Proceedings of the International Multiconference on
Computer Science and Information Technology, pp. 807 – 811

ISBN 978-83-60810-14-9
ISSN 1896-7094

Analysis of Different Architectures of Neural Networks for Application in Intrusion Detection Systems

Przemysław Kukiełka
Institute of Telecommunications
Warsaw University of Technology
Nowowiejska 15/19, 00-665 Warsaw, Poland
Email: Przemyslaw.Kukielka@telekomunikacja.pl

Zbigniew Kotulski
Institute of Fundamental Technological Research
Polish Academy of Sciences
Swietokrzyska 21, 00-049 Warsaw, Poland
Email: zkotulsk@ippt.gov.pl

***Abstract*—Usually, Intrusion Detection Systems (IDS) work using two methods of identification of attacks: by signatures that are specific defined elements of the network traffic possible to identification and by anomalies being some deviations form of the network behavior assumed as normal. In the both cases one must pre-define the form of the signature (in the first case) and the network's normal behavior (in the second one). In this paper we propose application of Neural Networks (NN) as a tool for application in IDS. Such a method makes possible utilization of the NN learning property to discover new attacks, so (after the training phase) we need not deliver attacks' definitions to the IDS. In the paper, we study usability of several NN architectures to find the most suitable for the IDS application purposes.**

I.I Introduction

BECAUSE of their generalization feature, neural networks are able to work with imprecise and incomplete data. It means that they can recognize also patterns not presented during a learning phase. That is why the neural networks could be a good solution for detection of a well-known attack, which has been modified by an aggressor in order to pass through the firewall system. In that case, traditional Intrusion Detection Systems, based on the signatures of attacks or expert rules, may not be able to detect the new version of this attack.

In this paper, we focus on three different network architectures: Backpropagation, Radial Basis Function and Self Organizing Map. The result of simulation is the information about the attack detection accuracy, represented as a number of false attacks and not detected attacks in comparison to number of validation vectors for each type of used NN. Performed tests allow us to find drawback of usage of each type of architectures for detection a specific type of the attack. As the input vector, we used data produced by the KDD99 project.

II.Neural Network: a Way of Work

An artificial neural network is a system simulating a work of the neurons in the human brain. In Fig. 1 it is presented the diagram of a neuron's operation.

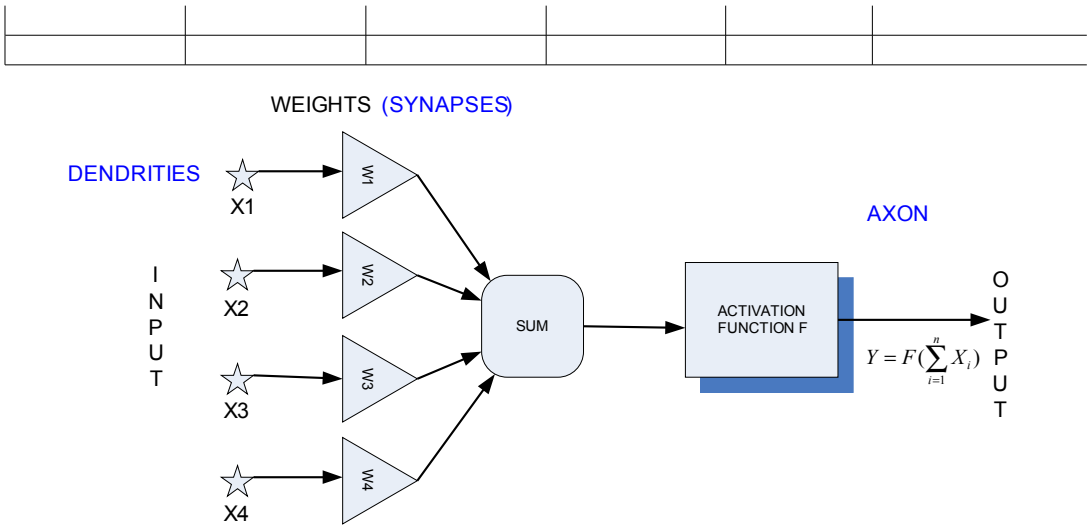

Fig. 1. A scheme of an artificial neuron

This is the work in progress

The neuron consists of some inputs emulating dendrites of the biological neuron, a summation module, an activation function and one output emulating an axon of the biological neuron. The importance of a particular input can be intensified by the weights that simulate biological neuron's synapses. Then, the input signals are multiplied by the values of weights and next the results are added in the summation block. The sum is sent to the activation block where is pro cessed by the activation function. Thus, we obtained neuron's answer for the input signals "x".

A. MLP (Multi Layer Perceptron)

One neuron cannot solve a complex problem that is why the neural network consisted of many neurons is used. One of the most often used architecture is the Multi Layer Perceptron. In such a network, all neurons' outputs of the previous layer are connected with neurons' inputs of the next layer. The MLP architecture consists of one or more hidden layers. A signal is transmitted in the one direction from the input to the output and therefore this architecture is called feedforward. The MLP networks are learned with using the Backward Propagation algorithm (BP). In order to reach better efficient and speed of learning process it arise many types of BP algorithm. In our research we used following variants of the BP algorithm:

- Batch Gradient descent (traingd)
- Batch Gradient descent with momentum (traingdm)
- Levenberg-Marquardt (trainln)
- Resilient Backpropagation (trainrp)
- Conjugate Gradient (traincgf)
- Quasi Newton (trainbfg)
- Quasi Newton 2 (trainnoss)

For the simulation procedure, the Matlab toolbox was used. The variants of BP algorithm are followed by the names of the learning Matlab's functions in the bracket.

B. Radial Based Function (RBF) Neural Network

The radial neuron networks in comparison to the MLP where a global approximation is used are working based on the local approximation.
Typically have three layers: an input layer, a hidden layer with a non-linear RBF activation function φ and a linear output layer.

Activation function for the radial neuron network is:

$$\varphi(x) = \exp\left(\frac{\|x - c_i\|^2}{2\sigma_i^2}\right),$$

where C- Center of the function σ Spread parameter of the radial function

The argument of radial function φ is the Euclidean distance sample x from center c.

In the RBF network, there are three types of parameters that need to be chosen to adapt the network for a particular task: the center vectors c_i , the output weights w_i , and the RBF spread parameter β_i.

C. SOM (Self Organizing Maps)

This kind of network is learned without teacher based on the competition rules. On the input of such network learning vector is put and in the next step distance from it to the weight vector is checked. Neuron for its distance is lowest becomes the winner and can modify values of his weights. Depends of the SOM networks architecture the weighs of the winner neuron can be modified (Winner Takes all) or weigh of the winner and neurons from his neighbor (Winner Takes most).

More information about neural network could be found in [2], [7].

III. KDD99 Input Data

We are using the KDD 99 data set as the input vectors for training and validation of the tested neural network. It was created based on the DARPA (Defense Advanced Research Project Agency) intrusion detection evaluation program. MIT Lincoln Lab that participates in this program has set up simulation of typical LAN network in order to acquire raw TCP dump data [1]. They simulated LAN operated as an normal environment, which was infected by various types of attacks. The raw data set was processed into connection records. For each connection, 41 various features were extracted. Each connection was labeled as normal or under specific type of attack. Four main categories of attacks were simulated:

- **DoS** (Denial of Service): an attacker tries to prevent legitimate users from using a service e.g. TCP SYN Flood, Smurf .
- **Probe:** an attacker tries to find information about the target host. For example: scanning victims in order to get knowledge about available services, using Operating System etc.
- **U2R** (User to Root): an attacker has local account on victim's host and tries to gain the root privileges.
- **R2L** (Remote to Local): an attacker does not have local account on the victim host and try to obtain it.

The KDD data sets are divided into tree subsets: 10%KDD, corrected KDD, whole KDD. Basis characteristic of KDD data sets are shown in Table I. It includes number of

Table I.
KDD99 data subsets

Dataset	DoS	Probe	U2r	U2l	Normal
10%KDD	391458	4107	52	1126	97277
Corrected KDD	229853	4166	70	16347	60593
Whole KDD	3883370	41102	52	1126	972780

TABLE II.
DATA SUBSETS FOR TRAINING PROCESS

Data Set name for training process	Number of normal connection	Number of connection labelled as attack .
Learn_set	1000	8653
Learn_set_radial_trad	100	1179
Learn_set_radial	1000	1179

connections assigned to the particular class (DoS, Probe etc.)..

A. 10%KDD

The 10%KDD data set is used for the training process of the IDS. It includes connections simulated following 22 types of the attacks: back, buffer_overflow, ftp_write, guess_passwd, imap, ipsweep, land, loadmodule, multihop, neptune, nmap, normal, perl, phf, pod, portsweep, rootkit, satan, smurf, spy, teardrop. The attacks are not represented by the same number of connections. The most of the simulated attacks are of DoS group because of the nature of this type attacks that are using many IP packets in order to block network services.

B. Corrected KDD

The Corrected KDD data set is used for testing process of IDS. It includes additional 14 types of new attack not presented in 10%KDD and Whole KDD. Thanks to them it is possible to check if tested IDS is able to detect new attack not presented in the training phase.

In our research only 10 %KDD and corrected KDD data set was used.

IV. TRAINING PROCESS

A. Process of Selection of the Input Vector

The data set 10% KDD includes the large number of connection. It is influence on the long time of training, high requirements for efficient of using implementation of neural network and hardware on that it is working. That's why in research presented in this publication for training purpose three randomly selected small date set was used. Table II shows how many connections are assigned to the particular training data set.

It was chosen the same number of connections represent each type of attack. In case when number of connections for particular type of attack was lower than assumed number all connections for this group was selected. Because some features of connection (e.g. protocol, flags) were existed as a characters string, it was transformed to numerical representation.

Moreover, for the SOM neural network normalization process of the input data "xi" was performed.

$$x_i' = \frac{x_i}{\sqrt{\sum_{i=1}^{n} x_i^2}}.$$

For Radial Neuron network it was noticed that the best results are obtained when the value of features is decreased by dividing it by constant number equal 1000.

For the research purpose three type of neural network architectures were used: MLP (multilayer perceptron), RBF (Radial Basis Function), SOM (Self Organizing Maps).

Main Assumption for Training Process of MLP

Number of Epochs = 1000.
MSE (Mean Square error) = 0.01.
Learning rate =1.
Momentum = 0.6.
Activation function =log-signoid.
Number of neurons in hidden layer=5.
Number of neurons in output layer =1.
Updates of weighs – batch mode (after presentation of entire training data set).

Results of the training process show that only the algorithms: **Levenberg-Marquardt and Quasi Newton** achieved to fulfill the above assumption. In other case MLP network cannot reach assumed MSE =0.1 in 1000 Epochs of training process. That is why in the next simulation we focus only on networks that can be trained according our assumptions.

B. For RBF Network

Implementation from Matlab toolbox was used (newrb). At the beginning, a network without radial neurons in the first layer is created. In the next step, the MSE is calculated and a new radial neuron with weighs equals to input vector that caused the max value of MSE is added to the first layer. The last operation is modification of weights of a lineal neuron in the output layer in direction of minimize MSE. All steps are repeated until the assumed MSE is reached.

Main Assumption for Training Process

MSE=0.1(because of high calculation power requirements not possible to achieve MSE =0.01 like for MLP)
Spread parameter σ = 3
MN (Maximum number of neurons) = 250
DF (Number of neurons to add between displays) = 2
MNE (Number of neurons for that RBF achieved MSE =0.1) = 178
Number of Epochs = 178

C. For SOM Network

Main Assumption for Training Process

Implementation of SOM in Matlab environment was used (function newsom).
Number of Epochs = 1000.
Neighbors topology= hextop
Distance function – mean as a number of links between neurons or steps that must be taken to get neuron under consideration = matlab linkdist.
Ordering Phase learning rate =0.9.

TABLE III.
THE DATA SUBSET FOR TESTING PHASE

Data Set name for testing process	Number of normal connection	Number of connection labelled as attack .
Test_set	60593	250436
Test_set_noanomaly	60593	231694

TABLE IV.
COMPARISON OF DIFFERENT NETWORK TOPOLOGY TRAINED WITH "LEARN_SET_RADIAL_TRAD" DATA SET

Neural Network topology	False alarm Number/ Percent of all normal connection	Not detected/ Detection rate	Duration of training
MLP learning algorithm **Levenberg-Marquardt**	31594 52%	382 98,5%	About 4 seconds
MLP learning algorithm **Quasi Newton**	37530 61%	90 99%	About 4 seconds
Radial	21938 36%	61568 76%	About 2 minutes 20 seconds
SOM	19677 32%	40957 84%	About 7 minutes

TABLE V.
RESULTS OF SIMULATION OF MLP NETWORK LEARNED WITH USAGE INPUT DATE SET „LEARN_SET" AND „LEARN_SET_RADIAL"

Variations of BP algorithm used for training MLP Network	False alarm Number/ Percent of all normal connection	Detection rate
MLP learning algorithm Levenberg-Marquardt **Input data set:** Learn_set, Test_set	5354 8,8%	19207 92%
MLP learning algorithm Levenberg-Marquardt **Input data set:** Learn_set, Test_noanomaly	5351 8,9%	4097 98%
MLP learning algorithm Levenberg-Marquardt **Input data set:** Learn_set_radial, Test_set	10068 16,6%	17139 93%
MLP learning algorithm Quasi Newton **Input data set:** Learn_set, Test_set	4790 7,9%	10294 94%
MLP learning algorithm Quasi Newton **Input data set:** Learn_set, Test_set_noanomaly	4790 7,9%	87 99,9%
MLP learning algorithm Quasi Newton **Input data set:** Learn_set_radial, Test_set	31894 52%	1556 99%
MLP learning algorithm Resilient Backpropagation **Input data set:** Learn_set, Test_set	4453 7,3%	12877 95%
MLP learning algorithm Resilient Backpropagation **Input data set:** Learn_set, Test_set_noanomaly	4454 7,35%	2714 99%
MLP learning algorithm Resilient Backpropagation **Input data set:** Learn_set_radial, Test_set	3564 5,9%	29805 88%

Ordering Phase steps = 1000.
Tuning phase learning rate =0.02.
Tuning phase neighbor distance = 1.

In the ordering phase neighbor distance between two neurons decreases from maximum values to the Tuning phase values, the learning rate decreases from the Ordering phase learning rate to the Tuning phase learning rate. Neuron's weights are expected to order themselves in the input space consistent with the associated neurons position.

In the Tuning phase neighbor distance stay on the same level equal Tuning phase neighbor distance, the learning rate continues to decrease but very slowly. Neuron's weights are expected to spread out ever the input space relatively evenly while retaining their topological order obtained during the ordering phase..

V. RESULTS OF THE TESTS

The neural networks architectures were tested with using whole date set from the Corrected KDD and with using modified data set that includes only attacks presented during training process. Data sets used for the testing phase were shown in Table III.

In Table IV is presented the comparison of particular neural network architecture learned with input data from „Learn_radial_trad" data set. The evaluation focuses on the

value of the detection rate and the false alarm rate for "Test_set" data set. For analysis of neural network answer, it was assumption that value from 0 to 0.5 concerning "normal" and from 0.5 to 1 "attack". The Corrected Data Set used for testing phase includes 311 029 vectors.

High number of false alarms is a result of small number of "normal" connections in the trained data set. It was reduced because of a problem of efficiency of SOM and RBF NN implementation in the Matlab environment.

The MLP network was the most efficient during comparison of different network architecture. That is why in the further research we plan to focus on it. For the training of the MLP network we used an input data set with bigger number of connection (Learn_set, Learn_set_radial). The experiment should show if increasing the number of input patterns improves detection rate and false alarms rate. The results of this test are presented in Table V.

VI. Conclusions

Usage of neural networks for intrusion detection with the input data from DARPA project was presented in many publication. Unfortunately, in description of simulation process very often is lack of information about assumptions that were made in before. For instance there is no information if the whole tests data set was used in the simulation or only some its subset.

The goal of this research was the comparison of different neural network architectures working with the same assumed parameters and tested with with usage of the whole DARPA tests data set (Corrected KDD). Our research made it possible to formulate precisely general assumptions for making benchmark simulations as well as gave us some conclusions concerning particular NN architectures applied to IDS. The main conclusions are:

- Selection of the input data is a very important issue. Representation of all types of attacks and normal activity should be included in the learning date set.
- Results of simulation show that detection rate was the best for MLP network learned with Backward Propagation Levenberg-Marquardt algorithm
- In the second phase of simulation MLP network number of input vectors are increased because of that summary amount of errors decreases. In particular number of false alarms decreases and numbers of non detected attack a little increases. The reason is that representation of more different type of normal activity was added to input vectors in learning phase.
- The long time is required for the learning phase of SOM and Redial neural network. Moreover, these two neural network architectures need very high CPU performance and Operational Memory (RAM). During tests phase input date set had to be divided into smaller subsets in order to avoid "lack of memory swap" Matlab error.
- MLP required less CPU performance and Operational Memory (RAM). That is why could be tested with using the whole tests data set not divided into smaller subsets.

References

[1] W. Lee, S. J. Stolfo, "A Framework for Constructing Features and Models for Intrusion Detection Systems", *ACM Transactions on Information and System Security* (TISSEC), 3(4): 227-261, 2000.

[2] L. Rutkowski, *Metody i techniki sztucznej inteligencji* , PWN, Warszawa 2005. (In Polish)

[3] W. Lee, S. J. Stolfo, "Data Mining Approaches for Intrusion Detection", *Proceedings of the Seventh USENIX security Symposium* (SECURITY '98), San Antonio, TX,1998.

[4] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, K. Das, "The 1999 Darpa Off-Line Intrusion Detection Evaluation", *Computer Networks: The International Journal of Computer and Telecommunications Networking* 34 (2000) 579-595, 2000.

[5] V. Paxson," Bro: A system for Detecting Network Intruder in Real Time" In *Proceedings of the 7th USENiX Security Symposium*, San Antonio 1998.

[6] Ch. Elkan, "Results of the KDD'99 Classifier-learning contest", In `http://www-cse.ucsd.edu/#elkan/clresults.html', September 1999.

[7] S. Osowski, *Sieci neuronowe do przetwarzania informacji* , Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa 2000, ISBN 83-7207-187-X. (In Polish)