World of Computer Science and Information Technology Journal (WCSIT)
ISSN: 2221-0741
Vol. 2, No. 3, 105-109, 2012

Learning Vector Quantization (LVQ) and k-Nearest Neighbor for Intrusion Classification

Reyadh Shaker Naoum
Department of Computer Science
Faculty of Information Technology
Middle East University
Amman, Jordan

Zainab Namh Al-Sultani
Department of Computer Science
Faculty of Information Technology
Middle East University
Amman, Jordan

Abstract— Attacks on computer infrastructure are becoming an increasingly serious problem nowadays, and with the rapid expansion of computer networks during the past decade, computer security has become a crucial issue for protecting systems against threats, such as intrusions. Intrusion detection is an interesting approach that could be used to improve the security of network system. Different soft-computing based methods have been proposed in recent years for the development of intrusion detection systems. This paper presents a composition of Learning Vector Quantization artificial neural network and k-Nearest Neighbor approach to detect intrusion. A Supervised Learning Vector Quantization (LVQ) was trained for the intrusion detection system; it consists of two layers with two different transfer functions, competitive and linear. Competitive (hidden) and output layers contain a specific number of neurons which are the sub attack types and the main attack types respectively. k-Nearest Neighbor (kNN) as a machine learning algorithm was implemented using different distance measures and different k values, but the results demonstrates that using the first norm instead the second norm and using k=1 gave the best results among other possibilities. The experiments and evaluations of the proposed method have been performed using the NSL-KDD 99 intrusion detection dataset. Hybrid (LVQ_kNN) was able to classify the datasets into five classes at learning rate 0.09 using 23 hidden neurons with classification rate about 89%.

Keywords- Intrusion Detection System; Learning Vector Quantization; k-Nearest Neighbor.

I. INTRODUCTION

Computer System Security is defined as the process of protecting the factors for any secure computer system. Those factors are: Confidentiality, Integrity, and Availability. These three concepts from what is often referred to as the CIA triad [10]. Information security is one of the cornerstones of Information Society. One mechanism of information security that has been the subject of much attention in recent years is the intrusion detection systems, or IDS [3]. Stallings & Brown (2008) [10] defined Intrusion Detection System (IDS) as a security service that monitors and analyzes system events for the purpose of finding, and providing real time or near time warning, attempts to access system resources in an unauthorized manner. Intrusion Detection Systems are classified according to their deployment either host-based or network-based. Host- based systems use system logs and operating system audit trials to detect intrusions. Network-based systems analyze network packets. There are two primary models to detect events: misuse detection model and anomaly detection model. Misuse detection model detects intrusions by looking for activity that corresponds to known signatures of intrusions. Anomaly detection model detects intrusions by finding abnormal network connection [9].

II. LITERATURE SURVEY

Jawhar and Mehrotra (2010) [2] designed a hybrid intrusion detection system of fuzzy logic and neural network. Fuzzy C-Means (FCM) clustering algorithm was used to detect normal records while Multi-Layer Perceptron (MLP) was used to detect attacks (4 attack types). The MLP was trained using the resilient backpropagation. Their results were extremely high; the detection rate was 99.9% with false positive rate of about 0.01%. The main drawbacks of the proposed paper that the distribution of the records in the training dataset is not close to be equal between classes, specifically using 23084 of DoS attack (Denial of Service), 7 U2R (User to Root), 608 R2L (Root to Local) and 1301 Prob. will automatically lead the classifier to detect DoS and it will be very difficult to detect U2R because it's only 7 records. Salama (2010) [8] proposed a hybrid system of Support Vector Machine (SVM) and Deep Belief Network (DBN). DBN was used firstly for feature reduction process where the original feature columns were reduced from 41 to 5 columns. SVM is a classification technique based on Statistical Learning Theory (SLT). It is based on the idea of a hyper plane classifier, where it first maps the input vector into a higher dimensional feature space and then obtains the optimal separating hyper-plane. The goal of

SVM is to find a decision boundary (i.e. the separating hyper-plane) so that the margin of separation between the classes is maximized. SVM was used to classify the reduced patterns to 5 classes: Normal, DoS, U2R, R2L and Prob. They have used NSL KDD dataset for evaluation and they achieved a classification rate of 92.84%. Naoum, Abid and Al-Sultani (2012) [6] proposed a hybrid intrusion detection system based on k-Nearest Neighbor and an enhanced resilient backpropagation artificial neural network. An optimal learning factor was derived to enhance the performance (speed up the convergence) of the enhanced resilient backpropagation. First Norm was used in the k-Nearest Neighbor implementation instead of Euclidean distance and they have used the first nearest neighbor where k equals 1. The enhanced resilient backpropagation neural network trained using an optimal number of hidden layers and neurons; therefore it was trained with only one hidden layer and 34 hidden neurons. The evaluation was performed on the NSL-KDD99 anomaly intrusion detection dataset. The proposed system has a classification rate (5 classes) of 97.2% with false negative rate of about 1%.

III. Proposed System Architecture

The proposed hybrid system is based on two phases of classification using Learning Vector Quantization as the first phase and k-Nearest Neighbor as the second phase. The NSL dataset was used for system evaluation. Below figure represents the hybrid system workflow:

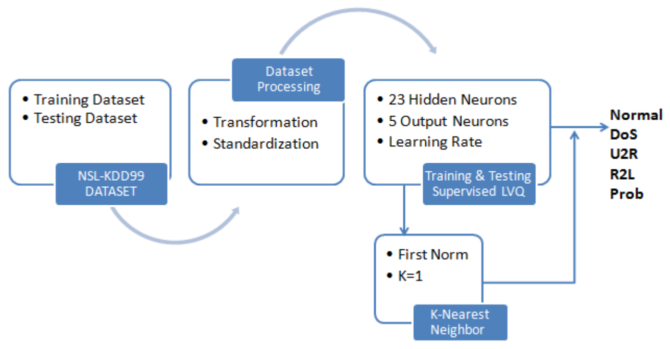

Figure 1. Proposed Hybrid Model (LVQ_kNN)

A. Data –Preprocessing Unit

NSL-KDDCup99 dataset suffers from two problems, first, it contains symbolic values and second, each column contains variety of numbers. To overcome these problems, symbolic columns must be converted to a numeric values and the values in each column must be converted to reasonable range using standardization.

1) Transformation

NSL-KDDCup99 dataset contains of 41 feature columns where 4 columns contains symbolic values. Each symbolic column must be converted to numeric using a customize transformation table.

Each column has its own customization table, depending on the column content. For example after analyzing the protocol column, it has been shown that it has three protocols values: TCP, UDP, & ICMP. Therefore the transformation table will be:

TABLE I. PROTOCOL TRANSFORMATION TABLE

Protocol -2	**No**
TCP	1
UDP	2
ICMP	3

2) Normalization

Normalization often means subtracting a measure of location and dividing by a measure of scale. Each row vector in the training and testing matrices are processed by mapping each row's means to 0 and deviations to 1.

B. Training and Testing Learning Vector Quantization (LVQ) Unit

In this paper a Learning Vector Quantization was trained to detect intrusions as the first step. Learning Vector Quantization (LVQ) is a prototype-based supervised classification algorithm [4].

It is a precursor to Self-organizing maps (SOM) and related to neural gas, and to the k-Nearest Neighbor algorithm (k-NN). LVQ was invented by Teuvo Kohonen [4].

Learning vector quantization (LVQ) is a method for training competitive layers in a supervised manner (with target outputs). A competitive layer automatically learns to classify input vectors. However, the classes that the competitive layer finds are dependent only on the distance between input vectors. If two input vectors are very similar, the competitive layer probably will put them in the same class. There is no mechanism in a strictly competitive layer design to say whether or not any two input vectors are in the same class or different classes [5]. LVQ networks, on the other hand, learn to classify input vectors into target classes chosen by the user.

An LVQ network has a first competitive layer and a second linear layer. The competitive layer learns to classify input vectors in much the same way as the competitive layers of Self-Organizing Feature Maps. The linear layer transforms the competitive layer's classes into target classifications defined by the user. The classes learned by the competitive layer are referred to as subclasses and the classes of the linear layer as target classes [5].

Both the competitive and linear layers have one neuron per (sub or target) class. Thus, the competitive layer can learn up to S1 subclasses. These, in turn, are combined by the linear layer to form S2 target classes. (S1 is always larger than S2) [5].

In the training process of LVQ different computational paradigms were used. First we have mentioned before that LVQ only consists of one competitive (hidden) layer and one output layer containing sub-class and main-class neurons respectively. Therefore in the competitive and output layers 23 (normal and 22 sub attack types) and 5 neurons were used respectively.

During the training of LVQ it was clearly that LVQ highly affected about how many patterns corresponding for each main class, therefore in designing the training dataset it was very critical to select approximately equal numbers of patterns to represent each class otherwise the LVQ will classify one main class and neglects the others.

Learning rate and number of epochs was selected iteratively where the best performance is the main criteria.

After the training process, the LVQ is ready to classify the testing dataset. LVQ will classify the dataset into five classes (Normal, Dos, U2R, R2L and Prob). Then the results of LVQ will be combined later with the results of the k-Nearest Neighbor classifier to provide maximum classification rates.

C. K-Nearest Neighbor machine learning algorithm Unit

In this stage the kNN classifier is used to classify the NSL-KDD dataset into 5 classes (Normal, DoS, U2R, R2L and Probe). The neighbor distance was measured using first norm instead of second norm (Euclidean distance) and first nearest neighbor was used instead of using large value of k. Using large values for k will lead destroys the locality of the estimation and in addition, it will increases the computational burden [11].

k- Nearest Neighbor classification algorithm procedure:

1. Store NSL-KDD99 training dataset with its corresponding label.
2. For each connection in the NSL-KDD99 testing dataset calculate the norm difference (distance) with every connection in the training set using first norm. The maximum absolute column sum norm $\|A\|_1$ is defined as [7]:

$$\|A\|_1 = \max_j \sum_{i=1}^{n} |a_{ij}|$$

3. Sort the distances in ascending order. Then with k=1 select the first minimum nearest neighbor.
4. Select the corresponding label for the nearest example. This label is the prediction for this test example.
5. Repeat the above procedure for all the connections in the testing dataset.

After classifying the testing dataset into 5 classes using the k-Nearest Neighbor, the testing dataset results for the LVQ will be combined with the results of the kNN classifier.

The results for both stages and for the hybrid system will be described in details in the following section.

IV. Experiment Results

In this paper learning vector quantization (LVQ) and k-Nearest Neighbor was used to detect intrusion. NSL-KDD99 dataset [1] was used in this paper. Training dataset was used to tune the weights and testing dataset was used for the network evaluation. Testing set contains some attacks that it is not represented in the training set. The testing dataset details are shown in the table below:

TABLE II. Testing Dataset Details

Dataset	Testing Dataset
Normal	1000
Dos	1200
U2R	37
R2L	500
Prob.	1200
All	3937

LVQ network as the first classifier was trained using the parameters shown in the table below:

TABLE III. The designed LVQ Training Parameters

Parameters	**Details**
Learning	Supervised
Input Nodes	Input Dimensionality: 41
Hidden Nodes	Number of sub-classes: 23
Output Nodes	Number of Main classes: 5
Distance Function	Negative Euclidean Distance (**negdist**)
Transfer Function in the Hidden Layer	Competitive Transfer Function (**compet**)
Transfer Function in the output layer	Linear Transfer Function (**purelin**)
Learning Function	Learning Vector Quantization 1
Training Function	Random Weight/Bias Rule
Learning Rate	0.09
Number of epochs	6
Network Performance	Mean Square Error (**MSE**)

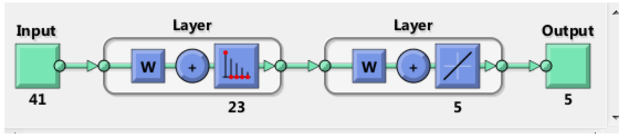

Figure 2. LVQ Architecture with 23 hidden nodes

Above figure represents the network model architecture that includes 41 input nodes which is the data dimensionality, 23 nodes in the competitive (hidden) layer that represents the subclass number and finally 5 nodes in the output (linear) layer which represents the number of main classes. The output should be either normal or main attack types (DoS, U2R, R2L & Prob.).

Following tables demonstrate the confusion matrix and the detection rate for each class using LVQ:

TABLE IV. LVQ CONFUSION MATRIX RESULTS

	LVQ Results					
Target	**1**	**2**	**3**	**4**	**5**	**Total**
1	387	39	478	89	7	1000
2	28	966	0	206	0	1200
3	2	1	30	4	0	37
4	248	8	152	53	39	500
5	0	1039	0	11	150	1200
Total	**665**	**2053**	**660**	**363**	**196**	**3937**

TABLE V. LVQ CLASS DETECTION RATES RESULTS

Dataset	**Testing Dataset**	**Detected**	**Percentage**
Normal	1000	387	39%
Dos	1200	966	81%
U2R	37	30	81%
R2L	500	152	30%
Prob.	1200	150	13%
All	3937	1586	40%

The detection rates for each class varied using LVQ, as shown in the table above the detection rates for normal, R2L and Prob was severely low but on the other hand the detection rate for the U2R attack was very good considering that this specific attack is very difficult to detect because the training dataset doesn't have enough records for this attack, therefore 81% is a very good result. Finally DoS attack has a good detection rate with 81%. The results of five classes will be combined with the results of the k-Nearest Neighbor classifier and the detection rates for all five classes will either rise up or stay the same.

As the second classifier k-Nearest Neighbor machine learning algorithm was used. First norm and second norm (Euclidean distance) were used in implementing kNN. First Norm demonstrates better result than the second norm (Euclidean distance). Tables below represent the confusion matrix and detection rates for the second norm (Euclidean distance).

TABLE VI. K-NEAREST NEIGHBOR (SECOND NORM) CONFUSION MATRIX RESULTS

	kNN Results – Second Norm					
Target	**1**	**2**	**3**	**4**	**5**	**Total**
1	920	23	4	14	39	1000
2	7	1048	0	0	145	1200
3	14	0	16	6	1	37
4	247	3	8	191	51	500
5	0	89	6	3	1102	1200
Total	**1188**	**1163**	**34**	**214**	**1338**	**3937**

TABLE VII. K-NEAREST NEIGHBOR (SECOND NORM) CLASS DETECTION RATES RESULTS

Dataset	**Testing Dataset**	**Detected**	**Percentage**
Normal	1000	920	92%
Dos	1200	1048	87%
U2R	37	16	43%
R2L	500	191	38%
Prob.	1200	1102	92%
All	3937	3277	83%

Using the first norm as the distance measure gave better results. The following tables represent the confusion matrix and class detection rates respectively for the first norm. The k-Nearest Neighbor demonstrates better results in classifying normal, DoS, R2L and Prob than LVQ, but the classification rate for R2L is still poor.

TABLE VIII. K-NEAREST NEIGHBOR (FIRST NORM) CONFUSION MATRIX RESULTS

	kNN Result – First Norm					
Target	**1**	**2**	**3**	**4**	**5**	**Total**
1	929	19	3	13	36	1000
2	7	1064	0	0	129	1200
3	12	0	18	6	1	37
4	238	10	18	190	44	500
5	0	44	4	0	1152	1200
Total	**1186**	**1137**	**43**	**209**	**1362**	**3937**

TABLE IX. K-NEAREST NEIGHBOR (FIRST NORM) CLASS DETECTION RATES RESULTS

Dataset	**Testing Dataset**	**Detected**	**Percentage**
Normal	1000	929	93%
Dos	1200	1064	89%
U2R	37	18	49%
R2L	500	190	38%
Prob.	1200	1152	96%
All	3937	3353	85%

Combining the results for both classifiers the detection rates for the five classes were improved in the hybrid system, specifically the denial of service attack detection rate rise up to 99%, while the root to local attack still suffers from low detection rate with only 39%.

TABLE X. HYBRID (LVQ_KNN) CLASS DETECTION RATES RESULTS

Dataset	Testing Dataset	Detected	Percentage
Normal	1000	938	94%
Dos	1200	1187	99%
U2R	37	30	81%
R2L	500	194	39%
Prob.	1200	1157	96%
All	3937	3506	89%

The proposed hybrid system (LVQ_kNN) performance is compared in details to the hybrid system (ERBP_kNN) proposed by [6].

TABLE XI. HYBRID (LVQ_KNN) VS. HYBRID (KNN_ERBP)

Dataset	Hybrid (kNN_ERBP)	%	Hybrid (LVQ_kNN)	%
Normal	929 - 1000	92.9%	938 – 1000	94%
Dos	2153 – 2200	97.9%	1187 -1200	99%
U2R	22 – 37	59.9%	30 – 37	81%
R2L	2125 – 2200	96.6%	194 – 500	39%
Prob.	2196 – 2200	99.8%	1157 – 1200	96%
All	7425 - 7637	97.2%	3506 - 3937	89%
Recall	99%		97.6%	
False Negative Rate	1%		2.4%	

V. CONCLUSION

In this paper a Learning Vector Quantization networks and k-Nearest Neighbor was trained to detect intrusion. The main issue in LVQ training that it requires a long time to be trained especially when comparing to other networks such as Multilayer Perceptron or Self Organizing Maps they require much less time to be trained. To train LVQ effectively it requires the size of the classes to be equally likely. Results show that the network was able to classify intrusion into Normal or Attacks when reasonable number of patterns for each class was used. The Testing rate error was greater than the training rate error at all learning rates means that the network was probably over-fitting. One way to solve this problem is to change the number of hidden neurons. LVQ is very interesting neural network especially because it combines the power of competitive function and the supervised ability for having a desired output. k- Nearest Neighbor on the other hand was able to classify intrusions with reasonably good detection rates except for the root to local attack. K-Nearest Neighbor parameters were selected according to the best classification performance; first norm had better results than the Euclidean distance. The power of integrating both supervised learning methods (LVQ and kNN) demonstrates that five classes were able to be identified; in addition this hybrid system can be improved by adjusting the learning rates and number of iterations.

REFERENCES

[1] Information Security Center of eXcellence (ISCX), The NSL-KDD Data Set, 2009. Retrieved October 26, 2011, from http://www.iscx.ca/NSL-KDD/

[2] M. Jawhar & M. Mehrotra, "Design Network Intrusion Detection System using hybrid Fuzzy-Neural Network", International Journal of Computer Science and Security, 4(3), 2010. Retrieved March 3, 2012, from http://www.cscjournals.org/csc/manuscript/Journals/IJCSS/volume4/Issue3/IJCSS-288.pdf

[3] S. Kholfi, M. Habib, & H. Aljahdali, "Best hybrid classifiers for intrusion detection", Journal of Computational Methods in Sciences and Engineering, 6 (5, 6), 2006. Retrieved March 11, 2011, from http://cit.tu.edu.sa/~sultan/opm.pdf

[4] Learning Vector Quantization. (n.d.). Retrieved January 5, 2012, from www.wikipedia.com

[5] MathWorks Matlab Help, "Learning Vector Quantization Networks", 2011.

[6] R. Naoum, N. Abid, & Z. Al-Sultani, "A Hybrid Intrusion Detection System Based on Enhanced Resilient Backpropagation Artificial Neural Network and K-Nearest Neighbor Classifier", International Journal of Academic Research IJAR, 4 (2), 2012. Retrieved from http://www.ijar.lit.az/en.php?go=march2012

[7] R. Naoum, Artificial Neural Network [Acrobat Reader], Middle East University (MEU), Jordan, 2012.

[8] M. Salama, H. Eid, R.Ramadan, A. Darwish, & A. Hassanien, "Hybrid Intelligent Intrusion Detection Scheme", 15th Online World Conference on Soft Computing in Industrial Applications, 15th to 27th November 2010, Springer in "Advances In Intelligent and Soft Computing, 2010, Retrieved March 14, 2012, from http://rabieramadan.org/papers/103_Hybrid%20Intelligent%20Intrusion%20Detection%20Scheme.pdf

[9] SANS Institute, Application of Neural Networks to Intrusion Detection, 2001. Retrieved October 25, 2011, from http://www.sans.org/reading_room/whitepapers/detection/application-neural-networks-intrusion-detection_336

[10] W. Stallings, & L. Brown, Computer Security Principals and Practice. USA: Pearson Education, 2008.

[11] TAMU Computer Science and Engineering (Texas A&M). (n.d.). PRISM Lectures. Retrieved March 12, 2012, from http://research.cs.tamu.edu/prism/lectures/iss/iss_l12.pdf