Available online www.ejaet.com

European Journal of Advances in Engineering and Technology, 2016, 3(2): 6-14

Research Article ISSN: 2394 - 658X

A Hybrid Approach of Intrusion Detection using ANN and FCM

Swain Sunita[1], Badajena J Chandrakanta[2] and Rout Chinmayee[3]

[1] *Research Scholar, College of Engineering and Technology, Bhubaneswar, Orissa, India*
[2]*Department of Information Technology, College of Engineering and Technology, Bhubaneswar, Orissa, India*
[3]*Department of Computer Science and Engineering, Ajay Binay Institute of Technology, Cuttack, Orissa, India*
j.chandrakantbadajena@gmail.com

ABSTRACT

The rapid development and expansion of World Wide Web and local network systems have changed the computing world in the last decade. However, this outstanding achievement has a drawback. The highly connected computing world has also equipped the intruders and hackers with new facilities for their destructive purposes. The costs of temporary or permanent damages caused by unauthorized access of the intruders to computer systems have urged different organizations to increasingly implement various systems to monitor data flow in their networks. These systems are generally referred to as Intrusion Detection Systems (IDSs). There are two main approaches to the design of IDSs. In a misuse detection based IDS, intrusions are detected by looking for activities that correspond to known signatures of intrusions or vulnerabilities. On the other hand, anomaly detection based IDS detect intrusions by searching for abnormal network traffic.

In the present study, an off-line intrusion detection system is implemented using Multi-Layer Perceptron (MLP) artificial neural network. While in many previous studies the implemented system is a neural network with the capability of detecting normal or attack connections, in the present study a more general problem is considered in which the attack type is also detected. Fuzzy C-mean clustering is used to classify the input into different classes of clusters.

Key words: Detection Systems (IDS), MLP, Artificial Neural Networks (ANN), KDD 99 dataset, FCM

INTRODUCTION

The rapid development and expansion of World Wide Web and local network systems have changed the computing world in the last decade. However, this outstanding achievement has a drawback. The highly connected computing world has also equipped the intruders and hackers with new facilities for their destructive purposes. The costs of temporary or permanent damages caused by unauthorized access of the intruders to computer systems have urged different organizations to increasingly implement various systems to monitor data flow in their networks. These systems are generally referred to as Intrusion Detection Systems (IDSs). There are two main approaches to the design of IDSs. In a misuse detection based IDS, intrusions are detected by looking for activities that correspond to known signatures of intrusions or vulnerabilities. On the other hand, anomaly detection based IDS detect intrusions by searching for abnormal network traffic. The abnormal traffic pattern can be defined either as the violation of accepted thresholds for frequency of events in a connection or as a user's violation of the legitimate profile developed for his/her normal behavior. One of the most commonly used approaches in expert system based intrusion detection systems is rule-based analysis. Rule-based analysis relies on sets of predefined rules that are provided by an administrator or created by the system. Unfortunately, expert systems require frequent updates to remain current. This design approach usually results in an inflexible detection system that is unable to detect an attack if the sequence of events is even slightly different from the predefined profile. The problem may lie in the fact that the intruder is an intelligent and flexible agent while the rule based IDSs obey fixed rules. This problem can be tackled by the application of soft computing techniques in IDSs [1].

Soft computing is a general term for describing a set of optimization and processing techniques that are tolerant of imprecision and uncertainty. The principal constituents of soft computing techniques are Fuzzy Logic (FL), Artificial Neural Networks (ANNs), Probabilistic Reasoning (PR), and Genetic Algorithms (GAs). The idea behind the application of soft computing techniques and particularly ANNs in implementing IDSs is to include an intelligent agent in the system that is capable of disclosing the latent patterns in abnormal and normal connection audit records, and to generalize the patterns to new (and slightly different) connection records of the same class.

The architecture of the ANN model is designed so as to reduce the error. Instead of classifying the inputs into two classes i.e normal and attack, the MLP is used to classify the inputs into five classes (DoS, Probe, U2R,R2L, Normal) defining the type of attack. Further using Fuzzy C-Mean Clustering algorithm the inputs are used to form five different clusters [2].

INTRUSION DETECTION SYSTEM

An **Intrusion detection system (IDS)** is a security system that monitors computer systems and network traffic and analyzes that traffic for possible hostile attacks originating from outside the organization and also for system misuse or attacks originating from inside the organization. An intrusion detection system (IDS) inspects all inbound and outbound network activity and identifies suspicious patterns that may indicate a network or system attack from someone attempting to break into or compromise a system.

Structure of IDS

The overall architecture of IDS can be illustrated as: It has been placed centrally to capture all the incoming packets that are transmitted over the network. Data are collected and send for pre-processing to remove the noise; irrelevant and missing attributes are replaced. Then the preprocessed data are analyzed and classified according to their severity measures. If the record is normal, then it does not require any more change or else it send for report generation to raise alarms. Based on the state of the data, alarms are raised to make the administrator to handle the situation in advance. The attack is modeled so as to enable the classification of network data. All the above process continues as soon as the transmission starts [3].

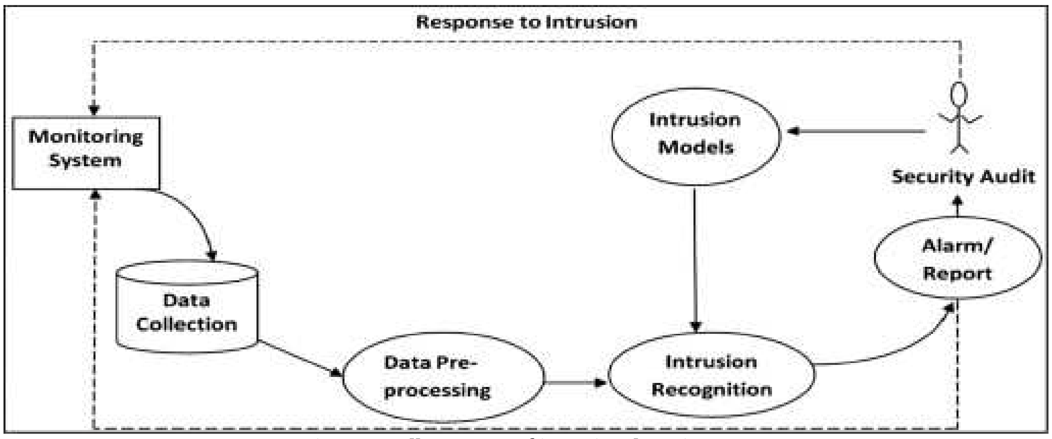


Fig. 1Overall structure of intrusion detection system

REVIEW OF LITERATURE

Currently, many IDSs are rule-based systems where the performances highly rely on the rules identified by security experts. Since the amount of network traffic is huge, the process of encoding rules is expensive and slow. Moreover, security people have to modify the rules or deploy new rules manually using a specific rule-driven language. To overcome the limitations of rule-based systems, a number of IDSs employ data mining techniques. Data mining is the analysis of large data sets to discover understandable patterns or models. Data mining can efficiently extract patterns of intrusions for misuse detection, identify profiles of normal network activities for anomaly detection, and build classifiers to detect attacks. Data-mining-based systems are more flexible and deployable. The security experts only need to label audit data to indicate intrusions instead of hand coding rules for intrusions. The author proposes different data mining techniques for intrusion detection. The techniques include Association Rule, Classification, Clustering Techniques, Decision Tree, Genetic Algorithm, Support Vector Machine, Neural Network, K-Nearest Neighbor and Fuzzy Logic [4]. The author proposes new systematic frameworks that apply a data mining algorithm called random forests in misuse, anomaly, and hybrid detection. The random forests algorithm is an ensemble classification and regression approach, which is one of the most effective data mining techniques. The random forests algorithm has been used extensively in different applications. For instance, it has been applied to prediction and probability estimation. However, the algorithm has not been applied in automatic intrusion detection. In our proposed system, the misuse component uses the random forests algorithm for the classification in intrusion detection, while the anomaly component is based on the outlier detection mechanism of the algorithm [5].

In the expert system-based IDS, a set of rules are used to describe intrusions. Audit events are translated into facts that carry their semantic significance in the expert system. Then, an inference engine can draw conclusions using these rules and facts. State transition analysis expresses attacks with a set of goals and transitions based on state transition diagrams. Any event that triggers an attack state is detected as an intrusion. Signature analysis describes attacks using signatures that can be found in audit trail. Any activity that matches the signatures is identified as an attack [6].

The author investigates three algorithms for unsupervised anomaly detection: cluster-based estimation, k-nearest neighbour, and one-class support vector machine (SVM). Supervised anomaly detection uses attack-free training data to build profiles of normal activities [7]. After that, it uses the deviation from the profiles to detect intrusions. Statistical methods and expert systems are also applied in supervised anomaly detection. Statistical methods build the profiles of user and system normal behaviour by a number of samples. Expert systems describe normal behaviour of users and systems by a set of rules and then apply the rules to detect anomalous behaviour [8].

EVALUATION OF DATASET

KDD 99 intrusion detection datasets, which are based on DARPA 98 dataset, provides labelled data for researchers working in the field of intrusion detection and is the only labelled dataset publicly available [9]. Numerous researchers employed the datasets in KDD 99 intrusion detection competition to study the utilization of machine learning for intrusion detection and reported detection rates up to 91% with false positive rates less than 1%.During the last decade, anomaly detection has attracted the attention of many researchers to overcome the weakness of signature-based IDSs in detecting novel attacks, and KDDCUP'99 is the mostly widely used data set for the evaluation of these systems. Having conducted a statistical analysis on this data set, we found two important issues which highly affect the performance of evaluated systems, and results in a very poor evaluation of anomaly detection approaches [10].

Attack Types

The simulated attacks fall in one of the following four categories [9]:

Denial of Service Attack (DoS): DoS is an attack in which the attacker makes some computing or memory resource too busy or too full to handle legitimate requests, or denies legitimate users access to a machine.

User to Root Attack (U2R): U2R is a class of exploit in which the attacker starts out with access to a normal user account on the system (perhaps gained by sniffing passwords, a dictionary attack, or social engineering) and is able to exploit some vulnerability to gain root access to the system.

Remote to Local Attack (R2L): R2L occurs when an attacker who has the ability to send packets to a machine over a network but who does not have an account on that machine exploits some vulnerability to gain local access as a user of that machine.

Probing Attack: Probing Attack is an attempt to gather information about a network of computers for the apparent purpose of circumventing its security controls.

Features

Single connection vectors each of which contains 41 features and is labeled as either normal or an attack, with exactly one specific attack type. KDD'99 features can be classified into three groups:

Basic Features: This category encapsulates all the attributes that can be extracted from a TCP/IP connection. Most of these features leading to an implicit delay in detection.

Traffic Features: This category includes features that are computed with respect to a window interval and is divided into two groups:

- **"same host" features:** examine only the connections in the past 2 seconds that have the same destination host as the current connection, and calculate statistics related to protocol behavior, service, etc.
- **"same service" features:** examine only the connections in the past 2 seconds that have the same service as the current connection.

The two aforementioned types of "traffic" features are called time-based. However, there are several slow probing attacks that scan the hosts (or ports) using a much larger time interval than 2 seconds, for example, one in every minute. As a result, these attacks do not produce intrusion patterns with a time window of 2 seconds. To solve this problem, the "same host" and "same service" features are re-calculated but based on the connection window of 100 connections rather than a time window of 2 seconds. These features are called connection-based traffic features.

Content Features: Unlike most of the DoS and Probing attacks, the R2L and U2R attacks don't have any intrusion frequent sequential patterns. This is because the DoS and Probing attacks involve many connections to some host(s) in a very short period of time; however the R2L and U2R attacks are embedded [11]in the data portions of the packets, and normally involves only a single connection. To detect these kinds of attacks, we need some features to be able to look for suspicious behavior in the data portion, e.g., number of failed login attempts. These features are called content features.

PROPOSED ARCHITECTURE

The following framework gives the overall description about the proposed approach. In this framework, KDD 99 dataset is used as training data for classification purpose. KDD 99 is an intrusion detection dataset. Artificial neural network is used to classify the attacks. Back propagation algorithm is applied to reduce the mean square error rate and train the system. Fuzzy C-mean clustering is applied to form the clusters.

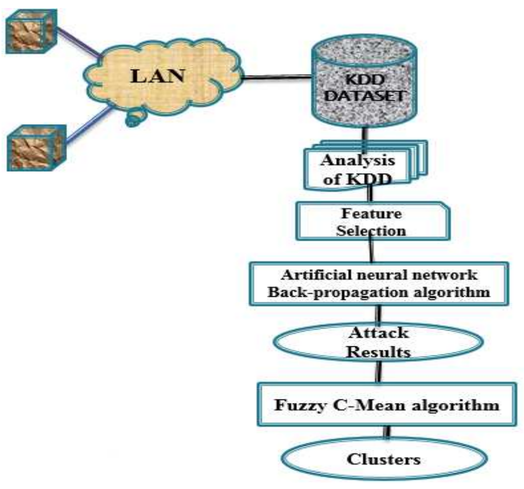


Fig.2 Architecture of the proposed system

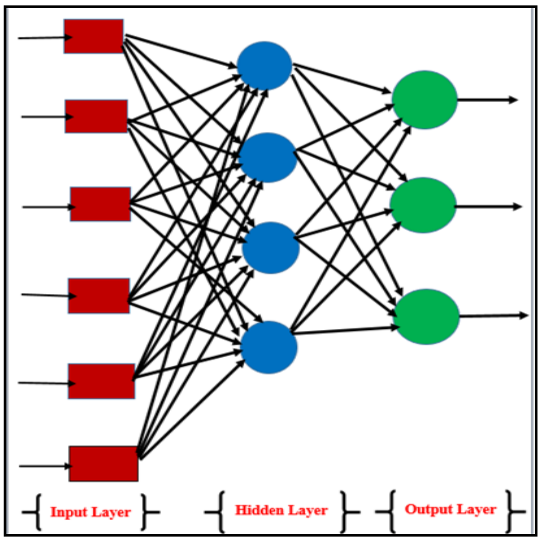


Fig.3 ANN Model

PROPOSED ANN MODEL

We are using the 41 features as input and to represent the 41 features we are using 6 bits. Each bit is represented as one neuron. Since there are 6 neurons in the input layer, one hidden layer with 4 hidden neurons are used. The attacks are classified into five classes and to represent the five classes 3 bits of output is used.

Implementation

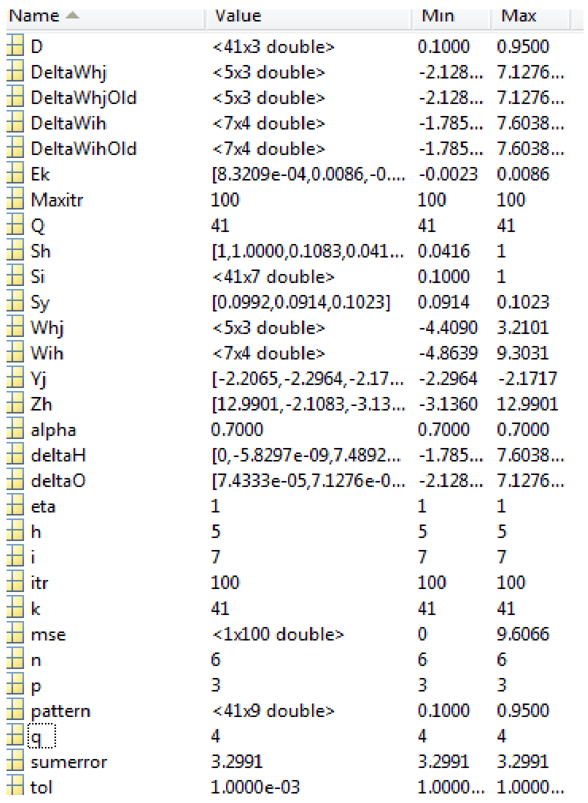
The present study was aimed to solve a multi class problem. Here, a five class case is described which can be extended to cases with more attack types. An output layer with three neurons (output states) was used. The desired output vectors used in training, validation, and testing phases were simply as mentioned above. In practice, sometimes the output of the neural network showed other patterns like which were considered irrelevant. It is straightforward to show that there are 5 possible irrelevant cases. A two layer neural network means a neural network with one hidden layers (the input layer is not counted because it acts just like a buffer and no processing takes place in it; however, the output layer is counted). The universal approximation theorem states that an MLP (with one or more hidden layers) can approximate any function with arbitrary precision and of course the price is an increase in the number of neurons in the hidden layer. The question is if anything is gained by using more than one hidden layer. One answer is that using more than one layer may lead to more efficient approximation or to achieving the same accuracy with fewer neurons in the neural network. As we increase the number of hidden layers the complexity of the system increases. The architecture of the Neural network was set as [6:4:3] which indicates 6 neurons in the input layer, 4 neurons in the hidden layer and 3 neurons in the output layer.

EXPERIMENTAL RESULTS

The implemented intrusion detector is a two layer multi-layer perceptron (one hidden layer with four neurons).The learning rate and momentum and the architecture of the model was set. The numbers of iterations are increased gradually to reduce the mean square error (MSE).The below snapshot (Fig. 4 and Fig. 5) shows how the MSE is gradually reducing with the increasing iterations.

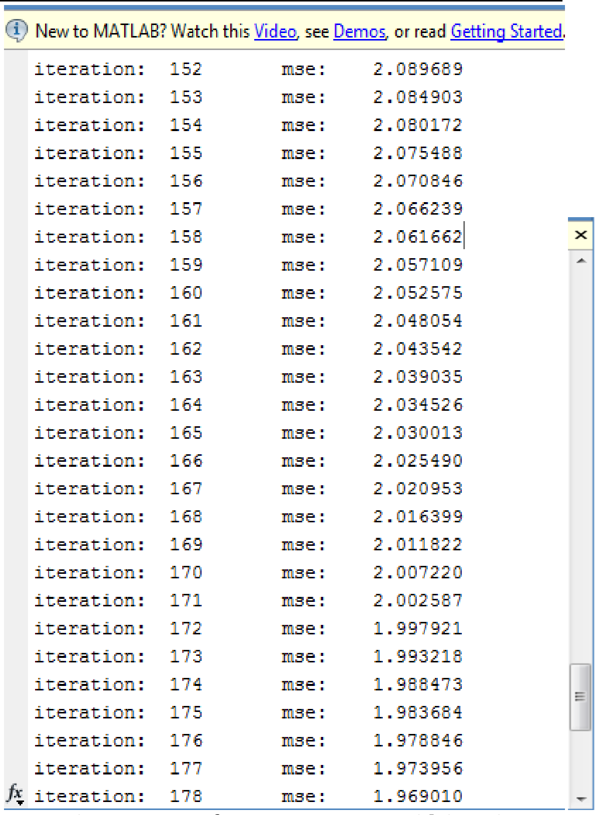
Fig. 6, Fig. 7, and Fig. 8 shows the MSE vs. number of iterations plot at 1000, 4000 and 8000 iterations respectively. The mean square error values at different iterations are given below:

Iteration:	999	mse:	0.005533
Iteration:	3999	mse:	0.000992
Iteration:	7999	mse:	0.000461



Name	Value	Min	Max
D	<41x3 double>	0.1000	0.9500
DeltaWhj	<5x3 double>	-2.128...	7.1276...
DeltaWhjOld	<5x3 double>	-2.128...	7.1276...
DeltaWih	<7x4 double>	-1.785...	7.6038...
DeltaWihOld	<7x4 double>	-1.785...	7.6038...
Ek	[8.3209e-04,0.0086,-0....	-0.0023	0.0086
Maxitr	100	100	100
Q	41	41	41
Sh	[1,1.0000,0.1083,0.041...	0.0416	1
Si	<41x7 double>	0.1000	1
Sy	[0.0992,0.0914,0.1023]	0.0914	0.1023
Whj	<5x3 double>	-4.4090	3.2101
Wih	<7x4 double>	-4.8639	9.3031
Yj	[-2.2065,-2.2964,-2.17...	-2.2964	-2.1717
Zh	[12.9901,-2.1083,-3.13...	-3.1360	12.9901
alpha	0.7000	0.7000	0.7000
deltaH	[0,-5.8297e-09,7.4892...	-1.785...	7.6038...
deltaO	[7.4333e-05,7.1276e-0...	-2.128...	7.1276...
eta	1	1	1
h	5	5	5
i	7	7	7
itr	100	100	100
k	41	41	41
mse	<1x100 double>	0	9.6066
n	6	6	6
p	3	3	3
pattern	<41x9 double>	0.1000	0.9500
q	4	4	4
sumerror	3.2991	3.2991	3.2991
tol	1.0000e-03	1.0000...	1.0000...

Fig.4 Parameters used in the NN model



New to MATLAB? Watch this Video, see Demos, or read Getting Started.

```
iteration:  152     mse:    2.089689
iteration:  153     mse:    2.084903
iteration:  154     mse:    2.080172
iteration:  155     mse:    2.075488
iteration:  156     mse:    2.070846
iteration:  157     mse:    2.066239
iteration:  158     mse:    2.061662
iteration:  159     mse:    2.057109
iteration:  160     mse:    2.052575
iteration:  161     mse:    2.048054
iteration:  162     mse:    2.043542
iteration:  163     mse:    2.039035
iteration:  164     mse:    2.034526
iteration:  165     mse:    2.030013
iteration:  166     mse:    2.025490
iteration:  167     mse:    2.020953
iteration:  168     mse:    2.016399
iteration:  169     mse:    2.011822
iteration:  170     mse:    2.007220
iteration:  171     mse:    2.002587
iteration:  172     mse:    1.997921
iteration:  173     mse:    1.993218
iteration:  174     mse:    1.988473
iteration:  175     mse:    1.983684
iteration:  176     mse:    1.978846
iteration:  177     mse:    1.973956
iteration:  178     mse:    1.969010
```

Fig.5 Decrease of mean square error with iterations

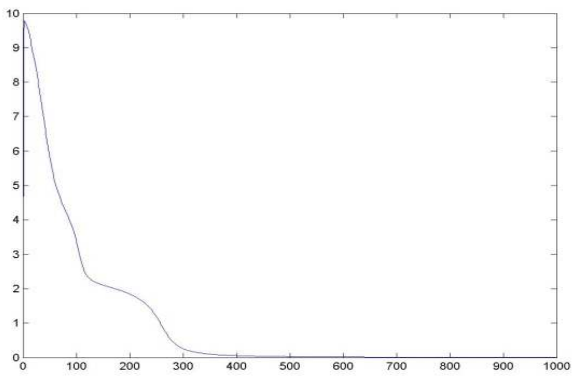


Fig.6 Plot for reduction of mean square error with an iteration count of 8000

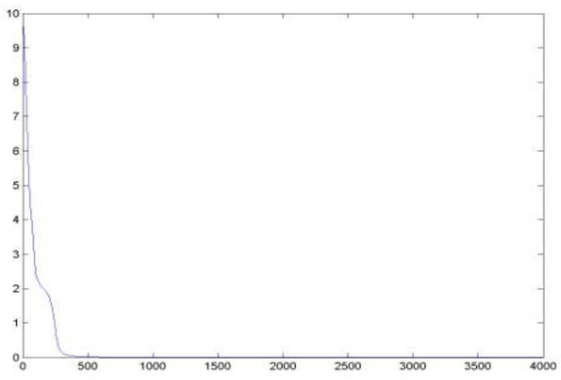


Fig. 7 Plot for reduction of mean square error with an iteration count of 4000

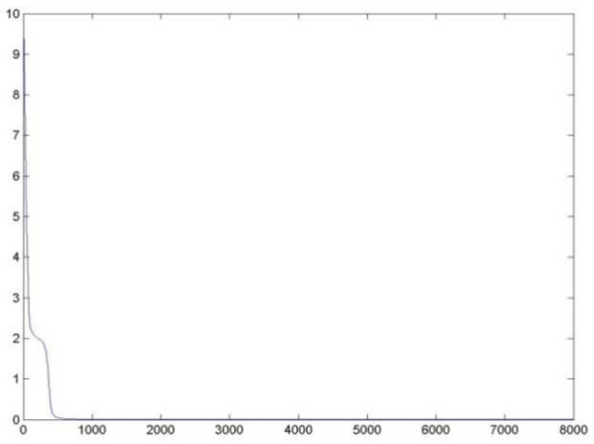


Fig.8 Plot for reduction of mean square error with an iteration count of 8000

FCM ALGORITHM

Bezdek introduced Fuzzy C-Means clustering method in 1981, extend from Hard C-Mean clustering method. FCM is an unsupervised clustering algorithm that is applied to wide range of problems connected with feature analysis, clustering and classifier design. FCM is widely applied in agricultural engineering, astronomy, chemistry, geology, image analysis, medical diagnosis, and shape analysis and target recognition. With the development of the fuzzy theory, the FCM clustering algorithm which is actually based on Ruspini Fuzzy clustering theory was proposed in 1980's. This algorithm is used for analysis based on distance between various input data points. The clusters are formed according to the distance between data points and the cluster centers are formed for each cluster.

In fact, FCM is a data clustering technique in which a data set is grouped into n clusters with every data point in the dataset related to every cluster and it will have a high degree of belonging (connection) to that cluster and another data point that lies far away from the center of a cluster which will have a low degree of belonging to that cluster.

Initialization:
Select the following parameters:
- The required number of clusters *N*, ($2 < N < k$).
- Measure distances as Euclidean distance; a fixed parameter q (usually 1.5).
- Initial (at zero iteration) matrix $U^{(0)}=(C_i)^{(0)}$ object ownership x_i with the given initial cluster centers c_j.

1. **Calculate the centers vectors $C(k)=[cj]$with $U(k)$**

In the t-th iteration step in the known matrix is computed in accordance with the above solution of differential equations.

$$C_j = \frac{\sum_{i=1}^{N} u_{ij}^{m} * x_i}{\sum_{i=1}^{N} u_{ij}^{m}}$$

2. **Modified membership measure u_{ij}**

$$\boldsymbol{u_{ij}} = \frac{\mathbf{1}}{\sum_{k=1}^{c} \left(\frac{\|x_i - c_j\|}{\|x_i - c_k\|} \right)^{\frac{2}{m-1}}}$$

3. **If $\| U^{(k+1)} - U^{(k)} \| < \epsilon$, then STOP; otherwise return to step 2.**

The advantages include:
- Gives best result for overlapped data set and comparatively better than k-means algorithm.
- Unlike k-means where data point must exclusively belong to one cluster center here data point is assigned membership to each cluster center as a result of which data point may belong to more than one cluster center.

The disadvantages include:
- Apriori[6] specification of the number of clusters.
- With lower value of β we get the better result but at the expense of more number of iteration.
- Euclidean distance measures can unequally weight underlying factors.

IMPLEMENTATION

The matlab function fcm performs FCM clustering. The function FCM takes a data set and a desired number of clusters and returns optimal cluster centers and membership grades for each data point. It starts with an initial guess for the cluster centers, which are intended to mark the mean location of each cluster. The initial guess for these cluster centers is most likely incorrect. Next, FCM assigns every data point a membership grade for each cluster. By iteratively updating the cluster centers and the membership grades for each data point, FCM iteratively moves the cluster centers to the right location within a data set. This iteration is based on minimizing an objective function that represents the distance from any given data point to a cluster center weighted by that data point's membership grade.
Before using the FCM algorithm, the following parameters must be specified:
- the number of clusters, c,
- the fuzziness exponent, m,
- the termination tolerance, ε.

FCM clustering is an iterative process. The process stops when the maximum number of iterations is reached, or when the objective function improvement between two consecutive iterations is less than the minimum amount of improvement specified.

The following function can be used as follows:

[center, U, obj_fcm] = fcm (data, cluster_n)

The arguments of this function are:

- data - lots of data to be clustering, each line describes a point in a multidimensional feature space;
- cluster_n - number of clusters (more than one).

The function returns the following parameters:

- center - the matrix of cluster centers, where each row contains the coordinates of the center of an individual cluster;
- U - resulting matrix;
- obj_fcn - the objective function value at each iteration.

So, for example we should write load ccc.dat

```
  [center, U, obj_fcn] = fcm(ccc, 2);
 maxU = max(U);
index1 = find(U(1, :) == maxU);
 index2 = find(U(2, :) == maxU);
```

EXPERIMENTAL RESULTS

We assign random values to the 41 inputs using the random function specifying the dimension as two. So the function can be written as: data = rand(41,2);

The following table shows the reduction of values of objective function with the increase in number of iterations.

Table-1 Values of Objective Function with respect to Iteration Number

```
Iteration count = 1, obj. fcn = 1.822512
Iteration count = 2, obj. fcn = 1.363840
Iteration count = 3, obj. fcn = 1.180942
Iteration count = 4, obj. fcn = 0.950938
Iteration count = 5, obj. fcn = 0.779923
Iteration count = 6, obj. fcn = 0.708741
Iteration count = 7, obj. fcn = 0.689441
Iteration count = 8, obj. fcn = 0.680589
Iteration count = 9, obj. fcn = 0.674269
Iteration count = 10, obj. fcn =0.669250
Iteration count = 11, obj. fcn =0.664890
Iteration count = 12, obj. fcn =0.659757
Iteration count = 13, obj. fcn =0.652149
Iteration count = 14, obj. fcn =0.642696
Iteration count = 15, obj. fcn =0.635101
Iteration count = 16, obj. fcn =0.631253
Iteration count = 17, obj. fcn =0.629721
Iteration count = 18, obj. fcn =0.629129
Iteration count = 19, obj. fcn =0.628891
Iteration count = 20, obj. fcn =0.628791
Iteration count = 21, obj. fcn =0.628747
Iteration count = 22, obj. fcn =0.628727
 Iteration count = 23, obj. fcn=0.628717
```

The plot for reduction of objective function with respect to iteration count is given below:

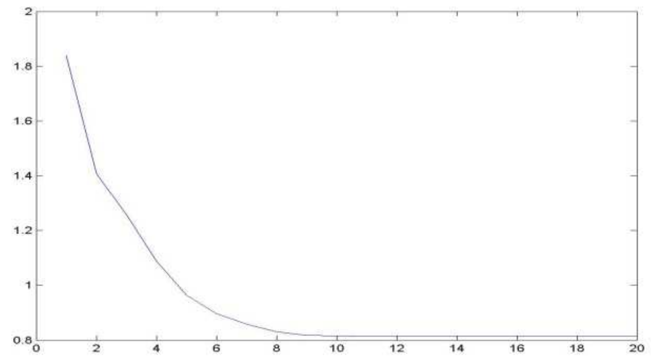


Fig.9 Plot for Reduction of Objective function with Iteration Count

The number of clusters is specified as five since we have five classes. The following table shows the index values of each cluster. The indexes of different clusters could be equated with the center values. It can be written as:

data1= [data(index1,1) data(index1,2)]

data2= [data(index2,1) data(index2,2)]

Table-2 Index Values of Each Cluster

Data1		Data2		Data3		Data4		Data5	
0.4386	0.2278	0.8843	0.2467	0.8335	0.4981	0.5144	0.5860	0.1673	0.5747
0.1548	0.0835	0.6787	0.1209	0.7689	0.9009	0.5880	0.6664	0.1999	0.6260
0.1363	0.0170	0.5606	0.0320	0.8620	0.8452	0.4070	0.6609	0.3185	0.7690
0.1476	0.2094	0.6967	0.3624	0.9899	0.7386	0.5341	0.5814	0.0900	0.9283
0.0005	0.2055	0.5828	0.0495	0.7487	0.7298	0.4952	0.8627	0.1117	0.5801
0.4795	0.2819	0.8790	0.1925	0.8256	0.8908	0.4950	0.8449	0.1897	0.4843
		0.9889	0.1231	0.7900	0.9823	0.5277	0.6352	0.0550	0.5523
		0.8654	0.1465	0.8507	0.6299				
		0.6126	0.1891	0.9296	0.6147				
		0.9900	0.0427	0.8154	0.4896				
				0.8013	0.5386				

The center values of each cluster are obtained as:

0.1593	0.1338
0.8204	0.1671
0.8324	0.7652
0.5293	0.6428
0.1624	0.5924

The graph of clusters was plotted taking the respective two dimensional center values. For example it can be written as:

Plot(center(1,1),center(1,2),'ko','markersize',15,'linewidth',2)

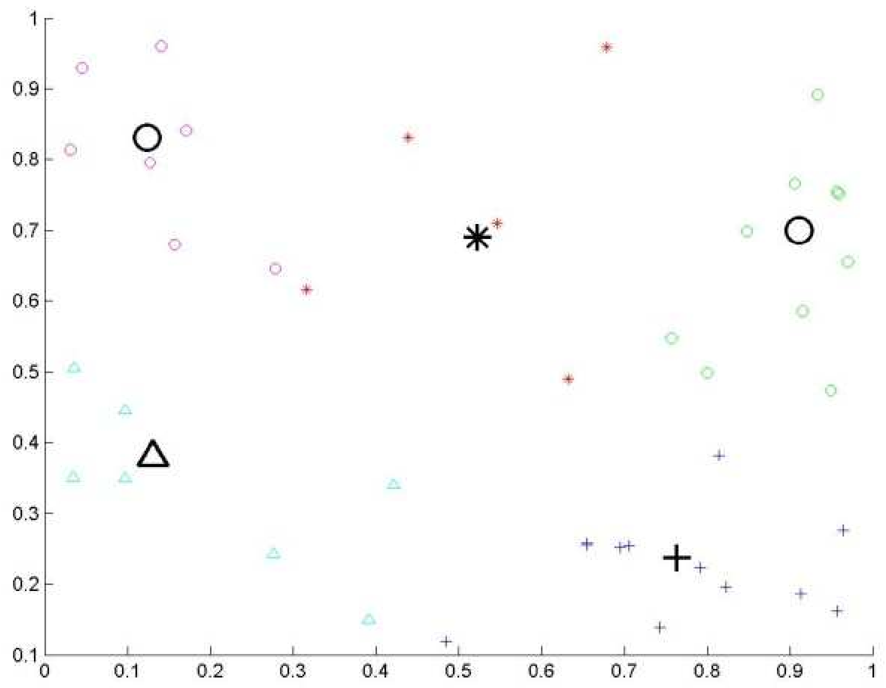


Fig.10 Plot showing the different clusters

CONCLUSION

An approach for a neural network based intrusion detection system, intended to classify the normal and attack patterns and the type of the attack is proposed. When the neural network parameters were determined by training, classification of a single record was done in a negligible time. Therefore, the neural network based IDS can operate as an online classifier for the attack types that it has been trained for. The proposed system presents a new approach of intrusion detection system based on neural network. Artificial neural networks are inspired by the learning processes that take place in biological systems. Artificial neurons and neural networks try to imitate the working mechanisms of their biological counterparts. Learning can be perceived as an optimization process. Neural networks can be considered as nonlinear function approximating tools (i.e., linear combinations of nonlinear basis functions), where the parameters of the networks should be found by applying optimization methods. A Multi Layer Perceptron (MLP) is used for intrusion detection system. The results show that the implemented and designed system detects the attacks and classify them in five groups. KDD Data set is used for the training and evaluation of the ANN classifier. The input pattern has been formed into five clusters representing the five classes. The center values and index values has been calculated for each cluster by using fuzzy c-mean clustering.

The most commonly reported application of neural networks in IDSs is to train the neural network on a sequence of information units, each of which may be an audit record or a sequence of commands. The ability of neural networks to learn and generalize in addition to their wide range of applicability makes them very powerful tools. From the practical point of view, the experimental results imply that there is more to do in the field of artificial neural network based intrusion detection systems especially solving irrelevant outputs. The implemented system solved classification problem. Its further development to several classes is straightforward. As a possible future development to the present study, one can include more attack scenarios in the dataset. Practical IDSs should include several attack types. In order to avoid unreasonable complexity in the neural network, an initial classification of the connection records to normal and general categories of attacks can be the first step. The records in each category of intrusions can then be further classified to the attack types. In future, we can use the center values and index values obtained to represent the data pattern and train the system in less iteration.

REFERENCES

[1] J Zhang, M Zulkernine, A Haque, Random Forests- Based Network Intrusion Detection Systems, *IEEE Transactions on Systems, Man and Cybernetics*, **2008**, 38 (5), 649-659.
[2] J Zhang, M Zulkernine, A Hybrid Network Intrusion Detection Technique using Random Forests, *Proceedings of the IEEE Computer Society First International Conference on Availability, Reliability and Security*, **2006.**
[3] D Anderson, T Frivold and A Valdes, *Next-Generation Intrusion Detection Expert System - A Summary*, SRI Int., Menlo Park, CA,Tech. Rep. SRI-CSL-95-07, **1995**.
[4] P Kalarani and S Selva Brunda, A Survey on Efficient Data Mining Techniques for Network Intrusion Detection System, *International Journal of Advanced Research in Computer and Communication Engineering*, **2014**, 3 (9), 8028-8031.
[5] L Breiman, *Random Forests and Machine Learning*, **2001**, 45(1), 5–32.
[6] D Barbara and S Jajodia, *Applications of Data Mining in Computer Security*, Norwell, MA: Kluwer, **2002**.
[7] V Kosamkar, SS Chaudhari, Improved Intrusion Detection System using C4.5 Decision Tree and Support Vector Machine, *International Journal of Computer Science and Information Technologies*, **2014**, 5 (2), 1463-1467.
[8] AA Eskin, M Prerau, L Portnoy and S Stolfo, *A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data in Applications of Data Mining in Computer Security*, Norwell, MA: Kluwer, **2002**.
[9] Incorporating Hidden Markov Model into Anomaly Detection Technique for Network Intrusion Detection, *International Journal of Computer Applications*, **2012**, 53 (11).
[10] MS Hoque, M Mukit, M Bikas, A Naser, An Implementation of Intrusion Detection System using Genetic Algorithm, *International Journal of Network Security and its Applications*, **2012**, 4(2), 109-120.
[11] KVR Swamy, KSV Lakshmi, Network Intrusion Detection Using Improved Decision Tree Algorithm, *International Journal of Computer Science and Information Security*, **2012**, 3 (5), 26-32.