System for Intrusion Detection with Artificial Neural Network

José Ernesto Luna Domínguez, Anabelem Soberanes Martín and Cristina Juárez Landín
*University Center UAEM Chalco Valley, Hermenegildo Galeana #3, Col. María Isabel,
Chalco Valley, Estate of Mexico, Mexico*

Keywords: Artificial Neural Networks, Intrusion Detection, Multilayer Perceptron, Training Strategies.

Abstract: With the rapid expansion of computer networks during the past decade, security has become a crucial issue for computer systems. Different soft-computing based methods have been proposed in recent years for the development of intrusion detection systems. This paper presents a neural network approach to intrusion detection. A Multi-Layer Perceptron (MLP) is used for intrusion detection based on an off-line analysis approach. While most of the previous studies have focused on classification of records in one of the two general classes - normal and attack, this research aims to solve a multi class problem in which the type of attack is also detected by the neural network. Different neural network structures are analyzed to find the optimal neural network with regards to the number of hidden layers. An early stopping validation method is also applied in the training phase to increase the generalization capability of the neural network. The results show that the designed system is capable of classifying records with about 91% accuracy with two hidden layers of neurons in the neural network and 87% accuracy with one hidden layer.

1 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 an Achilles' heel: 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 (Vigna, 2002). 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, an anomaly detection based IDS detects 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 using Denning's (Denning, 1987) profile model. 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.

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) (Bonissone, 2005). 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

Luna J., Martín A. and Landín C..
System for Intrusion Detection with Artificial Neural Network.
DOI: 10.5220/0005256704700475
In *Proceedings of the International Conference on Agents and Artificial Intelligence* (ICAART-2015), pages 470-475
ISBN: 978-989-758-074-1

abnormal and normal connection audit records, and to generalize the patterns to new (and slightly different) connection records of the same class.

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 (Cannady, 2006) 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. This feature enables the system to suggest proper actions against possible attacks. The promising results of the present study show the potential applicability of ANNs for developing practical IDSs.

Different structures of MLP are examined to find a minimal architecture that is reasonably capable of classification of network connection records. The results show that even an MLP with a single layer of hidden neurons can generate satisfactory classification results. Because the generalization capability of the IDS is critically important, the training procedure of the neural networks is carried out using a validation method that increases the generalization capability of the final neural network.

2 MANUSCRIPT PREPARATION

The 1999 version of MIT Lincoln Laboratory–DARPA (Defense Advanced Research Projects Agency) intrusion detection evaluation data was used in this research (Vigna, 2002). The sample version of the dataset included more than 450,000 connection records. A subset of the data that contained the desired attack types and a reasonable number of normal events were selected manually. The final dataset used in this study included 20,055 records.

2.1 Attack Types

There are at least four different known categories of computer attacks including denial of service attacks, user to root attacks, remote to user attacks and probing attacks (Kendall, 1999). Two different attack types were included in the dataset used for this study: *SYN Flood (Neptune)* and *Satan.* These two attack types were selected from two different attack categories (denial of service and probing) to check for the ability of the intrusion detection system to identify attacks from different categories. Availability of enough data records was the other factor in choosing these two specific types. Furthermore, there are studies that have used the same attack types (Cannady, 2006) Therefore, evaluation of the results by comparing them to previous studies was possible. In the following paragraphs, a description of the attack types is provided.

SYN Flood (Neptune) is a denial of service attack to which every TCP/IP implementation is vulnerable (to some degree). For distinguishing a Neptune attack network traffic is monitored for a number of simultaneous SYN packets destined for a particular machine. The host sending these packets is usually unreachable (Kendall, 1999).

Satan is a probing intrusion which automatically scans a network of computers to gather information or find known vulnerabilities. The network probes are quite useful for attackers planning a future attack (Kendall, 1999).

Table 1: Distribution of data vectors in different subsets for training, validation, and testing sets.

Record Types	Training SET	Validation Set	Test Set
Normal	4,752	200	3,824
Neptune	3,360	200	2,231
Satan	2,625	200	1,214

Table 1 shows detailed information about the number of records from normal and two attack types included in training, validation, and testing sets. There were 7,725 records of normal connections, 5,251 records of Neptune attack, and 2,474 records of Satan attack in the dataset.

In DARPA dataset each event (connection) is described with 41 features. 22 of these features describe the connection itself and 19 of them describe the properties of connections to the same host in last two seconds. In many attack scenarios, the signature of the attack record is identified through examination of some features in a sequence of records. Therefore, the IDS should analyze the service types used by the same user in previous connections and for this purpose these 19 features describing past events in the computer network were included in the feature vector.

A complete description of all 41 features is available (Mukkamala, 2002), (Vigna, 2002). Instead of describing all the features, here we divide them into three groups and provide descriptions and examples for each group.

Group 1 includes features describing the *commands* used in the connection (instead of the commands themselves). These features describe the aspects of the commands that have a key role in defining the attack scenarios. Examples of this group are number of file creations, number of operations on access control files, number of root accesses, etc.

Group 2 includes features describing the *connection specifications*. This group includes a set of features that present the technical aspects of the connection. Examples of this group include: protocol type, flags, duration, service types, number of data bytes from source to destination, etc.

Group 3 includes features describing the *connections to the same host in last 2 seconds.* Examples of this group are: number of connections having the same destination host and using the same service, % of connections to the current host that have a rejection error, % of different services on the current host, etc..

During inspection of the data it turned out that the values of six features (land, urgent, num_failed_logins, num_shells, is_host_login num_outbound_cmds) were constantly zero over all data records (see (Mukkamala, 2002) for descriptions). Clearly these features could not have any effect on classification and only made it more complicated and time consuming. They were excluded from the data vector. Hence the data vector was a *35 dimensional* vector Different possible values for selected features were extracted and a numerical value was attributed to each of them. For example, for the protocol type the possible numerical values were: tcp=0, udp=1, icmp=2. This numerical representation was necessary because the feature vector fed to the input of the neural network has to be numerical.

The ranges of the features were different and this made them incomparable. Some of the features had binary values where some others had a continuous numerical range (such as duration of connection). As a result, the features were *normalized* by mapping all the different values for each feature to [0, 1] range.

2.2 Implementation: Training and Validation Method

The present study was aimed to solve a multi class problem. Here, a three class case is described which can be extended to cases with more attack types. An output layer with three neurons (output states) was used: [1 0 0] for normal conditions, [0 1 0] for Neptune attack and [0 0 1] for the Satan attack. 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 [1 1 0] which were considered irrelevant. It is straightforward to show that there are 6 possible irrelevant cases.

In this paper, a three layer neural network means a neural network with two 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 (Theodorios, 1999). 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.

The performance of a 2 layer neural network is seldom reported in the previous studies as described in Section II. One of the objectives of the present study is to evaluate the possibility of achieving the same results with this less complicated neural network structure. Using a less complicated neural network is more computationally efficient. Also it would decrease the training time. MATLAB^{TM} Neural Network Toolbox was used for the implementation of the MLP networks. Using this tool one can define specifications like number of layers, number of neurons in each layer, activation functions of neurons in different layers, and number of training epochs. Then the training feature vectors and the corresponding desired outputs can be ted to the neural network to begin training.

All the implemented neural networks had 35 input neurons (equal to the dimension of the feature vector) and three output neurons (equal to the number of classes). Number of the hidden layers and neurons in each were parameters used for the optimization of the architecture of the neural network. Error back-propagation algorithm was used for training.

One problem that can occur during neural network training is over-fitting. In an over fitted ANN, the error (number of incorrectly classified patterns) on the training set is driven to a very small value, however, when new data is presented, the error is large. In these cases, the ANN has memorized the training examples; however, it has not learnt to generalize the solution to new situations.

One possible solution for the over-fitting problem is to find the suitable number of training epochs by trial and error. In this study, the training time was too long (25 hours in the first experiment). Therefore, it was not reasonable to find the optimal number of epochs by trial and error. A more reasonable method for improving generalization is called early stopping. In this technique, the available data is divided into three subsets. The first subset is the training set, which is used for training and updating the ANN parameters. The second subset is the validation set. The error on the validation set is monitored during the training process. The validation error will normally decrease during the initial phase of training similar to the training set error. However, when the ANN begins to over-fit the data, the error on the validation set will typically begin to rise. When the validation error increases for a specified number of iterations, the training is stopped, and the weights that produced the minimum error on the validation set are retrieved. In the present study, this training-validation strategy was used in order to maximize the generalization capability of the ANN.

3 EXPERIMENTAL RESULTS

The first implemented intrusion detector was a three layer MLP (two hidden layers with 35 neurons in each). This structure is referred to as: {35 35 35 3}. At this stage early stopping validation was not applied and the training was performed for 200 times. The training process took more than 25 hours. Figure 1 shows the mean square error of the back propagation training process versus the progress of training epochs. The error clearly decreased to an outstanding level (comparable to zero). Therefore, it was expected to have good classification results. The final correct classification rate on training set confirmed this theory: it was very close to 100%. However, when unseen data (test set) was fed to the neural network, the result was undesirable. The correct classification rate was less than 80%.

3.1 Application of Early Stopping Validation Method

The initial result was a clear indication of over-fitting of the neural network. As explained, the reasonable solution was to define a validation data set and monitor the classification error on this data set while the neural network was being trained. The validation set used in this study consisted of 900 data records (300 of each class). The same neural network {35 35 35 3} was trained this time by applying early stopping validation method. Figure 2 shows the error of the training process versus progress of training epochs for one training session.

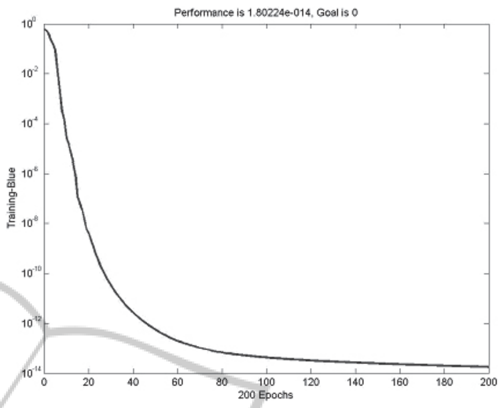

Figure 1: The mean square error of the back-propagation training procedure versus training epochs for a 3 layer neural network {35 35 35 3}.

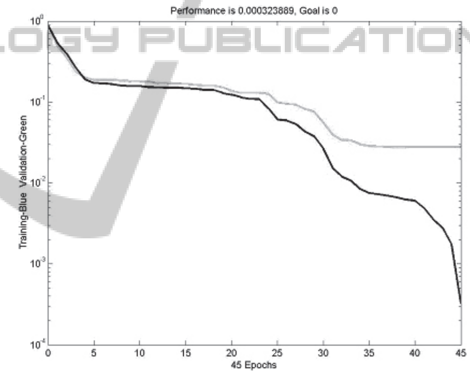

Figure 2: The training process error when the early stopping validation method is applied.

The error on the training set (darker curve) was decreasing after epoch number 45; however, the training process was stopped because the error on the validation set was constant for ten epochs.

As expected, the correct classification rate on the training set declined slightly (98% compared to 100% in the first experiment). Instead, when unseen data (test set) was fed to the neural network the result was considerably better than the first experiment in which the early stopping method was not applied. The correct classification rate was more than 90% showing an 11% increase (from 80% in the first experiment).

There was another advantage associated with application of early stopping method: the training time was decreased because the number of training epochs was restricted by early stopping. The training-validation time in this implementation was less than 5 hours which is an improvement over 25 hours training time in the first experiment?

Because of the stochastic nature of the neural networks, it is usually common to report results of multiple training- testing procedures. Table 2 illustrates the results of three training-validation-testing sessions of {35 35 35 3} MLP used in this study. The correct classification results are reported separately for training and test data sets.

Table 2: Correct classification rates in three different training.

Training Session	Correct Classification on Training Set	Correct Classification on Test Set
1	98.2	89.2
2	98.1	90.9
3	96.9	90.3
Average	97.46	90.13

3.2 Discussion

There were three categories of incorrect outputs: false positive, false negative, and irrelevant neural network output. The irrelevant outputs were those that did not represent any of the output classes in the data set (normal, Neptune attack, Satan attack). While in a two state neural network implemented with one output neuron there is no irrelevant output state, in a three output neural network, there are 6 irrelevant states. An analysis showed that in the three layer neural network with 90.9% correct classification, more than half of the incorrect results were from the category of irrelevant results. The number of incorrect classifications of this category can be decreased by classifying each irrelevant pattern in the class corresponding to the output neuron that has the highest value of activation function.

Are the results presented in the previous section satisfactory? To answer this question, they should be compared to the results of similar studies. In a previous study (Mukkamala, 2002), a result of more than 99% correct classification on this dataset using the neural network structure {41-40-40-1} was reported. However, a two class problem was solved in which the records were classified either in normal or in attack classes. In another similar study with different dataset (Cannady, 2006), the success rate was comparable to the results of the present study (89-99%) and again a two class problem was implemented.

4 CONCLUSION AND FUTURE WORK

An approach for a neural network based intrusion detection system, intended to classify the normal and attack patterns and the type of the attack, has been presented in this paper. We applied the early stopping validation method which increased the generalization capability of the neural network and at the same time decreased the training time. It should be mentioned that the long training time of the neural network was mostly due to the huge number of training vectors of computation facilities. However, 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 only factor that makes the neural network off-line is the time used for gathering information necessary to compute the features.

A two layer neural network was also successfully used for the classification of connection records. Although the classification results were slightly better in the three layer network, application of a less complicated neural network was more computationally and memory wise efficient.

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. The implemented system solved a three class problem. However, 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.

REFERENCES

Becker, M., 2001. *A neural network component for an intrusion detection system.* Oakland, s.n., pp. 240-250.

Bonissone, P. P., 2005. Soft computing: the convergence of emerging reasoning technologies. *Soft Computing Journal,* 3(4), pp. 6-18.

Cannady, J., 2006. *Artificial Neural networks for misuse detection.* Arlington, s.n.

Denning, D. E., 1987. An Intrusion detection model. *IEEE*

Transactions on software engineering, pp. 222-232.

Fox, K., 2004. *A neural network approach towards intrusion detection.* Baltimore, s.n., pp. 125-134.

Kendall, K., 1999. *A database of computer attacks for the evaluation of intrusion detection systems,* s.l.: MIT.

Lippmann, R., 2000. *Improving intrusion detection performance using kerword selection and neural networks.* Purdue, s.n.

Mukkamala, S., 2002. *Intrusion detection using neural networks and support vector machine.* Honolulu, s.n.

Poole, D., 2003. *Computational Intelligence.* New York: Oxford University Press.

Sinclair, C., 1999. *An application of machine learning to network intrusion detection.* Phoenix, s.n., pp. 371-377.

Theodorios, S., 1999. *Pattern Recognition.* Cambridge: Academic Press.

Vigna, G., 2002. *Intrusion detection: a brief history and overview.* Phoenix: s.n.

Zincir, A. N., 2002. *Host-based intrusion detection using self-organizing maps.* Honolulu, s.n., pp. 1714-1719.