In this paper, we compare soft computing approaches based on type-1, type-2 and interval type-2 fuzzy-neural systems to detect intrusions in a cloud environment. Using a standard benchmark data from a CIDD (Cloud Intrusion Detection Dataset) derived from DARPA Intrusion Detection Evaluation Group of MIT Lincoln Laboratory, experiments are conducted and the results are presented in terms of mean square error.*

Keywords: *Fuzzy neural networks, Hybrid intelligent systems, Intrusion detection, Partitioning algorithms, Pattern analysis.*

1. Introduction

In this present era of computing, information is a major asset for every organization. From the local area network to the currently available highly connected internet, the world is being benefited from the easiness of data storage and access. With the introduction of cloud computing, maintenance of data has also become a simple task. However, this flexibility introduces a problem of data security as a major issue. This is due to the fact that, intruders and hackers are also enjoying technologies for their security-threatening activities. Data security and privacy are very serious problems in the victory of an industry process. Hence organizations are using different solutions to achieve data security.

An efficient intrusion detection system should be fast, self-monitored, fault tolerant, easily configurable, difficult to cheat, available without interruption and free from false errors with an overhead as minimum as possible. Its aim is to evaluate information systems and to perform early detection of malicious activity for reducing the security risk to an acceptably low level. It should generate alarms when it detects intrusive activity. High false positive alarm rate may disrupt information availability whereas high false negative alarm rate may result in serious damage to the protected systems in the form of inappropriate access to sensitive information and data damaging. It involves the overhead in terms of storage and CPU time. This is due to the fact that the performance of IDS is based on the amount of sufficient log data, its continuous updates on them and the correct and quick detection of intrusion from the comparison between current activity of the user and the historical data.

The rest of this paper is organized as follows: Section 2 outlines IDS in cloud. The concept of fuzzy neural architecture in intrusion detection is described in Section 3 and our proposed method is described in Section 4. Experimental results are presented in Section 5. Finally, Section 6 concludes our work.

2. Intrusion Detection in Cloud

For the purpose of achieving a green world and also due to the invention of new technologies, paper-work is vanishing from existence and the concept of digital data is introduced and is gaining importance. Large amount of personal information and potentially secure data can be stored on computers. But this digital data may require hardware, software and networks for their storage and access which may lead to practical complexities. A potential solution to these issues is a cloud storage service. This service offers several benefits including user-friendly access, maintenance and sharing of large volume of data synchronization of various devices and more importantly cost efficiency. By subscribing to the cloud, the cloud consumer is surrendering some control to an external source. This distance between the user and the physical location of data creates a hurdle. The information stored on the cloud is precious to individuals with malicious motive. Many cloud providers have standard terms and conditions to answer these issues. Firewalls and encryption are the most commonly used technologies for providing security and privacy. But they may not be of enough efficiency to protect data against intrusions. However, they can be used at the earlier part of the intrusion detection process. It is essential to choose a cloud provider that considers the security of cloud users' data as a major responsibility.

Machine learning is one of the methods used to train the system for intrusion detection. In [2], an IDS is modeled for networks using ANN and extended classifier system. A Grid and Cloud Computing Intrusion Detection System [33] was proposed to detect attacks by using an audit system. This system used artificial neural network to train the system and a prototype is developed using a middleware called Grid-M at the University Of Santa Catarina Brazil. An IDS with the central management approach [35] has been developed, addressing heterogeneity and virtualization features of cloud computing. A model based on hypervisor [31] has been proposed for protecting the system from different types of attacks in the infrastructure layer (IaaS) which proves improvement in the reliability and availability of the system. Various anomaly-based intrusion detection techniques [23] were employed, and offered an IDS named for SaaS with the conclusion that the anomaly-based ID as a hopeful technique to be used in the application layer. In [5], an individual IDS is suggested for each user of cloud computing services where a single controller manages the instances of IDSs exploiting the knowledge base and ANN pattern matching techniques. The fully distributed intrusion detection system developed in [12] is of P2P network architecture, implementing hybrid detection techniques using network and host based audit data for cloud computing. Most of the current proposed techniques on cloud operate at each of the infrastructure, platform, and application layers independently, and support detection independent from other layers [28].

Autonomic clouds have emerged as a result of employing autonomic computing techniques to cloud computing, resulting in fault tolerant and easy to operate cloud architectures and deployments. So autonomic computing solutions has contemporarily fascinated researchers to design, build and manage cloud intrusion detection engines with negligible human intervention. This system should be capable of accommodating its behavior to suit its context of use through methods of self-configuration, self-diagnosis, and self-healing [25]. An autonomic mechanism for anomaly detection in a cloud computing environment was proposed in [27]. This method provided a machine learning method for analyzing data, extracting features for dimensionality reduction, and detecting the nodes bearing abnormal behaviour. A virtualization-based NIDPS [8] for cloud computing environment which used network data flow monitoring and real time file integrity without any control over the host. Snort [32] was identified as a financially, technically and administratively easier tool to be implemented in small networks, but it is not cost efficient.

3. Neuro-Fuzzy Model

An artificial neural network (ANN) is a way of reasoning that is inspired by principles studied in neurons of the central nervous system of living things. It is a set of one or more layers of nodes interconnected by directed, weighted links and trained with desired input-output patterns through which it can learn weights on the links that provide the correct outputs for the training inputs and other similar inputs. The ANN can be used to create a Decision Support System (DSS) when sufficient training examples exist. Though it suffers from relatively slow learning process, the benefits are its learning, adaptation, fault-tolerance, parallelism and generalization. Fuzzy logic is a technique for the representation and handling of imprecise and vague information. A common approach of using fuzzy logic in a DSS is creating a fuzzy rule based system that can symbolize domain knowledge in the form or rules and can make numerical calculations. Except it takes a lot of time to design and tune the membership functions which quantitatively define these linguistic labels, it can achieve stable state in a quick time interval. And also it can reason with imprecise information. Unfortunately, it does not have an efficient learning algorithm to improve the member functions to minimize the output errors. To complement each of the above individual systems with their advantages, fuzzy neural systems or neuro-fuzzy models were developed as intelligent hybrid systems which are proving their effectiveness in a wide variety of real world applications.

Neural-fuzzy systems take up type-1 fuzzy sets, which characterize uncertainties using numbers in the range [0, 1]. However, the context of the words that are used in the rules and the measurements that activate a type-1 system can be uncertain and the data used to tune the parameters of a type-1 fuzzy logic system may also be noisy [20]. Membership functions of type-1 fuzzy sets are often very exact and require each element of the universal set to be assigned a particular real number. But, type-2 fuzzy sets allow modeling such uncertainties because their membership functions are expressed as type-1 fuzzy sets. The concept was proposed as a development of an ordinary fuzzy set [21, 22, 24, 38]. The centroid of a type-2 fuzzy set and a practical algorithm for its computation are developed [10, 11] for interval type-2 fuzzy sets. Inference involving interval type-2 fuzzy sets is simple and less time consuming than that involving type- 2 fuzzy sets. A complete theory for interval type-2 fuzzy sets, tuning of free parameters within interval type-2 fuzzy sets using training data are explained in [15]. In [16, 19] interval type-2 Takagi-Sugeno-Kang (TSK) fuzzy logic systems are explained and the implementation of interval type-2 fuzzy logic systems are explained in [26]. The application of type-2 fuzzy systems requires type reduction which reduces a type-2 fuzzy set to a type-1 fuzzy set followed by defuzzification of the resulting type-1 fuzzy set which gives the desired crisp number.

4. Proposed Intrusion Detection System

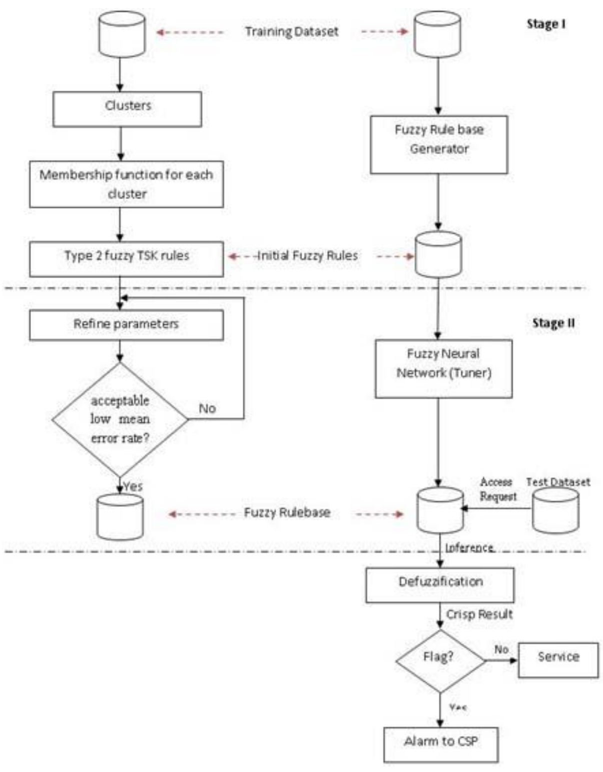

Figure 1. Proposed IDS.

Figure 1 shows our proposed system of intrusion detection in cloud environment. For extracting fuzzy rules from a set of data (for Stage I shown in Figure 1), several methods were proposed.

A fuzzy partitioning method [17] is proposed to create a set of fuzzy rules from input-output data by dividing the input space into a set of subspaces. But it is hard to decide the locations of cuts and the time complexity is high. In a hybrid clustering and gradient descent approach [36], convergence is very slow when the amount of the training data is enormous. Another method [30] for extracting fuzzy rules from a set of training patterns is done by splitting the universe of the output variable into multiple clusters, and then by assigning a cluster to each of the training data according to the desired value of the output variable. The data of each cluster in the input set is partitioned. This process of partitioning is concluded according to a given criterion to prevent exaggerated partition. This method needs to solve the problem of overlapping among different clusters.

For the construction of refined fuzzy rules from the initial set of fuzzy rules (for Stage II shown in Figure 1), parameters of the rules have to be refined. Numerous methods for refining the parameters of type-2 fuzzy rules were proposed. Type-2 fuzzy neural network (T2FNN) [6, 34] is formed by combining type-2 fuzzy logic system and neural network using gradient descent for parameters refinement. A recurrent neural network for interval type-2 fuzzy rules using asymmetric Gaussian principal membership functions is proposed in [14] with the refining algorithm. A self-evolving interval type-2 fuzzy neural network (IT2FNN) [9] is proposed with online structure. This approach combines online clustering and rule-ordered Kalman filter algorithm. In [3], three interval type-2 fuzzy neural network architectures were proposed through gradient descent backpropagation and gradient descent with adaptive learning rate backpropagation for refining the parameters.

To develop initial fuzzy rules from the given set of input–output training data, a self-constructing fuzzy clustering algorithm is applied to separate the training data into a group of fuzzy clusters. Then, these clusters are transformed into a rule base of type-2 fuzzy TSK IF-THEN rules. The clustering algorithm [7, 37] is a learning approach which follows incremental self-construction. It partitions the training dataset into clusters through input-similarity and output-similarity tests. For each pattern, the similarity to each existing cluster is calculated to determine whether it has to be associated with an existing cluster or a new cluster has to be created. If the pattern is added into an existing cluster, the membership function of that cluster is revised. Else if a new cluster is created, and the corresponding membership function is initialized.

Assume the system to be modeled has k inputs $a_1, a_2, ..., a_k$, and one output b. The given training dataset contains r patterns $[p_1, d_1]$, $[p_2, d_2]$,..., $[p_r, d_r]$ where $p_j = [p_{1j}, p_{2j}, \cdots, p_{kj}]$ denotes the k input values and d_j denotes the corresponding desired output value of the j^{th} pattern, $1 \le j \le r$. Let n be the number of currently existing clusters represented by $C_1, C_2, \cdots, C_n$. Initially no clusters exist and hence $n = 0$. Each cluster C_i has mean $m_i = [m_{1i}, m_{2i}, \cdots, m_{ki}]$, deviation $\sigma_i = [\sigma_{1i}, \sigma_{2i}, \cdots, \sigma_{ki}]$ and a height h_i which is the average of the desired outputs of the patterns enclosed in the cluster. Let S_i be the number of training patterns available in S_i .

For each pattern $p_i = [p_{1i}, p_{2i}, \cdots, p_{ki}], 1 \le i \le r$, the similarity of p_i to each existing cluster C_j, $1 \le j \le n$ is calculated by $\mu_{C_j}(p_i) = \prod_{l=1}^{k} \exp\left[-\left(\frac{p_{li} - m_{lj}}{\sigma_{lj}}\right)^2\right]$. This pattern p_i clears the input similarity test on cluster C_j if $\mu_{C_j}(p_i) \ge \rho$, where ρ is a predefined threshold and $0 \le \rho \le 1$. The value of ρ decides the number of clusters in a directly proportional way. When ρ increases, it builds the boundaries of the Gaussian function sharper and hence clusters are formed in micro level.

The output similarity of pattern p_i to each existing cluster C_j is calculated by computing $e_j = |d_i - h_j|$ and this pattern passes the output similarity test on cluster C_j if $e_j \le \tau(d_{high} - d_{low})$, where d_{high} and d_{low} represent the highest and the lowest values of the desired outputs. The expression $(d_{high} - d_{low})$ sets the limit on the maximum allowed deviation between the actual and the desired results during output similarity test. If the achieved result does not fall within this limit, it is concluded that the current pattern does not clear the output similarity test. So, these two parameters put a restriction on the output similarity test to improve accuracy. Here, τ, $0 \le \tau \le 1$, is a predefined threshold. The value of τ decides the number of clusters in an inversely proportional way. When τ increases, the test gets tougher and less number of clusters will be created.

If a pattern p_i has not passed both tests for all clusters, then a new cluster C_{n+1} is created by incrementing n by 1 and initializing m_n to p_i, σ_n to σ_0 and h_n to d_i . On the contrary, if there are existing clusters on which p_i has passed the similarity tests, the cluster with the largest membership degree g is chosen as C_g. Then m_g, σ_g, and h_g are modified according to the following four equations:

$$m_{jg}^{v} = \frac{\sum_{l=1}^{S_g^v} p_{jl}}{S_g^v} = \frac{S_g^o \times m_{jg}^o + p_{ji}}{S_g^o - 1} \qquad (1)$$

$$\sigma_{jg}^{v} = \sqrt{\frac{\sum_{l=1}^{S_g^v}\left(p_{jl} - m_{jg}^v\right)^2}{S_g^v - 1}} \qquad (2)$$

$$h_g^v = \frac{\sum_{l=1}^{S_g^v} d_l}{S_g^v} \qquad (3)$$

$$S_g^v = S_g^o + 1 \qquad (4)$$

Here, the superscripts o and v represent the values before and after modification. When all training patterns have been considered, we get a number of clusters that are then converted to type-2 fuzzy TSK IF-THEN rules. At first all data are normalized into the range [0, 1] and then each cluster C_i is converted into a type-2 fuzzy rule of the form:

If a_1 is $\tilde{A}_{1i}$ and a_2 is $\tilde{A}_{2i}$ and … and a_k is $\tilde{A}_{ki}$

Then b is $c_i = \omega_{0i} + \omega_{1i}a_1 + \cdots + \omega_{ki}a_k$

where $\tilde{A}_{1i}, \tilde{A}_{2i}, \cdots \tilde{A}_{ki}$ are type-2 fuzzy sets for a_1, $a_2, \ldots, a_k$ and each ω is a real valued rule weight. The membership function of $\tilde{A}_{ij}$, $1 \le i \le k$, is $\mu_{\tilde{A}_{ij}}(a_i) = gauss(u; gauss(a_i; m_{ij}^p, \sigma_{ij}^p), \sigma_{ij}^s$, where $u \in [0,1]$. Fuzzy systems comprise of several types of membership functions. Out of them, Gaussian membership function is used in our system due to its greater continuous smoother transition in intervals, straightforward learning laws and reduced degree of freedom. Additionally, it always has nonzero values. So, every rule in the rulebase gets fired. Hence, our system will not have dead rules. And more importantly, it provides the basic things to generate hybrid systems like fuzzy neural systems and gives more precise results.

Each rule includes mean, deviation and the scaled deviation as its antecedent parameters and the rule weight as its consequent parameters. The weight is tied to a rule to determine a degree of fulfillment by multiplying with antecedents. We proceed to improve the precision of these rules, by refining the antecedent and consequent parameters involved, through the application of a dynamical optimal learning algorithm. An iteration of learning involves the presentation of all training patterns. In each iteration of learning, we first treat all the consequent parameters as fixed to refine the antecedent parameters. Then we treat all the antecedent parameters as fixed and use to refine the consequent parameters. The process is iterated until the desired approximation precision is achieved. Due to the computational complexity in using type-2 fuzzy sets, interval type-2 fuzzy sets (IT2FNN) are used. As mentioned earlier, we treat all the consequent parameters as fixed and use interval type-2 fuzzy neural network architecture with gradient descent learning to refine the antecedent parameters. To refine consequent parameters, we treat all antecedent parameters as fixed. The objective function to be optimized is the mean square error (MSE) with respect to the training patterns. By using the BP method, for input–output training data $[p_j : d_j]$, $j = 1,2,\cdots,r$ the following error function can be minimized:

$$e_j = \frac{1}{2}\left[y(p_j) - d_j\right]^2 \tag{5}$$

We use equation (6) to refine mean and a similar type of equation is used to refine standard deviation by keeping the weight as fixed.

$$m^i_{j1}(l+1) = m^i_{j1}(l) - \left(\frac{\alpha\left(y(p_l)-d_l\right)\left(p_{jl}-m^i_{j1}\right)N\left(m^i_{j1},\sigma^i_j;p_{jl}\right)}{2\sigma^{i^2}_j} \times \frac{\left(\prod_{q=1,\, q\neq j}^{t} \bar{u}_{\tilde{A}^i_q}\right)\left(\omega_{1i}-y_1\right)}{\left(\sum_{i=1}^{L}\bar{f}^i + \sum_{i=L+1}^{M}\underline{f}^i\right)} \right) \tag{6}$$

Equations (7) and (8) are used to tune consequent parameters by keeping antecedent parameters fixed.

if $i \le L$,

$$\omega_{1i}(l+1) = \omega_{1i}(l) - \alpha\left(\frac{\left(y(p_l)-d_l\right)}{2}\right)\left(\frac{\prod_{q=1}^{t}\bar{u}_{\tilde{A}^i_q}}{\sum_{i=1}^{L}\bar{f}^i + \sum_{i=L+1}^{M}\underline{f}^i}\right) \tag{7}$$

if $i > L$,

$$\omega_{1i}(l+1) = \omega_{1i}(l) - \alpha\left(\frac{\left(y(p_l)-d_l\right)}{2}\right)\left(\frac{\prod_{q=1}^{t}\underline{u}_{\tilde{A}^i_q}}{\sum_{i=1}^{L}\bar{f}^i + \sum_{i=L+1}^{M}\underline{f}^i}\right) \tag{8}$$

where α is the learning rate parameter, M is the total number of rules in the rule base of the T2FNN, L and R obtained from the iterative Karnik–Mendel procedure, t is the number of inputs to rule i of the T2FNN, $\underline{u}_{\tilde{A}^i_q}(p_j)$ & $\bar{u}_{\tilde{A}^i_q}(p_j)$ are the lower & upper membership function values of p_j to the interval type-2 fuzzy set $\tilde{A}^i_q$, and $*$ is the product t-norm.

Here,

$$N\left(m^i_j,\sigma^i_j;p_j\right) = \exp\left(-\frac{1}{2}\left(\frac{p_j - m^i_j}{\sigma^i_j}\right)^2\right) \tag{9}$$

$$\underline{f}^i = \underline{u}_{A_{i1}}(p_1) * \underline{u}_{A_{i2}}(p_2) * \cdots * \underline{u}_{A_{in}}(p_n) = \prod_{q=1}^{t}\underline{u}_{A^i_q}(p_q) \tag{10}$$

and

$$\bar{f}^i = \bar{u}_{A_{i1}}(p_1) * \bar{u}_{A_{i2}}(p_2) * \cdots * \bar{u}_{A_{in}}(p_n) = \prod_{q=1}^{t}\bar{u}_{A^i_q}(p_q) \tag{11}$$

After the system is trained, it can be used for detecting intrusions in real-time. When the cloud user requires access to the service, his/her pattern of activity is checked against fuzzy rulebase. Inference from this comparison is used in making decisions regarding intrusions, after type reduction and defuzzification.

5. Experimental Results

The 1999 version of MIT Lincoln Laboratory – DARPA (Defense Advanced Research Projects Agency) intrusion detection evaluation data is usually used in researches concerning intrusion detection. This dataset describes each event (connection) with 41 features and 24 attack types. There are four main categories of attacks given in the data set: denial-of-service(DoS), probe, remote-to-local (R2L) and user-to-root (U2R).

The original data set is of size 744 MB with 4,940,000 records. Using all 41 variables could result in a big IDS model with increased computation time, which could be an overhead for online detection. Moreover, if the dataset contains unrelated features, analysis will be difficult to detect suspicious behaviors. IDS must therefore reduce the amount of data to be processed. The technique of omitting one feature at a time is employed in [29]. Each diminished feature set was then experimented on Support Vector Machines and Neural Networks to grade the importance of input features and found that by using only 19 of the most influential features, change in accuracy of intrusion detection was trivial. The simplest way to do this is by doing an intelligent input feature selection which improves classification by searching for the subset of features using data mining techniques including Bayesian networks and Classification and Regression Trees (CART). So the number of variables is reduced to 12 [1, 4]. Conditional mutual information based feature selection is suggested in [18]. Features in the datasets are of different forms such as symbolic, discrete, and continuous. Most pattern classification methods are not capable to process data in such a format. So each of the mapped features is linearly scaled to the range [0, 1] by preprocessing.

Input feature selection is vital to design efficient IDS for real world detection systems. Current dataset cannot be used due to the assortment of user requirements, the distinct operating systems installed in the Virtual Machines, and the data size of cloud systems. So, a Log Analyzer and Correlator System (LACS) [13] has been applied to the logs from the DARPA Intrusion Detection Evaluation Group of MIT Lincoln Laboratory to generate Cloud Intrusion Detection Dataset (CIDD) that consists of both knowledge and behavior based audit data collected from both UNIX and Windows users. The data set for our experiments encloses 11,982 records made at random, having 12 features. This dataset has the same distribution of attacks as "10% KDD" dataset. The training set includes 8,988 samples and the remaining 2,994 samples build the test set.

Pattern	Accuracy % (ρ = 0.0000007)			Accuracy % (ρ = 0.0000128)			Accuracy % (ρ = 0.0002187)		
	IT2FNN	T2FNN	T1FNN	IT2FNN	T2FNN	T1FNN	IT2FNN	T2FNN	T1FNN
Normal	99.86	99.99	99.71	99.73	99.89	99.12	99.96	100	99.53
Probe	99.90	99.99	99.86	99.81	99.96	99.06	99.93	99.96	99.75
DoS	99.95	99.96	99.96	99.65	99.75	98.97	99.76	99.82	99.65
U2R	87.82	90.12	80.72	86.40	90.50	85.63	88.10	89.90	85.01
R2L	99.81	99.77	99.64	99.12	99.52	98.76	99.92	99.94	98.96

Table 1. Performance Comparison between T1FNN, T2FNN and IT2FNN in ID.

The results of experiments are explained by comparing the detection rate values of type-1, type-2 and interval type-2 fuzzy neural systems on CIDD dataset. In our system, for the measurement of the input similarity, we use a predefined threshold ρ, where we may have two cases. In a first case, when ρ increases, the number of clusters also get increased, resulting in large number of smaller clusters. So, the patterns in a cluster are needed to be more analogous to each other. Besides that, there is a possibility for overlapping of clusters, where a single pattern may belong to more than one cluster. It makes the test hard. So, for a single input pattern, several fuzzy rules may be fired. On the other case, when ρ decreases, the number of clusters decrease, resulting in small number of larger clusters. So, a typical cluster may include patterns of considerable dissimilarity. Moreover, less number of larger clusters will lead to a situation of having significant gap between clusters. Hence, when an input pattern is presented to the detection system, there may not be any cluster corresponding to that pattern. So an abnormal activity may go undetected. Both these cases are not optimal. So we conducted experiments by having constant τ and σ_0 (τ = 0.5 and σ_0 = 0.2) for three different ρ values (0.0000007, 0.0000128, and 0.0002187). Results are explained by comparing the detection rate values of type-1, type-2 and interval type-2 fuzzy neural systems on CIDD dataset and illustrated in table 1. As can be seen from the table, T2FNN achieves significantly better accuracy on all classes.

Figure 2 shows the results in terms of mean square error (MSE) measured during these experiments for the above mentioned values of parameters. In all cases, as seen from figure 2, T2FNN attains the best MSE for the testing data.

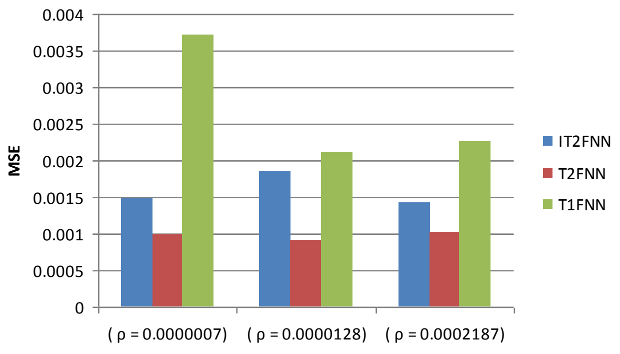

Figure 2. Results with different ρ values.

Next, we conducted experiments based on output similarity test by fixing ρ = 0.0000001 and σ_0 = 0.2 for three different τ values (0.4, 0.5, and 0.6). The setting of τ will also have an effect on the number of clusters obtained. Smaller τ results in large number of smaller clusters. Conversely, when τ increases, the number of clusters decreases, resulting in small number of larger clusters. Hence the circumstances discussed previously in setting ρ value take place here also. The results of these experiments in terms of MSE are given in figure 3. But in all cases, it appears that, T2FNN gives the best MSE.

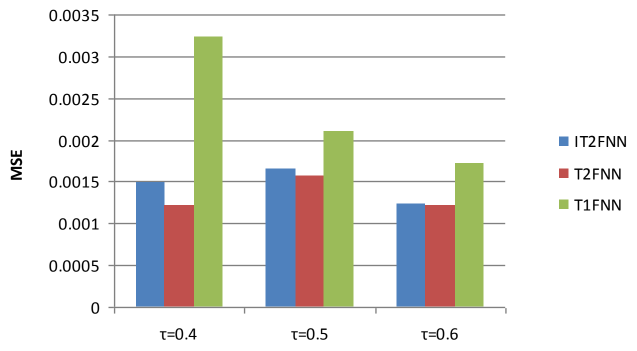

Figure 3. Results with different τ values.

\

Finally, figure 4 shows the results of experiments with ρ = 0.0000001 and τ = 0.5 for three different σ_0 values (0.15, 0.2, and 0.25). Here σ_0 is the initial value for the deviation to create a new cluster, if no cluster exists to represent the incoming pattern. When the pattern has not passed both similarity tests, a new cluster is created to represent that pattern. This cluster has only one pattern and hence its deviation is zero, which could not be in fuzzy similarity calculation. So σ_0 is used to initialize its deviation. As new patterns are added into this cluster, its deviation along with other parameters is updated according to equations from (1) to (4). Similar to previous discussions, σ_0 may affect the number of clusters. As σ_0 decreases, the patterns in a cluster are required to be more similar to each other and thus the number of clusters obtained increases. Again, T2FNN achieves the best MSE for the testing data.

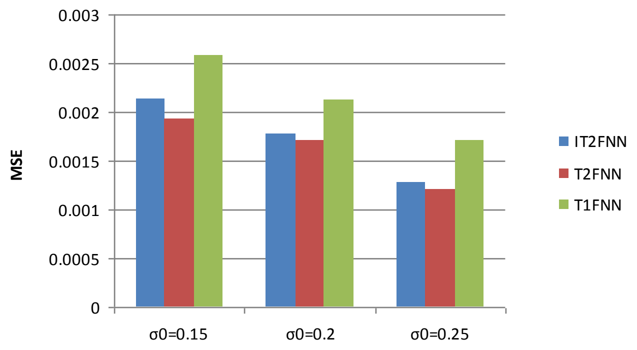

Figure 4. Results with different σ_0 values

6. Conclusion

In this paper, we have compared three fuzzy neural systems based on type-1, type-2 and interval type-2 fuzzy sets for modeling intrusion detection system in a cloud. The intrusion detection training dataset is partitioned into several clusters with similar patterns belonging to same cluster. Each of the resulting clusters is defined with the membership functions by statistical means and deviations. One fuzzy TSK-rule is derived from each cluster. A fuzzy neural network is constructed accordingly and the associated parameters are refined with dynamical optimal learning. For a new input from the test data set, a corresponding crisp output of the system is obtained by combining the inferred results of all the rules. This result is compared with the desired result based on which MSE values are calculated. In all cases, T2FNN achieves best MSE than IT2FNN and T1FNN.

Acknowledgment

The authors would like to express thanks to the anonymous reviewers for their comments, who were very supportive in improving the quality and presentation of this paper.

References

[1] Abraham A., and Jain R., “Soft Computing Models for Network Intrusion Detection Systems,” *Classification and Clustering for Knowledge Discovery, Studies in Computational Intelligence*, vol.4, pp.191-207, 2005.

[2] Alsharafat W., “Applying Artificial Neural Network and eXtended Classifier System for Network Intrusion Detection,” *The Interantional Arab Journal of Information Technology*, vol. 10, no. 3, pp. 230-238, 2013.

[3] Castro J.R., Castillo O., Melin P., and Rodríguez-Díaz A., “A hybrid learning algorithm for a class of interval type-2 fuzzy neural networks,” *Inf. Sci*., vol. 179, no. 13, pp. 2175–2193, 2009.

[4] Chebrolu S., Abraham A., and Thomas J.P., “Feature deduction and ensemble design of intrusion detection systems,” *Computers and Security*, Elsevier, 2004.

[5] Dhage S.N., Meshram B.B., Rawat R., Padawe S., Paingaokar M., and Misra A., “Intrusion detection system in cloud computing environment,” *International Conference & Workshop on Emerging Trends in Technology*, NewYork, USA, 235–239, 2011.

[6] Hagras H., “Comments on dynamical optimal training for interval type- 2 fuzzy neural network (T2FNN),” *IEEE Trans. Syst., Man, Cybern., Part B: Cybern*., vol. 36, no. 5, pp. 1206–1209, 2006.

[7] Jiang J.-Y., Liou R.-J., and Lee S.-J., “A Fuzzy Self-Constructing Feature Clustering Algorithm for Text Classification,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 3 pp. 335-349, 2011.

[8] Jin H., Xiang G., Zou D., Wu S., Zhao F., Li M.,

Zheng W., "A VMM-based intrusion prevention system in cloud computing environment," *The Journal of Super- computing*, 1–19, 2011.

[9] Juang C.-F., and Tsao Y.-W., "A self-evolving interval type-2 fuzzy neural network with online structure and parameter learning," *IEEE Trans. Fuzzy Syst*., vol. 16, no. 6, pp. 1411–1424, 2008.

[10] Karnik N.N., and Mendel J.M., "Centroid of a type-2 fuzzy set," *Inform. Sci*., vol. 132, pp. 195–220, 2001.

[11] Karnik N.N., Mendel J.M., and Liang Q., "Type-2 fuzzy logic systems" *IEEE Trans. Fuzzy Syst*., vol. 7, pp. 643–658, 1999.

[12] Kholidy H.A., and Baiardi F., "CIDS: a Framework for Intrusion Detection in Cloud Systems," *Ninth International Conference on Information Technology: New Generations* (ITNG), LasVegas, NV, 379–385, 2012.

[13] Kholidy H.A., and Baiardi F., "CIDD: A Cloud Intrusion Detection Dataset for Cloud Computing and Masquerade Attacks," Ninth *International Conference on Information Technology - New Generations*, Las Vegas, Nevada USA, 2012.

[14] Lee C.-H., Hu T.-W., Lee C.-T., and Lee Y.-C., "A recurrent interval type-2 fuzzy neural network with asymmetric membership functions for nonlinear system identification," *Proc. IEEE Int. Conf. Fuzzy Syst*., Hong Kong, pp. 1496–1502, 2008.

[15] Liang Q., and Mendel J.M., "Interval type-2 fuzzy logic systems: Theory and design," *IEEE Trans. Fuzzy Syst*., vol. 8, no. 5, pp. 535–550, 2000.

[16] Liang Q., and Mendel J.M., "Equalization of nonlinear time-varying channels using type-2 fuzzy adaptive filters," *IEEE Trans. Fuzzy Syst*., vol. 8, no. 5, pp. 551–563, 2000.

[17] Lin Y., Cunningham G.A. III, and Coggeshall, S.V., "Using fuzzy partitions to create fuzzy systems from input-output data and set the initial weights in a fuzzy neural network," *IEEE Trans. Fuzzy Syst*., vol. 5, no. 4, pp. 614–621, 1997.

[18] Liu H., Mo Y., and Zhao J., "Conditional Dynamic Mutual Information-Based Feature Selection," *Computing and Informatics*, vol. 31, no. 6, pp. 1193—1216, 2012.

[19] Mendel J.M., "UNCERTAIN Rule-Based Fuzzy Logic Systems: Introduction and New Directions," *Englewood Cliffs*, NJ: Prentice-Hall, 2001.

[20] Mendel J.M., and John R.I.B., "Type-2 Fuzzy Sets Made Simple," *IEEE Transactions On Fuzzy Systems*, vol. 10, no. 2, pp. 117-127, 2002.

[21] Mizumoto M., "Fuzzy sets and type 2 under algebraic product and algebraic sum," *Fuzzy Sets Syst*., vol. 5, pp. 277–290, 1981.

[22] Mizumoto M., and Tanaka K., "Some properties of fuzzy sets of type-2," *Inform. Control*, vol. 31, pp. 312–340, 1976.

[23] Nascimento G., and Correia M., "Anomaly-based intrusion detection in software as a service," Dependable *Systems and Networks Workshops*, 19–24, 2011.

[24] Nieminen J., "On the algebraic structure of fuzzy sets of type-2," *Kybernetica*, vol. 13, no. 4, pp. 261-273, 1977.

[25] Patel A., Qassim Q., Shukor Z., Nogueira J., Junior J., and Wills C., "Autonomic Agent-Based Self-Managed Intrusion Detection and Prevention System," *South African Information Security Multi-Conference*(SAISMC2010), Port Elizabeth, South Africa, 223–234, 2009.

[26] Sepúlveda R., Castillo O., Melin P., and Montiel O., "An efficient computational method to implement type-2 fuzzy logic in control applications," *Anal. Design Intell. Syst. Soft Comput. Tech*., vol. 41, pp. 45–52, 2007.

[27] Smith D., Guan Q., and Fu S., "An Anomaly Detection Framework for Autonomic Management of Compute Cloud Systems," *34th Annual Computer Software and Applications Conference Workshops*(COMPSACW), Seoul, 376–381, 2010.

[28] Subashini S., and Kavitha V., "A survey on security issues in service delivery models of cloud computing," *Journal of Network and Computer Applications*, Elsevier, 34(1), 1–11, 2011.

[29] Sung A.H., and Mukkamala S., "Feature Selection for Intrusion Detection using Neural Networks and Support Vector Machines," *TRB Annual Meeting* CD-ROM, 2003.

[30] Thawonmas R., and Abe S., "Function approximation based on fuzzy rules extracted from partitioned numerical data," *IEEE Trans. Syst., Man, Cybern., Part B: Cybern*., vol. 29, no. 4, pp. 525–534, 1999.

[31] Tupakula U., Varadharajan V., and Akku N., "Intrusion Detection Techniques for Infrastructure as a Service Cloud," *IEEE International Conference on Dependable, Autonomic and Secure Computing*, 744–751, 2011.

[32] Vanathi R., and Gunasekaran S., "Comparison

of network intrusion detection systems in cloud computing environment," *International conference on computer communication and informatics* (ICCCI), Coimbatore, 1–6, 2012.

[33] Vieira K., Schulter A., Westphall C.B., and Westphall C.M., "Intrusion detection for grid and cloud computing," *IT Professional*, 12, 38–43, 2010.

[34] Wang C.-H., Cheng C.-S., and Lee T.-T., "Dynamical optimal training for interval type-2 fuzzy neural network (T2FNN)," *IEEE Trans. Syst., Man, Cybern., Part B: Cybern*., vol. 34, no. 3, pp. 1462–1477, 2004.

[35] Wang X., Ting-lei H., and Xiao-yu L., "Research on the Intrusion detection mechanism based on cloud computing," *International Conference on Intelligent Computing and Integrated Systems (ICISS)*, Guilin, 125–128, 2010.

[36] Wong C.-C., and Chen C.-C., "A hybrid clustering and gradient descent approach for fuzzy modeling," *IEEE Trans. Syst., Man, Cybern., Part B: Cybern*., vol. 29, no. 6, pp. 686–693, 1999.

[37] Yeh C.-Y., Jeng W.-H.R., and Lee S.-J., "Data-Based System Modeling Using a Type-2 Fuzzy Neural Network with a Hybrid Learning Algorithm," *IEEE Transactions on Neural Networks*, vol. 22, no. 12, pp. 2296-2309, 2011.

[38] Zadeh L.A., "The concept of a linguistic variable and its application to approximate reasoning–I," *Inf. Sci*., vol. 8, no. 3, pp. 199–249, 1975.

Sivakami Raja received B.E. degree from the Dept. of Computer Science and Engineering, Madurai Kamaraj University, India, in 2002 and the M.E. degree from the Dept. of Information and Communication Engineering, Anna University, Chennai, in 2006. Currently, she is an Associate Professor in the department of Information Technology, PSNA College of Engineering and Technology, Dindigul, Tamilnadu, India. Her research interest includes data mining, information security and cloud computing.

Saravanan Ramaiah received B.E. degree in Electrical and Electronics Engineering from Thiagarajar College of Engineering, India, in 1994, M.E. in Computer Science and Engineering from Madurai Kamaraj University, India, in 2000 and the Ph.D. in Distributed Computing from Anna University, in 2010. Currently, he is a Director in RVS Educational Trust's Group of Institutions, Dindigul, Tamilnadu, India. His research interest includes distributed computing, information security and mobile computing.