Neural Networks Learning Improvement using the K-Means Clustering Algorithm to Detect Network Intrusions

K. M. Faraoun and A. Boukelif

Abstract—In the present work, we propose a new technique to enhance the learning capabilities and reduce the computation intensity of a competitive learning multi-layered neural network using the K-means clustering algorithm. The proposed model use multi-layered network architecture with a back propagation learning mechanism. The K-means algorithm is first applied to the training dataset to reduce the amount of samples to be presented to the neural network, by automatically selecting an optimal set of samples. The obtained results demonstrate that the proposed technique performs exceptionally in terms of both accuracy and computation time when applied to the KDD99 dataset compared to a standard learning schema that use the full dataset.

Keywords—Neural networks, Intrusion detection, learning enhancement, K-means clustering

I. INTRODUCTION

INTRUSION detection is a critical process in network security. Traditional methods of network intrusion detection are based on the saved patterns of known attacks. They detect intrusion by comparing the network connection features to the attack patterns that are provided by human experts. The main drawback of the traditional methods is that they cannot detect unknown intrusions. Even if a new pattern of the attacks were discovered, this new pattern would have to be manually updated into the system. On the other hand, as the speed and complexity of networks develop rapidly, especially when these networks are open to the public Web, the number and types of the intrusions increase dramatically. Hence, with the changing technology and the exponential growth of Internet traffic, it is becoming difficult for any existing intrusion detection system to offer a reliable service. From earlier research, we have found that there exists a behavioural pattern in the attacks that can be learned. That is why an artificial neural network is so successful in detecting network intrusions; it is also capable of identifying new attacks to some degree of resemblance to the learned ones. The neural networks are widely considered as an efficient approach to adaptively classify patterns, but their high computation intensity and the long training cycles greatly hinder their applications, especially for the intrusion detection problem, where the amount of treated data is very important.

Neural networks have been identified since the beginning as a very promising technique of addressing the intrusion detection problem. Many researches have been performed to this end, and the results varied from inconclusive to extremely promising. The primary premise of neural networks that initially made it attractive was its generalization property, which makes it suitable to detect day-0 attacks. In addition neural networks also posses the ability to classify patterns, and this property can be used in other aspects of intrusion detection systems such as attack classification, and alert validation. In this work, an attempt is made to improve the learning capabilities of a multi-layered neural network and reduce the amount of time and resource required by the learning process by sampling the input dataset to be learnt using the K-means algorithm. This paper is organized as follow: section 1 gives some theoretic background about the use of neural networks for intrusion detection and the k-means clustering technique, then describe the proposed technique of samples reduction. The section 2 presents the architecture of the used neural networks with the different used parameters. Section 3 summarizes the obtained results with comparison and discussions. The paper is finally concluded with the most essential points and possible future works.

II. THEORY

A. Neural network models for IDS

A neural network contains no domain knowledge in the beginning, but it can be trained to make decisions by mapping exemplar pairs of input data into exemplar output vectors, and adjusting its weights so that it maps each input exemplar vector into the corresponding output exemplar vector approximately [1]. A knowledge base pertaining to the internal representations is automatically constructed from the data presented to train the network.
Well-trained neural networks represent a knowledge base in which knowledge is distributed in the form of weighted interconnections where a learning algorithm is used to modify the knowledge base from a set of given representative cases. Neural networks might be better suited for unstructured problems pertaining to complex relationships among variables rather than problem domains requiring value-based human reasoning through complex issues. Any functional form relating the independent variables (i.e., input variables) to the dependent variables (i.e., output variables) need not be imposed in the neural network model. Neural networks are thought to better capture the complex pattern of relationships among variables than statistical models because of their capability to capture non-linear relationships in data. The rules with logical conditions need not be built by developers as neural networks investigate the empirical distribution among the variables and determine the weight values of a trained network. A neural network is an appropriate method when it is difficult to define the rules clearly as is the case in the misuse detection or anomaly detection.

In order to measure the performance of an intrusion detection system, two types of rates are identified, false positive rate and true positive rate (detection rate) according to the threshold value of the neural network. The system reaches its best performance for high value of detection rate and low value of false positive rate. A good detection system must establish a compromise between the two situations.

A generic form of a neural network intrusion detector is presented in the Figure. This system uses the input labelled data (normal and attack samples) to train a neural network model. The resulting model is then applied to the new samples of the testing data to determine the corresponding class of each one, and so to detect the existing attacks. Using the label information of the testing data, the system can compute the detection performance measures given by the false alarms rate, and the detection rate. A classification rate can also be computed if the system is designed to perform attacks multi-classification.

\[ J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^j - c_j \right\|^2 \]

Where \( \left\| x_i^j - c_j \right\|^2 \) is a chosen distance measure between a data point and the cluster centre \( c_j \), an indicator of the distance of the \( n \) data points from their respective cluster centres. The general algorithm is composed of the following steps:

- Place \( K \) points into the space represented by the objects that are being clustered. These points represent initial group centroids.
- Assign each object to the group that has the closest centroid.
• When all objects have been assigned, recalculate the positions of the K centroids.
• Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.

Although it can be proved that the procedure will always terminate, the k-means algorithm does not necessarily find the most optimal configuration, corresponding to the global objective function minimum. The algorithm is also significantly sensitive to the initial randomly selected cluster centres. The k-means algorithm can be run multiple times to reduce this effect. K-means is a simple algorithm that has been adapted to many problem domains. The proposed procedure is a simple version of the k-means clustering. Unfortunately there is no general theoretical solution to find the optimal number of clusters for any given data set. A simple approach is to compare the results of multiple runs with different k classes and choose the best one according to a given criterion, but we need to be careful because increasing k results in smaller error function values by definition, but also an increasing risk of over-fitting.

C. The proposed method

In the present work, the role of the k-means algorithm is to reduce the computation intensity of the neural network, by reducing the input set of samples to be learned. This can be achieved by clustering the input dataset using the k-means algorithm, and then take only discriminant samples from the resulting clustering schema to perform the learning process. By doing so, we are trying to select a set of samples that cover at maximum the region of each class in the N-dimensional space (N is the size of the training vectors). The input classes are clustered separately in such a way to produce a new dataset composed with the centroid of each cluster, and a set of boundary samples selected according to their distance from the centroid. Reducing the number of used samples will enhance significantly the learning performances, and reduce the training time and space requirement, without great loss of the information handled by the resulting set, due to its specific distribution. The Figure 2 illustrates an example of the application of this selection schema to a 2-dimensional dataset.

The number of fixed clusters (the k parameter) can be varied to specify the coverage repartition of the samples. The number of selected samples for each class is also a parameter of the selection algorithm. Then, for each class, we specify the number of samples to be selected according to the class size. When the clustering is achieved, samples are taken from the different obtained clusters according to their relative intra-class variance and their density (the percentage of samples belonging to the cluster). The two measurements are combined to compute a coverage factor for each cluster. The number of samples taken from a given cluster is proportional to the computed coverage factor. Let A be a given class, to which we want to apply the proposed approach to extract S sample. Let k be the number of cluster fixed to be used during the k-means clustering phase. For each generated cluster cl_i (i:1..k), the relative variance is computed using the following expression:

\[ Vr(cl_i) = \frac{1}{\text{Card}(cl_i)} \sum_{x \in cl_i} \text{dist}(x, c_i) \]

(2)

When \( \text{Card}(X) \) give the cardinality of a given set X, and \( \text{dist}(x,y) \) give the distance between the two points x and y. Generally, the distance between two points is taken as a common metric to assess the similarity among the components of a samples set. The most commonly used distance measure is the Euclidean metric which defines the distance between two points \( x=(p_1,...,p_N) \) and \( y=(q_1,...,q_N) \) from \( \mathbb{R}^N \) as:

\[ \text{dist}(x, y) = \sqrt{\sum_{i=1}^{N} (p_i - q_i)^2} \]

(3)

The density value corresponding to the same cluster cl_i is computed like the following:
The coverage factor is then computed by:

\[
\text{Den}(cl_i) = \frac{\text{Card}(cl_i)}{\text{Card}(A)} \quad (4)
\]

The coverage factor is then computed by:

\[
\text{Cov}(cl_i) = \frac{(\text{Vr}(cl_i) + \text{Den}(cl_i))}{2} \quad (5)
\]

We can clearly see that: \(0 \leq \text{Vr}(cl_i) \leq 1\) and \(0 \leq \text{Den}(cl_i) \leq 1\) for any cluster \(cl_i\). So the coverage factor \(\text{Cov}(cl_i)\) belong also to the [0,1] interval. Furthermore, it is clear that:

\[
\sum_{i=1}^{k} \text{Vr}(cl_i) = 1 \quad \text{and} \quad \sum_{i=1}^{k} \text{Den}(cl_i) = 1 \quad (6)
\]

We can so deduce easily that:

\[
\sum_{i=1}^{k} \text{Cov}(cl_i) = 1 \quad (7)
\]

Hence, the number of samples selected from each cluster is determined using the expression:

\[
\text{Num\_samples}(cl_i) = \text{Round}(S \times \text{Cov}(cl_i)) \quad (8)
\]

Using (8), the algorithm presented in the figure.3 will permit to select \(S\) sample from a class \(A\) clustered with the k-means algorithm into \(k\) cluster. The parameter \(\varepsilon\) serve to ensure that the selected samples are placed in separated regions, and are not duplicated. The choice of \(\varepsilon\)'s value depend on the size of the cluster. We have proposed the following heuristic expression to compute an approximate value of \(\varepsilon\):

\[
\varepsilon = \text{Max}\left(\text{dist}(x, c_i)\right) / 10 \quad (9)
\]

This expression is only an approximate heuristic. No theoretic background was used to determine the value of \(\varepsilon\). The performances of the expression were evaluated experimentally. Finally, the resulting set of samples is then used to train the neural network.

When dealing with the intrusion detection problem, the proposed technique is applied only to the large classes. With the KDD99 dataset used in our experiments, the technique is applied to the class: normal, Dos, Probe and R2L. The U2R class is very small according to the other classes mentioned, so the totality of its samples is used during the learning process. The figure.4 illustrate the general operation schema of the proposed approach.

III. DATASETS AND EXPERIMENTS

Because the goal of this work is to study and enhance the learning capabilities of the neural network techniques for intrusions detection, the proposed method is compared to a classic neural networks implementation that use the full set of samples sampled from the KDD99 dataset [4], and witch contain 24788 sample. The use of the full ‘10% KDD’ dataset containing 972780 samples is impracticable using the neural networks on any machine configuration. Even with the used subset, the experiments show that the learning process is very process and take hours and hours to converge. The Table.1 lists the class’s distributions of our used sets.

Let \(A\) be the input class; \(k\): the number of cluster; \(S\): the number of samples to be selected (\(S \geq k\)); \(\text{Sam}(i)\): the resulting selected set of samples for the cluster \(i\). \(\text{Out\_sam}\): the output set of samples selected from the class \(A\); \(\text{Candidates}\): a temporary array that contain the cluster points and their respective distance from the centroid. \(\text{i}, \text{min}, \text{x}\): intermediates variables; \(\varepsilon\): Neighborhood parameter.

1-Cluster the class \(A\) using the k-means algorithm into \(k\) cluster.

2-For each cluster \(cl_i\) (i:1..k) do

\{ \(\text{Sam}(i) :=\{\text{centroid}(cl_i)\};\) \\
\(j:=1;\) \\
\(\text{For each } x \text{ from } cl_i \text{ do} \) \\
\(\{ \text{Candidates } [j].\text{point} :=x;\) \\
\(\text{Candidates } [j].\text{location} :=\text{dist}(x, \text{centroid}(cl_i));\) \\
\(j:=j+1 ;\} ;\) \\
\(\text{Sort the array Candidates in descending order with respect to the values of location field;}\) \\
\(j:=1;\) \\
\(\text{While}(\text{card(Sam(i)))<\text{Num\_samples}(cl_i)) \) \\
\(\text{and} \ (j<\text{card}(cl)) \text{ do} \ \{ \text{min}:=100000;\) \\
\(\text{For each } x \text{ from } \text{Sam}(i) \text{ do} \) \\
\(\{ \text{if dist(Candidates}[j].\text{point},x)<\text{min} \text{ then } \text{min}:=\text{dist(Candidates}[j].\text{point},x);\) \\
\(\} \) \\
\(\text{if } (\text{min} > \varepsilon) \) \\
\(\text{Sam}(i):=\text{Sam}(i) \cup \{\text{Candidates}[j].\text{point}\};\) \\
\(j:=j+1 ;\} ;\) \\
\(\text{if card(Sam(i))) < \text{Num\_samples}(cl_i) \text{ then} \) \\
\(\text{repeat} \ (\text{Sam}(i):=\text{Sam}(i) \cup \text{Candidates[random].point}) \) \\
\(\text{until} \ (\text{card(Sam(i))) = \text{Num\_samples}(cl_i));\) \\
\} \}

3-For \(i=1 \text{ to } k \) do \(\text{Out\_sam}:=\text{Out\_sam} \cup \text{Sam}(i);\)

![Fig 3. The proposed samples selection algorithm](image)

![Fig 4. The general operating mechanism of the proposed method](image)
At the first time, we tried to implement an intrusion classification system, to classify each intrusion to one of the learned attack classes (Dos, Prob, U2R, R2L), but the result demonstrate that a poor classification rate is obtained in this case. This can be interpreted by the fact that the power of the neural networks approach reside in their ability to discriminate the normal comportment from the intrusive one, and the discrimination between attack classes remain a hard task and give limited performances, especially for the classes U2R and R2L. The presented result demonstrates that considering the attacks classes as a single one improve significantly the detection rate with respect to the multi-classification approach. The new proposed technique was also implemented using the same principle, and the attack classes were merged in a single intrusion class, regrouping attacks categories with a relatively equivalent distribution.

In the following, we describe the architecture of the neural networks used in the experiments with the relevant parameters. The next section detail the obtained results for each implementation, and compares the performances achieved by each detection system. Attributes in the KDD datasets had all forms: continuous, discrete, and symbolic, with significantly varying resolution and ranges. Most pattern classification methods are not able to process data in such a format. Hence, pre-processing was required before pattern classification models could be built. Pre-processing consisted of two steps: first step involved mapping symbolic-valued attributes to numeric-valued attributes and second step implemented scaling. In the present work, we have used the data codification and scaling used in [10]. All the resulting scaled fields belong to the interval [0, 1]

### A. Network architecture used with the standard method

As indicated above, the first experiments were performed using a multi-layered neural network to classify the input data samples of the training set presented in the Table 1, to one of 5 classes: Normal, Dos, Prob, U2R, R2L corresponding to the normal and intrusive possible situations. The used network is composed of 3 hidden layers containing 30, 15 and 30 neurons respectively, and has 41 input and 5 outputs. The neural network was designed to produce a value of 1.0 in the output node corresponding to the class of the current sample and the value of 0.0 for the other output nodes. When testing new samples with the network, the outputs can be any value from [0, 1] due to the approximate nature of the learning, so we consider the nearest output value to 1.0 as the activated output node.

In the case of two-category learning (normal and attack), the network has only one output neuron corresponding to the involved classes. The outputs activation is handled in the following way: during the learning phase, the output value is set to 0 for normal samples, and 1.0 for the attack samples. During the test phase, the output value is rounded to the nearest value 0 or 1.0.

We have used the feed-forward back propagation [3] as learning algorithm, the Table 2 show the set of parameters used during the learning process used for all the implementations presented in this work.

### B. Network architecture used with the proposed method

Since the proposed schema use a reduced set of samples, the network architecture can be more trivial. We use only two hidden layers with 18 and 5 neurons respectively, an input layer of 41 neurons, and the output layer contain 1 neuron for normal and intrusive classes. The same parameters of learning are used as illustrated in the Table 2.

The described experiments were implemented using the MATLAB 7 environment, on a Pentium4 2.88 GHz, with 256 Mb of memory.

### C. Clustering and selection parameters

As described above, the sampling algorithm has two parameters to be defined as inputs: the cluster number k of each class, and the number of samples to be extracted S. Different possible values were tested during the experiments to find a good compromise between the size of the resulting dataset and its coverage of the input classes’ space. The Table 3 list the final chosen parameters for each class. The class U2R was totally selected, because it present a very small portion of the initial dataset (0.02% only).

---

**TABLE 1 DISTRIBUTION OF THE NORMAL AND ATTACK RECORDS IN THE USED TRAINING AND TESTING SETS.**

<table>
<thead>
<tr>
<th></th>
<th>Training Set</th>
<th>Testing Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>11673</td>
<td>60593</td>
</tr>
<tr>
<td>DOS</td>
<td>7829</td>
<td>229853</td>
</tr>
<tr>
<td>PBR</td>
<td>4107</td>
<td>4166</td>
</tr>
<tr>
<td>R2L</td>
<td>1119</td>
<td>16347</td>
</tr>
<tr>
<td>U2R</td>
<td>52</td>
<td>70</td>
</tr>
</tbody>
</table>

**TABLE 2 SET OF PARAMETERS USED TO TRAIN THE PROPOSED NEURAL NETWORKS**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network type</td>
<td>Feed-forward backpropagation</td>
</tr>
<tr>
<td>Number of inputs</td>
<td>41</td>
</tr>
<tr>
<td>Number of outputs</td>
<td>1 or 5</td>
</tr>
<tr>
<td>Hidden layers</td>
<td>3</td>
</tr>
<tr>
<td>Hidden layers size</td>
<td>15 or 30</td>
</tr>
<tr>
<td>Input and output ranges</td>
<td>[0, 1]</td>
</tr>
<tr>
<td>Training function</td>
<td>TRAINGDX</td>
</tr>
<tr>
<td>Adaptation learning function</td>
<td>LEARNGDM (the gradient descent function)</td>
</tr>
<tr>
<td>Performances function</td>
<td>MSEREG</td>
</tr>
<tr>
<td>Transfer function</td>
<td>TANGSIG</td>
</tr>
<tr>
<td>Training epochs</td>
<td>1000</td>
</tr>
</tbody>
</table>
The selected parameters were determined heuristically. For the intrusion classes, we have chosen the number of clusters according to the number of attack types included in each class. We have tried also to choose an equivalent distribution of the total number of selected samples over the different classes to avoid that one class dominate the learning process.

IV. RESULTS AND COMPARISON

In the following, we present the different obtained results for the implemented approaches. The performances of each method are measured according to the detection rate and false positive rate calculated using the following expressions:

Detection rate:
\[ DR = \frac{\text{False negatives number}}{\text{Total Number of Attacks}} \]

False Positive Rate:
\[ FP = \frac{\text{False Positives}}{\text{Total Number of normal connections}} \]

The classification rate computed for the first approach was calculated for each class using the following formula:
\[ CR = \frac{\text{Number of samples classified correctly}}{\text{Number of samples used for training}} \times 100 \]

A. Result of the standard NN-classification method: multi-classification approach

As mentioned in the section 2.1, the proposed presented neural network architecture is trained using the dataset presented in the Table.1. When learning is achieved, the resulting neural network is benchmarked using the 'Corrected (Test)' containing 14 additional (unseen) attacks, and used by almost all the classification systems developed for the KDD99 dataset.

The Table.4 illustrate the obtained classification matrix. Figure.5 show the training error evolution during the learning epochs. Obtained performances are summarized in the Table.5. We can see clearly from the comparative table (Table.6) that the classification results are relatively poor with respect to the other mentioned approaches that gives better performances with less computation and time requirements.

B. Result of the standard NN-classification method: 2-category classification approach

When we consider all the attacks as one category, the intrusion detection problem can be handled with the network proposed above (in the section 2.1) with one output handling the normal and intrusive classes. The learning is still very slow and convergence is difficult, but the detection rate is significantly enhanced compared to the multi-category learning approach. The Table.7 summarizes the obtained performances results.
C. Result of the proposed classification method

Using the output set of samples obtained form the clustering phase, we construct a new training set composed by the normal samples, and the grouped attacks samples labelled as intrusive. The resulting set is presented to the neural network described above (section 2.2). The Table 8 summarize the obtained performances results. Table 9 give the detailed description of the detection rate for each attack class from the used dataset. Figure 6 show the ROC curve \([9]\) of the detection rate according to different vale of detection threshold \(\delta\). This parameter is used to control the output of the network, and determine from witch value we consider it as intrusion.

The obtained results demonstrate that we can achieve relatively the same detection performances with very less computation resources and time. The Table 10 compare the obtained performance with the full learning method and the clustering-based proposed one. It is clear that our goal to reduce the computation requirement is achieved.

### Table 6: A Comparative Summary of the Detection Rates for Each Attack Class

<table>
<thead>
<tr>
<th>Classification method</th>
<th>Dos DR</th>
<th>FP</th>
<th>Prob DR</th>
<th>FP</th>
</tr>
</thead>
<tbody>
<tr>
<td>KDD cup Winner [5]</td>
<td>0.971</td>
<td>0.003</td>
<td>0.833</td>
<td>0.006</td>
</tr>
<tr>
<td>SOM map [6]</td>
<td>0.951</td>
<td>-</td>
<td>0.643</td>
<td>-</td>
</tr>
<tr>
<td>Linear GP [7]</td>
<td>0.967</td>
<td>-</td>
<td>0.857</td>
<td>-</td>
</tr>
<tr>
<td>Multi-classifier NNet</td>
<td>0.950</td>
<td>0.001</td>
<td>0.876</td>
<td>0.020</td>
</tr>
<tr>
<td>Gaussian classifier [8]</td>
<td>0.824</td>
<td>0.009</td>
<td>0.902</td>
<td>0.113</td>
</tr>
<tr>
<td>K-means clustering [8]</td>
<td>0.973</td>
<td>0.004</td>
<td>0.876</td>
<td>0.026</td>
</tr>
<tr>
<td>Nearest cluster algo. [8]</td>
<td>0.971</td>
<td>0.003</td>
<td>0.888</td>
<td>0.005</td>
</tr>
<tr>
<td>Radial basis [8]</td>
<td>0.750</td>
<td>0.002</td>
<td>0.932</td>
<td>0.188</td>
</tr>
<tr>
<td>C4.5 decision tree [8]</td>
<td>0.970</td>
<td>0.003</td>
<td>0.808</td>
<td>0.007</td>
</tr>
</tbody>
</table>

### Table 7: Performances Results for the NNET 2-CATEGORY APPROACH

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detection rate</td>
<td>93.02%</td>
</tr>
<tr>
<td>False alarm rate</td>
<td>1.5%</td>
</tr>
<tr>
<td>Execution run time</td>
<td>22 hour 38 minute</td>
</tr>
</tbody>
</table>

### Table 9: Detailed Detection Rate for the Learned Classes

<table>
<thead>
<tr>
<th>Samples</th>
<th>Attacks detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>60593</td>
</tr>
<tr>
<td>Dos</td>
<td>229853</td>
</tr>
<tr>
<td>Pbr</td>
<td>4166</td>
</tr>
<tr>
<td>R2L</td>
<td>16347</td>
</tr>
<tr>
<td>U2R</td>
<td>70</td>
</tr>
</tbody>
</table>

### Table 10: Comparison of the Obtained Performances between the Proposed Methods

<table>
<thead>
<tr>
<th></th>
<th>Standard NNet detection</th>
<th>K-means learning based detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detection rate</td>
<td>93.02 %</td>
<td>92 %</td>
</tr>
<tr>
<td>False alarm rate</td>
<td>1.5 %</td>
<td>6.21 %</td>
</tr>
<tr>
<td>Execution run time</td>
<td>22 hour 38 minute</td>
<td>28 minute 21second</td>
</tr>
<tr>
<td>Training samples</td>
<td>24788 sample</td>
<td>738 sample</td>
</tr>
</tbody>
</table>

Fig 6.ROC curve for different value of the \(\delta\) threshold parameter

V. CONCLUSION AND FUTURE WORK

In this work, we study the possible use of the neural networks learning capabilities to classify and detect network intrusions from a collected dataset of network traffic trace. A multi-layered neural network was used with a back propagation feed-forward learning algorithm. The intrusion detection problem is considered as a pattern recognition one, the neural network must learn to discriminate between the attack and the normal patterns. The experiments show that the neural networks are more suitable for 2-category classification problem, the discrimination between attacks classes remain a hard task.
Since the high computation intensity and the long training cycles are the main obstacle to any neural networks IDS, we propose a new learning schema to reduce the amount of used samples using a k-means clustering algorithm. The input data are automatically clustered to a fixed number of clusters and the new samples set is constructed with the centroids of the obtained clusters and their relative boundaries; this will permit to give a maximum coverage of the initial space region occupied by the class data. The technique is independent of the dataset and structures employed, and can be used with any real values training dataset.

The proposed system is shown to be capable of learning attack and normal behaviour from the training data and make accurate predictions on the test data, in very less runtime, and with reasonable computation requirements. According to the obtained results, it can be asserted that substantial improvements of the NN-IDS performance are feasible, even if other classification methods can perform better.

In terms of future work, more work must be performed to find an optimal way to determine the number of used clusters and selected samples of each class. This work use only heuristics and trays to determine these parameters. A statistical study of the information distribution of the information in each class seem to be a good appropriate approach.

REFERENCES


K.M. Farooun Was born in Sidi Bel abbès, Algeria, in February 23, 1978. Earned a master’s degree in computer science at the computer science department of Djilali Liabbes University- Sidi-Bel-abbès – Algeria in 2002. His current research areas include computer safety systems; genetic algorithms, fractal images compression evolutionary programming and grammatical inferences and physical materials structures modeling. He is currently a teacher at the computer sciences Institute of Djilali Liabbes University, he teaches operational researches and human-machine interaction, and is actually preparing his Ph.D thesis in the field of computer security using artificial intelligence systems. He has published several papers in international journals: Genetic Fractal images compression (Multidimensional Systems and Signal Processing, Springer Science 2005).

Dr. Farooun is a member of the Evolutionary Engineering and Distributed Information Systems Laboratory, EEDIS.

A. Boukelfif earned a bachelor of science in electrical engineering from the University of Pittsburgh and a Ph.D. honours degree in electrical engineering, image processing option. He is currently an assistant professor at the University of Sidi Bel Abbes and head of a research team dealing with information and communication technologies applied to distant learning. His main research areas include digital television, digital image compression, satellite communications information and communication technologies (ICTs ). He is the author of many publications, including HDTV (Centre National des Etudes en Télécommunications, Paris, 1994), Digital Television Techniques (Masson and Paris, 1997), and Image Compression Techniques (Algiers, 2004).

Prof. Boukelfif is a member of the Telecommunication and digital signal processing laboratory at the Sidi bel abbes university.