Intrusion Detection using Fuzzy Clustering and Artificial Neural Network

Shraddha Surana

Research Scholar, Department of Computer Engineering, Vishwakarma Institute of Technology, Pune India shraddha.surana@gmail.com

Abstract

This paper presents the outline of a hybrid Artificial Neural Network (ANN) based on fuzzy clustering and neural networks for an Intrusion Detection System (IDS). While neural networks are effective in capturing the non-linearity in data provided, it also has certain limitations including the requirement of high computational resources. By clustering the data, each ANN is trained on a particular subset of data, reducing time required and achieving a higher detection rate. The outline of the implemented algorithm is as follows: first the data is divided into smaller homogeneous groups/ subsets using a fuzzy clustering technique. Subsequently, a separate ANN is trained on each subset. Finally, Results of each ANN from step 2 is aggregated to form the final output which will decide the classification of the data point.

Keywords: Artificial Neural Network, Intrusion Detection System, Fuzzy clustering

1 Introduction

Intrusion detection is an important aspect in today's world where security is of utmost importance. A single intrusion in a network can cause information leaks or data modification which can prove to be hazardous to any company or organization. An intrusion detection system attempts to detect misuse or unauthorised access of a system or a network. An IDS does not usually perform any action to prevent intrusions; its main function is to alert the system administrators that there is a possible security violation; as such it is a proactive tool rather than a reactive tool [1].

Intrusion detection systems (IDS) can be classified as: (1) Host based or Network based (2) Online or Offline (3) Misuse based or Anomaly based. A host based IDS makes use of log files from individual computers, whereas a network based IDS captures packets from the network and analyses its contents. An online IDS is able to flag an intrusion while it's happening whereas an offline IDS analyses records after the event has occurred and raises a flag indicating that a security breach had occurred since the last intrusion detection check was performed. An Anomaly based IDS detects deviation from normal behaviour while misuse based IDS compare activities on the system with known behaviours of attackers.

This paper outlines a hybrid approach using Artificial Neural Networks and Fuzzy clustering to detect intrusions in a network. The method outlined is network based which extracts features from packets in the network. The algorithm classifies the packet as normal or the type of attack depending upon the contents of the packet. Attacks fall into four main categories: Denial of Service (DoS); User to Root (U2R); Remote to Local (R2L) and Probe. The KDD repository was used to train the algorithm.

Neural networks which are a class of machine learning algorithms used to classify data can be used where the problem is too complex to be programmed by hand. Instead, the neural network is trained to give more importance to the features that are the main characteristics of a particular class in order to help the network classify the incoming data. Neural networks have been successfully implemented [2] [3] [4] to detect intrusions.

Neural networks, however, require substantial amount of data to train on, before they can successfully classify incoming data. Due to this limitation, the neural network is not able to train well on low frequency attacks such as R2L and U2R resulting in their lower detection accuracy [5]. Different hybrid approaches have been explored in the past to overcome the drawbacks of any one individual method [6] [7] [8]. The two approaches used in this paper viz. ANN and Fuzzy clustering are used so that they complement each other. Neural Networks are good at being able to classify unseen data points whereas fuzzy clustering enables the algorithm to generalize well.

2 Research Background

Various approaches using ANN have been used for IDS. The Neural Network Intrusion Detector (NNID) system proposed by Ryan et al. [9] took into account the behaviour of individual users and created profiles for each user. The input pattern was then matched to the user profiles to identify the user (A node corresponding to a user with value > 0.5 is attributed to that user). A flag was raised if no match was found and the input was considered as an anomaly. However, this required large amount of data to train the network for each user. Insufficient data for a user might lead to false positives for that user's behaviour on the network [9]. The NNID system had an anomaly detection rate of 96% and a false alarm rate of 7%. The Multi-layered Perceptron (MLP) by Cannady [10] using backpropagation algorithm modelled general use rather than creating user specific profiles. Training of the neural network required 26.13 hours to complete with approximately 98% match in the training dataset and 97.5% match in the test dataset.

In contrast to the above two approaches that used backpropagation algorithm, Silva, et al. [11] used a Hamming Net to classify network event in real time. The Hamming Net is a fast pattern matcher that finds the most similar class, according to a pre-defined similarity threshold, providing great flexibility and fault tolerance by finding small attack variations. On an average 70% of wellknown malicious input data within the payload file encountered its common pair in the exemplar file and is classified as suspicious information.

Lei and Ghorbani [12] compared the performance of Self-Organizing Map (SOM) and Improved Competitive Learning Networks (ICLN). While the accuracy obtained for both SOM and ICLN were similar, the computation time for SOM was generally higher than that of ICLN (specifically, ICLN requires one fourth the computational time of the SOM). The clustering result is also independent of the number of initial neurons which is not the case in SOM. The results of the simulated annealing approach in Gao and Tian's [13] paper show that the mean squared errors of training samples of improved simulated annealing neural network is smaller than that of a backpropagation network. Moreover the mean squared errors of testing samples of improved simulated annealing neural network is also smaller than that of a backpropagation network. When the number of the training samples change, we can get the same result. It shows that the network intrusion detection method based on improved simulated annealing neural network has higher stability, and can obtain higher detection and recognition accuracy.

The Probabilistic Neural Network (PNN) implemented by Devaraju and Ramakrishnan [14] performs better than Feed Forward Neural Network (FFNN) and Radial Basis Neural Network (RBNN). However, PNN (accuracy = 80.38%) performs only 0.02% better than FFNN (accuracy = 80.4%) which is not a significant difference. The accuracy of RBNN is 75.4%. These results are comparatively lower as compared to other algorithms implemented in the other papers discussed above. The hybrid approach presented in Wang et al. [15] using fuzzy logic and artificiaANN neural network have obtained an average accuracy of 96.71% which can be considered as very successful. The fuzzy logic provides some flexibility to the uncertain nature of detecting intrusions [16].

The hybrid approach in this paper implements fuzzy clustering and ANN. Fuzzy clustering - a form of unsupervised technique is used to divide the training data into smaller groups/ subsets. ANNs are trained using these subsets. Since the size of data is reduced, the training time required to train each ANN is also reduced. Aggregating the results of these individual ANNs by a final aggregating ANN helps increase its detection rate as any misclassifications made by individual ANNs will be corrected by the final aggregating ANN. Thus the objective is to lower the training time required while increasing the detection rate of identifying attacks.

3 **Problem Solution**

Figure 1 shows the general outline of the method using Fuzzy Clustering and Artificial Neural Network (FCANN). The dataset used for the experiments is the 1999 KDD Cup dataset. This dataset contains about five million connection records as training data and about two million connection records as test data. It includes a set of 41 features derived from each connection and a label which specifies the status of connection records as either normal or the specific type of attack. Random records are taken from the training set and given to the fuzzy clustering module (the first module). This module divides the training sets into smaller clusters. Each cluster forms a subset to be given to every ANN in the ANN module (second module). Each ANN trains on this subset to classify a record as one of the five groups normal, DoS, U2R, R2L and Probe. This output is then given to the third module - the final aggregating module. The aggregator ANN takes the outputs of the individual ANN and trains on them to reduce any misclassifications. This module gives the final classification of the record.



Fig 1 Block diagram of fuzzy clustering and ANN for IDS

3.1 Training

The FCANN algorithm can be divided into three submodules viz. fuzzy clustering; ANN; and fuzzy aggregation. The following sections describe these three submodules in detail.

Module 1: Fuzzy Clustering

The training data is divided into 'x' number of clusters such that there is homogeneity within the clusters and heterogeneity between clusters. Each data point belongs to a particular cluster with the degree specified by its membership grade. The training set is thus divided into several subsets decreasing the size and complexity of each subset.

The Fuzzy c-means (FCM) clustering algorithm originally introduced by Bezdek [17] is used to divide the data into several clusters. The FCM algorithm is based on the minimization of the following objective function [18] [19]:

$$J_m = \sum_{j=1}^k \sum_{i=1}^n u_{ij}^m ||x_i - c_j||^2$$
, $1 < m < \infty$

where, u denotes the degree of membership of data point x falling into cluster center c and m is the weighting exponent greater than 1. The fuzzy clustering module is composed of the following steps:

- 1. Fix c and m and randomly initialize 'x' number of cluster centers. Initialize membership matrix U to U(0) and step k=0.
- 2. At each step k, calculate the cluster centers with the membership matrix U(k)

$$c_j = \frac{\sum_{i=1}^n u_{ij} x_i}{\sum_{i=1}^n u_{ij}}$$

3. Compute an updated membership matrix U(k+1)

$$u_{ij} = \frac{1}{\sum_{p=1}^{k} \left(\frac{||x_i - c_j||}{||x_i - c_p||}\right)^{\frac{2}{m-1}}}$$

4. Compare U(k+1) and U(k). If $||U(q + 1) - U(q)|| < \varepsilon$ then stop. Otherwise set U(k) = U(k + 1) and return to step 2.

Once the termination criteria is reached, the whole training set is divided into 'x' number of subsets, each of which is given to a different ANN for learning features specific to that subset.

Module 2: Artificial Neural Network

The ANN module consists of a separate ANN for each cluster formed and aims to learn the patterns present in every subset. A simple feed-forward network is used for each ANN which consists of simple processing units called nodes and weighted connections between nodes in adjacent layers. The ANN employed in our experiments uses three layers - input, hidden and output layer. Data is given at the input layer and is traversed through the neural network and is classified into one of the five classes at the output layer. The five classes are: Normal, DoS, U2R, R2L and Probe.

To learn the weights of this multi-layered neural network we use the backpropagation algorithm with the gradient descent weight update rule. The gradient descent aims to minimize the squared error between the ANN predicted output and the actual target values. The error function used is:

$$E_m = \frac{1}{2n} \sum_k \sqrt{(T_k - Y_k)^2}$$

The back propagation algorithm used to train the ANN is outlined as follows [20]:

- 1. Create an ANN with number of input nodes corresponding to the number of features in the dataset; the number of output nodes corresponding to the number of output classes and an appropriate number of hidden layer nodes.
- 2. Initialize the weights to small random numbers.
- 3. For every training example, forward propagate the input through the network:
 - a. Each hidden node receives the weighted summation of the inputs and bias

$$hid(j) = b_j + \sum_{i=1}^n x_i w_{ij}$$

where j is the j^{th} hidden unit and i denoted the i^{th} example

b. This is then passed through a nonlinear activation function. A unipolar sigmoid activation function is used:

$$f(x) = \frac{1}{(1 + \exp(-x))}$$

c. Output of the hidden layers is then given to the output layer in a similar manner:

$$y(k) = b_k + \sum_{i=1}^n x_i w_{ik}$$

and is passed through the activation function

- d. The output computed through the ANN is then compared to the target value and the error is calculated using the error function
- 4. This error is then backpropagated through the ANN and the weights updated according to the expression:

$$w(t+1) = w(t) - \eta \partial E(t) / \partial w(t)$$

where t is the number of epochs and η is the learning rate.

5. The momentum parameter α (0< α <1) is used to accelerate the learning process

$$w(t+1) = w(t) - \frac{\eta \partial E(t)}{\partial w(t)} + \alpha \Delta w(t)$$

6. If the error E_m < threshold defined then stop training. Else return to step 3.

Thus in the ANN module every individual ANN is trained on its own subset and the resulting output is then given to the final aggregating ANN.

Module 3: Fuzzy Aggregation

Once the individual ANNs have been trained on their subsets, their results must be aggregated to reduce any errors introduced by individual ANNs in the ANN module. To achieve this, another ANN is used to learn and remove errors made by the sub ANNs. The following steps are used to accomplish this task [15]:

- 1. Forward propagate the whole training set through every sub ANN in module two. Each sub ANN will output its 'opinion' as to which class the particular record must fall into.
- 2. This output of ANN_x (one sub ANN in module two) is multiplied by its membership grade belonging to the cluster the sub ANN was trained on.

3. The input to the final aggregating ANN is

$$Y_{input} = [y_1 . U_1; y_2 . U_2; ...; y_n . U_n]$$

The output is compared with the target output. The aggregating ANN uses the same backpropagation algorithm used for the individual sub ANNs in the ANN module.

3.2 Testing

During the stage of testing, the working methodology of ANN module and fuzzy aggregation module is similar as described above. First, the membership grade is calculated based on the cluster centres C. For a new input x_i , the membership U is calculated based on C by:

$$u_{ij} = \frac{1}{\sum_{p=1}^{k} \left(\frac{||x_i - c_j||}{||x_i - c_p||}\right)^{\frac{2}{m-1}}}$$

Once the membership of the input data is determined, the data point is given to the next ANN module. The outputs of all the ANNs in module 2 are then aggregated using the final ANN to determine which class the input should be classified as.

The test data used has cases not seen by the network during the training phase. Thus, the testing will not only test the system on previously seen data but also on unseen data. This makes the intrusion detection task more realistic.

The time taken for training is to be noted down for various combinations of parameters (learning rate and momentum). The test set will be presented to the system with those parameters and the detection rate noted. The ideal combination is to have minimum training time and maximum detection rate for the test set.

4 Experiments and Results

4.1 Experiment

The software used to implement the FCANN algorithm was MATLAB R2008a on a Windows7 PC with i5 core 2.30 GHz CPU and 4GB RAM.

In the experiments, KDD CUP 1999 [21] dataset is used which is a version of the original 1998

DARPA intrusion detection evaluation program dataset. Random selection was used to reduce the size of the dataset. Table 1 shows detailed information of the number of records used to train the network. The 'kddcup data 10 percent' dataset was used for training. All the records belonging to U2R, R2L and Probe attacks were selected due to their low frequency in the dataset. 3,000 Normal records and 10,000 DoS records were randomly selected [15].

Table 1 Number and distribution of training and test dataset

	Training	dataset	Testing dataset		
Connection	Number	% of	Number	% of	
type	of	records	of	records	
	records		records		
Normal	3000	16.41	60,593	19.48	
DoS	10,000	54.69	229,853	73.89	
U2R	52	0.28	288	0.09	
R2L	1126	6.16	16,189	5.2	
Probe	4107	22.46	4166	1.34	

The KDD dataset contains 41 features and the class of attack (or normal). Symbolic value features such as protocol_type are converted to numeric values to be given to the ANNs. The dataset is normalized before being used so that all feature values are in consistent ranges.

The training data was divided into 6 clusters in the fuzzy clustering module [15]. Each subset was then given to its individual ANN in the ANN module (2nd module). This ANN has 41 input nodes corresponding to the 41 features in the dataset; 5 output nodes corresponding to the five classes viz. Normal; DoS; R2L; U2R and Probe. The number of hidden nodes was determined using the empirical formula $\sqrt{I+O} + \alpha$ where I is the number of input nodes, O is the number of output nodes and α is taken as 10 due to the complexity of intrusion detection [15]. Thus, the second module ANN architecture is 41-18-5. The output of the second module is given to the final aggregating module. Thus the input features to the final aggregating ANN is 5. The number of output nodes too will be 5 corresponding to the 5 output classes and the number of hidden layer nodes are 13 calculated using the formula stated above. The final ANN architecture is 5-13-5.

Training time		Learning Rate											
taken	(CPU												
seconds)		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1		
	0	2350.7	1470.0	753.0	790.7	1799.0	1263.0	358.1	352.4	242.9	249.3		
	0.1	2687.9	1238.4	839.2	624.6	413.2	1416.7	363.6	403.5	290.9	250.4		
	0.2	1824.5	2103.9	635.4	502.2	449.5	347.0	288.3	252.2	196.9	256.2		
omentum	0.3	2641.2	880.6	607.4	700.0	420.0	340.5	250.6	211.5	247.6	186.3		
	0.4	1427.8	823.0	541.5	392.9	431.3	259.6	239.3	179.1	185.8	152.0		
	0.5	1443.9	878.5	453.0	326.8	302.3	220.3	195.7	177.5	137.1	131.8		
Š	0.6	1158.6	1098.9	383.7	329.6	183.3	175.9	154.4	109.2	116.1	130.0		
	0.7	1053.9	461.2	330.8	235.0	144.3	139.8	124.1	101.0	95.5	75.4		
	0.8	526.3	295.0	157.9	121.5	118.0	74.8	67.1	63.8	59.4	50.2		
	0.9	255.0	128.1	140.2	71.8	56.2	46.9	43.8	60.6	84.4	49.9		

Table 2 Training time taken for various values of momentum and learning rate

Table 3 Test set detection rate for various momentum and learning rate values

Test	: set	Learning Rate										
dete	ction											
rate(%)		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	
	0	64.01	64.44	75.35	72.73	80.95	82.52	82.19	81.78	81.82	79.55	
	0.1	64.32	64.82	65.38	75.29	77.34	35.70	78.22	81.76	83.25	81.87	
	0.2	74.95	64.55	75.89	68.97	72.10	78.61	78.88	78.89	82.03	78.52	
entum	0.3	64.55	64.64	64.97	73.62	72.80	77.74	83.14	79.06	81.93	78.71	
	0.4	22.14	64.55	76.73	70.58	79.64	77.65	81.38	81.85	78.90	79.30	
ome	0.5	74.96	74.90	72.74	69.34	78.36	78.45	78.89	81.78	83.25	80.80	
Mc	0.6	74.77	64.57	65.21	73.00	74.71	72.82	81.78	81.96	79.14	81.61	
	0.7	64.44	65.97	67.80	72.37	77.58	73.10	78.70	81.71	78.29	81.69	
	0.8	69.77	74.91	75.03	72.59	74.48	73.14	77.04	77.78	79.56	77.29	
	0.9	67.89	70.44	67.56	68.71	73.69	39.54	39.19	73.40	75.13	72.79	

4.2 Results

Table 2 shows the training time taken in CPU seconds for various values of momentum and learning rate and Table 3 shows the detection rate over the test set on the trained neural network. The same random dataset generated was used for all the experiments.

The general trend seen in the training time taken from Figure 2 is that as learning rate increases (from 0.1 to 1) the training time decreases. A higher learning rate leads to faster convergence to the global minima. Higher momentum also leads to lower training times as the convergence takes place in the right direction. Figure 3 shows the test set detection rate for error threshold of 0.01. It can be seen from the figure that higher detection rate is obtained for higher learning rate and lower momentum values.





To understand how the clusters are formed and how they affect the performance of the ANN, the records belonging to each class in each cluster were identified.



Fig 3 3D plot for Test set detection rate with error threshold of 0.01

The distribution of records in the 6 clusters formed is as shown in Table 4. The detection rate of the training and test set, the cluster it belongs to and the classes identified by each ANN are shown in Table 5.

			Cluster Number								
		1	2	3	4	5	6	Total			
	Normal	0	1	5	167	25	2,802	3,000			
s	DoS	0	2,173	7,172	580	4	71	10,000			
las	U2R	0	0	0	0	1	51	52			
0	R2L	0	8	0	50	15	153	1,126			
	Probe	1,585	141	0	937	1,251	193	4,107			

Table 4 Distribution of records in each cluster produced by the FCM algorithm

Table 5 Analysis of ANN output in module 2 and module 3

ANN	Training detection rate (%)	Cluster	Test set detection rate (%)	Classes identified
1	99.05	2	74.88	DoS; Probe
2	99.15	5	19.53	Normal; DoS; R2L; Probe
3	100.00	1	1.34	Probe
4	99.02	6	21.00	Normal; DoS; U2R; R2L; Probe
5	99.93	3	73.90	DoS
6	99.08	4	28.77	Normal; DoS; R2L; Probe
Final aggregating ANN	99.07	-	81.60	Normal; DoS; U2R; R2L; Probe

The FCM algorithm clusters data such that the records within each cluster are similar as can be verified from Table 4. Cluster 1 contains only Probe records and cluster 3 contains mainly DoS records. Cluster 6 is the only cluster that has significant amount of records from all classes. The values in Table 5 were taken for learning rate = 0.8, momentum = 0.6 and error threshold=0.01. These values were chosen as they gave a good combination of low training time and high detection rate (refer Table 2 and Table 3). Cluster 1 which had all Probe records, gave a 100% training detection rate but only 1.34% in the test set. This is because the ANN trained on cluster 1 received all Probe records and hence it classified all records as Probe achieving 100% in training set. For the test set too, it classified all records as Probe giving it a very low detection rate of 1.34% (the percentage of Probe records in test set). Exactly similar is the case for cluster 3 where the ANN classifies all incoming records as DoS. Since 73.89% of the test set is made of DoS records, its test set detection rate too is 73.90%. Though the detection rate is high, it is merely due to the large number of DoS records present in the test set and not because the ANN has learnt to classify correctly. This basically renders ANN 3 and ANN 5 (belonging to cluster 1 and 3 respectively) redundant as it classifies all records as Probe and DoS respectively. If the data within each cluster is too homogeneous (belonging to the same class), the ANN will just flag any data as belonging to that particular class. Due to this, the membership grades play an important role as it effectively decides the input values to the 5 nodes of the final aggregating ANN. For this, the membership grade of the records belonging to the different clusters must have significant difference to overcome the default output of each ANN. However, upon inspection it was found that the membership grades of the records do not differ significantly from each other. This makes getting the records classified correctly at the module 2 level more important. Due to this, majority of the load in classifying the record correctly falls on the final aggregating ANN.

5 Conclusion and Future work

In this paper, the FCANN algorithm was implemented to detect network intrusions. The algorithm was implemented with several combinations of learning rate and momentum to find the best learning rate and momentum combination which gives a lower training time and higher detection rate. The results of each module of the algorithm are analysed to understand its working. Results have shown that homogeneity within each cluster is not preferable and is not an ideal way to divide the training data. Each ANN must receive records belonging to different classes to get a better training at classifying records. An ideal way to divide the data to achieve good results (low training and high detection rate) remains an open problem for future research.

References

- [1] W. W. Fu and L. Cai, "A Neural Network based Intrusion Detection Data Fusion Model," in *Third International Joint Conference on Computational Science and Optimization*, 2010.
- [2] C. Zhang, J. Jiang and M. Kamel, "Intrusion Detection using hierarchical neural networks," *Pattern Recognition Letters*, pp. 779-791, 2005.
- [3] X. Tong, Z. Wang and H. Yu, "A research using hybrid RBF/ Elman neural networks for intrusion detection system secure model," *Computer Physics Communication*, pp. 1795-1801, 2009.
- [4] S.-C. O. K. Y. Wonil Kim, "Intrusion Detection Based on Feature Transform Using Neural Network," in *Computational Science -ICCS* 2004, vol. 3037, Springer Berlin Heidelberg, 2004, pp. 212-219.
- [5] R. Beghdad, "Critical study of neural networks in detecting intrusions," *Computers & Security*, pp. 168-175, 2008.
- [6] G. Liu, Z. Yi and S. Yang, "A hierarchical intrusion detection model based on the PCA neural networks," *Neurocomputing*, pp. 1561-1568, 2007.
- [7] L. Ren, "Research of Web Data Mining based on Fuzzy Logic and Neural Networks," in Sixth International Conference on Fuzzy Systems and Knowledge Discovery, 2006.
- [8] F. M.-P. F. J. M.-G. R. L.-F. J. A. G.-M.-A. D. M.-J. Iren Lorenzo-Fonseca, "Intrusion detection method using Neural Networks based on the reduction of characteristics," in *Bio-Inspired Systems: Computational and Ambient Intelligence*, vol. 5517, Springer Berlin Heidelberg, 2009, pp. 1296-1303.
- [9] J. Ryan, M.-J. Lin and R. Miikkulainen., "Intrusion Detection with Neural Networks.,"

in AAAI Technical Report WS-97-07., 1997.

- [10] J. Cannady, "Artificial Neural Networks for Misuse Detection," in *National Information* Systems Security Conference (NISSC'98), Arlington, VA., 1998.
- [11] L. d. S. Silva, A. C. F. d. Santos, J. D. S. d. Silva and A. Montes, "A Neural Network Application for Attack Detection in Computer Networks.," in *ISBN 0-7803-8359-1.*, 2004.
- [12] J. Zhong Lei and A. Ghorbani, "Network Intrusion Detection using an Improved Competitive Learning Neural Network," in Proceedings of the second annual conference on communication networks and services research (CNSR'04) 0-7695-2096-0,, 2004.
- [13] M. Gao and J. Tian, "Network Intrusion Detection Method Based on Improved Simulated Annealing Neural Network," in International Conference on Measuring Technology and Mechatronics Automation, 2009.
- [14] S. Devaraju and D. S. Ramakrishnan, "Performance analysis of intrusion detection system using various neural network classifiers," in *IEEE-International Conference on Recent Trends in Information Technology, ICRTIT*, MIT, Anna University, Chennai, 2011.
- [15] G. Wang, J. Hao, L. Huang and J. Ma, "A new approach to intrusion detection using Artificial Neural Networks and Fuzzy clustering," *Expert systems with application*, vol. 37, pp. 6225-6232, 2010.
- [16] N. B. Idris and B. Shanmugam, "Artificial Intelligence techniques applied to Intrusion Detection," in *IEEE Indicon* 2005 *Conference*, Chennai, India, 2005.
- [17] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, 1981.
- [18] J. C. Bezdek, R. Ehrlich and W. Full, "FCM: The Fuzzy c-means clustering algorithm," *Computers & Geosciences*, vol. 10, no. 2-3, pp. 191-203, 1984.
- [19] C. L. Stephen, "Fuzzy model identification based on Cluster Estimation," *Journal of Intelligent and Fuzzy systems*, vol. 2, pp. 267-278, 1994.
- [20] T. M. Mitchell, Machine Learning, McGraw-Hill, 1997.
- [21] "KDD Cup 1999," [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kd dcup99.html. [Accessed August 2012].