International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Ingénierie des Systèmes d'Information
Vol. 25, No. 3, June, 2020, pp. 377-382
Journal homepage: http://iieta.org/journals/isi

An Intrusion Detection Method for Enterprise Network Based on Backpropagation Neural Network

Fei Chen[1*], Rui Cheng[1], Yayun Zhu[2], Siwei Miao[2], Liang Zhou[2]

[1] School of Electrical and Electronic Engineering, North China Electric Power University, Beijing 102206, China
[2] Information & Communication Department, China Electric Power Research Institute, Beijing 100192, China

Corresponding Author Email: chenfei0428@126.com

https://doi.org/10.18280/isi.250313

Received: 20 January 2020
Accepted: 8 April 2020

Keywords:
Backpropagation neural network, intrusion detection system (IDS), network security, enterprise network

ABSTRACT

Network security, as the prerequisite for the normal operation of enterprise network, should not focus on a single point, but all aspects of the network, ranging from physics, network, system, application to management. To ensure enterprise network security and prevent network attacks, it is of great importance to build an intrusion detection system (IDS) capable of protecting the network and computers from malicious attacks based on the Internet or host. In light of the above, this paper puts forward an intrusion detection method for enterprise network based on backpropagation neural network (BPNN), and carries out Python simulation of the proposed method on four problems, namely, normal state, the SYN flood (denial-of-service attack), snoop (unauthorized access from a remote host), and saint (reconnaissance attack). The simulation results show that the BPNN-based method could effectively check the network security environment, and accurately identify and detect intrusions.

1. INTRODUCTION

With the development of Internet technology, network security has become an increasingly prominent issue that can no longer be guaranteed by firewall alone. This gives rise to various intrusion detection systems (IDSs) [1]. The IDS mainly enables the enterprise to monitor the data flow in its network and identify anomalies, such that no unauthorized intruder could enter the network or computer system and cause temporary or permanent losses to the enterprise [2]. During the operation, the IDS scans the current activities in the network, monitors and records network traffic, filters the traffic from the host adapter to the network by preset rules, and issue alarms in real time.

There are two main design methods for IDSs: misuse detection and anomaly detection [3]. To detect intrusions, the IDS for misuse detection looks for activities that correspond to the signatures of known intrusions or vulnerabilities, while the IDS for anomaly detection searches for abnormal network traffic. As a reasonable supplement to firewalls, intrusion detection helps the system respond to network attacks, and enhances the capacity of system security administrators, ensuring the integrity of information security infrastructure.

Considering the various forms of network intrusions faced by enterprises, relevant experts and scholars have applied many intelligent algorithms to improve the efficiency of network intrusion detection [4]. The existing intrusion detection algorithms are based on either support vector machine (SVM) or neural networks (NNs) [5-7]. For instance, Sani and Ghasemi [8] proposed a method to learn a suitable distance function according to the set of monitored information, and measured the similarity and difference of features by the distance function based on SVM clustering. Their approach could effectively improve the performance of the IDS. Yang and Karahoca [9] developed a network intrusion detection method based on cellular neural network (CNN) model, which features a multi-dimensional array of neurons and local connections between cells, learned the templates and biases in the CNN classifier by the recursive perceptron learning algorithm (RPLA), and applied the CNN model to select and normalize features from the KDD Cup 1999 dataset; the results show that the CNN model can effectively detect intrusions, and achieve a higher attack detection rate and a lower false positive rate than the backpropagation neural network (BPNN).

Later, Yang et al. [10] developed a CNN template learning method for network intrusion detection based on tabu search (TS), in which the TS is coupled with the CNN with symmetric template, and verified the effectiveness of the method by simulation; the simulation results show that the TS-based template learning method outperforms the genetic algorithm (GA) and simulated annealing (SA) algorithm in the calculation time and quality of the optimal solution. Based on adaptive specifications, Mitchell and Chen [11] designed an IDS to detect the malicious drones in airborne systems, and proved that the IDS is more accurate and adaptive than multi-agent system (MAS) and ant colony clustering models. With the aid of Gaussian mixture model, Hu et al. [12] improved the traditional online AdaBoost intrusion detection method, and demonstrated that the improved method has a higher detection rate and a lower false alarm rate than the traditional method, which is based on decision stumps. Fung et al. [13] suggested that the collaborative intrusion detection network can effectively detect the knowledge of intrusion events, using the distributed IDS, and thus improving the detection ability of new intrusion events.

In addition, many enterprises have adopted NNs to detect intrusions into enterprise network [14, 15]. Cannady [16] designed a three-layer NN to differentiate between the normal and misuse access records offline; instead of a preliminary classifier, the designed NN is an independent system, whose results can be used in a rule-based system. Ryan et al. [17] presented a BPNN-based offline anomaly detection system, which relies on the BPNN to identify the user's configuration file and evaluate the user's commands at the end of each log session, thereby detecting possible intrusions. Ke and Hong [18] put forward a network intrusion detection algorithm, which optimizes the NN weights with GA. Specifically, the GA searches for the most suitable NN weights, and the optimized NN learns the data on network intrusion detection before detecting intrusions. Proposed by Rumelhart and McClelland in 1986, the BPNN is a multi-layer feedforward NN trained by the error backpropagation algorithm [19]. Being an important pattern recognition method, the BPNN is capable of self-organization, self-learning and generalization. If applied to the IDS, the BPNN could promote the system capabilities to identify known attacks and to detect unknown attacks.

Drawing on the above results, this paper presents an intrusion detection algorithm based on the BPNN. The proposed algorithm can distinguish attack records from normal records, and identify the type of attacks. The remainder of this paper is organized as follows: Section 2 models the BPNN-based intrusion detection algorithm, according to the features of network intrusions; Section 3 carries out an example analysis on the proposed method, revealing the effectiveness of our method; Section 4 puts forward the main conclusions.

2. BPNN-BASED INTRUSION DETECTION ALGORITHM

The BPNN is an information processing paradigm inspired by the way our brain processes information [20]. As shown in Figure 1, the BPNN is a multi-layer feedback network, consisting of an input layer, multiple hidden layers, and an output layer. The main advantage of the BPNN lies in the quick classification of highly nonlinear problems.

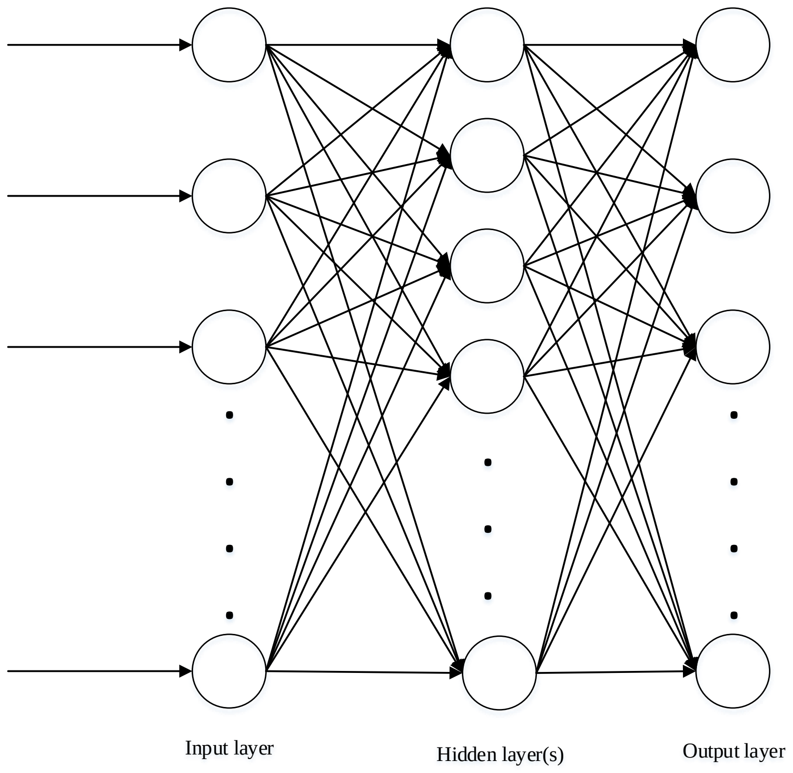


Figure 1. The structure of the BPNN

2.1 Hypotheses

The BPNN algorithm mainly divides the learning process into two stages: the forward computation and the reverse computation. During the first stage, the hidden layer(s) processes the input layer information, and computes the actual output of the output layer. During the second stage, if the output is below the expected level, the error between the actual value and the desired value will be calculated to adjust the network weights. The two stages are repeated until the output value is as desired. Mathematically, the BPNN algorithm can be described as follows:

Suppose that the BPNN has n nodes and L layers. The nodes on each layer can receive the information from the nodes on the previous layer and forward it to the nodes on the next layer. Every node satisfies the Sigmoid function.

For simplicity, it is hypothesized that the entire BPNN has only one output *y*. For *N* given samples $(x_k, y_k)(k=1, 2,\ldots, N)$, it is assumed that any node *i* outputs O_i, and the output is y_k corresponding to the input of x_k. In this case, the output of node *i* is O_{ik}. When the k-th sample is inputted to node j on the l-th layer, then the input of node j can be expressed as:

$$net_{ij}^{l} = \sum_{j} w_{ij}^{l} O_{jk}^{l-1} \tag{1}$$

$$O_{jk}^{l} = f\left(net_{jk}^{l}\right) \tag{2}$$

where, O_{jk}^{l-1} is the output of node j on the l-1-th layer at the input of the k-th sample; w_{ij} is the connection weight between nodes i and j; $f(x)$ is the Sigmoid function.

The error function can be defined as:

$$E_k = \frac{1}{2}\sum_{l}\left(y_{lk} - \overline{y}_{lk}\right)^2 \tag{3}$$

where, $\overline{y}_{lk}$ is the actual output of node j.

The total error can be computed by:

$$E = \frac{1}{2N}\sum_{k=1}^{N} E_k \tag{4}$$

It is further assumed that:

$$\delta_{jk}^{l} = \frac{\partial E_k}{\partial net_{jk}^{l}} \tag{5}$$

Then, we have:

$$\frac{\partial E_k}{\partial w_{ij}^{l}} = \frac{\partial E_k}{\partial net_{jk}^{l}}\frac{\partial net_{jk}^{l}}{\partial w_{ij}^{l}} = \frac{\partial E_k}{\partial net_{jk}^{l}} O_{jk}^{l-1} = \delta_{jk}^{l} O_{jk}^{l-1} \tag{6}$$

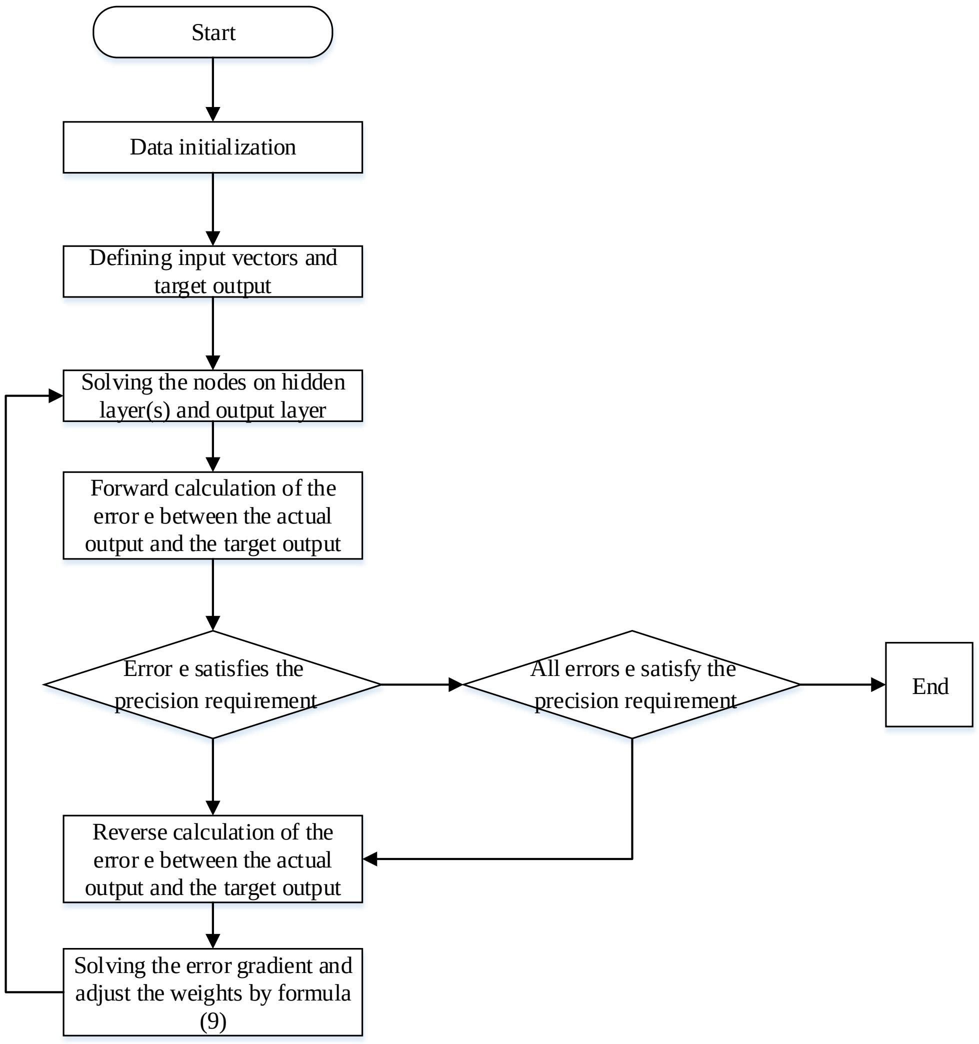
$$\delta_{jk}^{l} = \frac{\partial E_k}{\partial net_{jk}^{l}} = \begin{cases} \dfrac{\partial E_k}{\partial \overline{y}_{jk}}\dfrac{\partial \overline{y}_{jk}}{\partial net_{jk}^{l}} = -\left(y_k - \overline{y}_k\right) f'\left(net_{jk}^{l}\right), if\ node\ j\ is\ the\ output\ node, and\ O_{jk}^{l} = \overline{y}_{jk} \\ \dfrac{\partial E_k}{\partial \overline{y}_{jk}}\dfrac{\partial O_{jk}^{l}}{\partial net_{jk}^{l}} = \dfrac{\partial E_k}{\partial O_{jk}^{l}} f'\left(net_{jk}^{l}\right), if\ node\ j\ is\ not\ the\ output\ node \end{cases} \tag{7}$$

2.2 Solving algorithm

The workflow of BPNN algorithm is illustrated in Figure 2. Note that the error index is calculated iteratively until the index meets the precision requirement:

$$E = \frac{1}{2N}\sum_{k=1}^{N} E_k < \varepsilon \tag{8}$$

where, ε is the precision requirement.



Start
Data initialization
Defining input vectors and target output
Solving the nodes on hidden layer(s) and output layer
Forward calculation of the error e between the actual output and the target output
Error e satisfies the precision requirement
All errors e satisfy the precision requirement
End
Reverse calculation of the error e between the actual output and the target output
Solving the error gradient and adjust the weights by formula (9)

Figure 2. The workflow of BPNN algorithm

If k=1,2……, N, the forward calculation will compute the O_{jk}^{l-1} of the nodes on each layer; the reverse calculation will compute the O_{jk}^{l-1} and δ_{jk}^{l} of the nodes on each layer. If the error index does not meet the precision requirement, the weights will be modified according to:

$$\omega_{ij} = \omega_{ij} - \mu \frac{\partial E}{\partial w_{ij}}, \mu > 0 \tag{9}$$

And the previous step will be executed again until the index meets the requirement.

During the solving process, the training samples of the BPNN algorithm are generated randomly, and the learning rate is adjusted dynamically with the number of iterations.

3. EXAMPLE ANALYSIS

3.1 Classification of attacks on enterprise network

The enterprise network mainly faces four kinds of attacks [21]. The first kind is denial-of-service (DoS) attacks. During a DoS attack, the attacker or hacker generates a huge amount of traffic to occupy memory resources, leaving no room for legitimate network requests. In this way, any user access to the computer will be rejected. The common DoS attacks include SYN flood, teardrop attack, low-rate DoS attack, Internet Control Message Protocol (ICMP) flood, and peer-to-peer (P2P) attack.

The second kind is the unauthorized access from a remote host. During such an attack, the attacker sends data packets to the computer through the Internet, making users inaccessible. The common types include xlock, guest, snoop, etc.

The third kind is reconnaissance attacks, which collects the weaknesses in the network for further attacks. Reconnaissance attacks are divided into scanning attacks and network monitoring. Scanning attacks include port scanning, host scanning and vulnerability scanning. Meanwhile, network monitoring mainly refers to setting the network card in user's computer to promiscuous mode via software only, and then viewing important plaintext information passing through the network. The common reconnaissance attacks include ipsweep, mscan, nmap, portsweep, saint, and satan.

The fourth kind is worms, viruses and Trojan horses. Sometimes, the host is infected with malicious software, which damages the system, duplicates itself or denies access to the network, system or service. Such software is either a worm, a virus or a Trojan horse.

3.2 Problem description

The proposed intrusion detection algorithm was applied to solve four problems: normal state, the SYN flood (DoS attack), snoop (unauthorized access from a remote host), and saint (reconnaissance attack). The attacks in the problems are all typical ones that may occur in other cases.

The BPNN transforms the input into output of attack type, indicating the probability of each kind of attacks. Here, the four states of each problem are expressed in the form of a matrix, where 1 means the corresponding problem has occurred and 0 means the corresponding problem has not occurred. Hence, the matrices for the four problems can be expressed as [1, 0, 0, 0] for normal state, [0, 1, 0, 0] for SYN flood, [0, 0, 1, 0] for snoop, and [0, 0, 0, 1] for saint.

According to the states detected above, four kinds of nodes were placed in the input layer, provided that each node satisfies the Sigmoid function. To prevent the BPNN from overfitting, the early stopping criteria (ESC) [22] was introduced to divided the sample data into a training set, a verification set, and a test set. The training set was used to train and update the BPNN parameters, the verification set was used to identify the errors in the training set, and the test set was used to evaluate the goodness-of-fit of the BPNN. The principle of the ESC is that, once the overfitting occurs, the error in the verification set will increase. Thus, the training will stop at the point of error increment, and the weights at this time will be selected to check the network performance on the test set [23].

3.3 Simulation

The BPNN-based IDS was simulated based on Python. It is assumed that the first hidden layer has 50 nodes and the second has 30 nodes. The connection weights were maximum between the input layer and the first hidden layer, and minimized between the second hidden layer and the output layer.

Figure 3 shows the cross-validation error rates on the training set (red line) and the validation set (blue line). It can be seen that the ESC point appeared at the 100-th iteration on the verification set, corresponding to the cross-validation error rate of 0.10.

Then, the data training continued until the 1,000-th iteration. It is observed that the cross-validation error rate started to grow after the 100-th iteration, indicating that this point is the ESC point. The weights at this point were chosen to verify the performance of the BPNN on the test set. As shown in Table 1, the correct classification rates of the BPNN averaged at 94.50%.

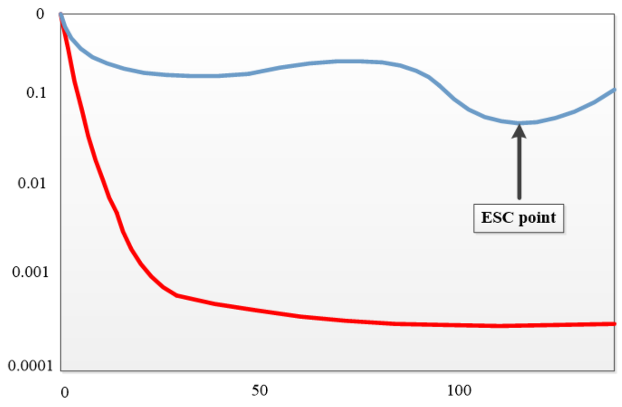


Figure 3. The cross-validation error rates on the training set and the validation set

Table 1. The correct classification rates of the BPNN on four problems and the average values

Index \ Problem	Normal state	SYN flood	snoop	saint	Average
Number of test samples	1045	1045	1045	1045	1045
Number of correctly classified samples	982	965	1018	985	987.5
Correct classification rate	93.97%	92.34%	97.42%	94.26%	94.50%

4. CONCLUSIONS

This paper probes deeply into the intrusion detection problem of enterprise network. Considering the types of enterprise network intrusions, an intrusion detection method was proposed based on the BPNN algorithm. Then, the enterprise network intrusions were classified into four kinds. After that, four problems were selected to verify the effectiveness of the BPNN-based method through Python simulation, namely, normal state, SYN flood, snoop, and saint. The simulation results show that our method can effectively detect the network security environment, and classify 94.50% of all attack states correctly. Therefore, our intrusion detection method is an excellent tool to identify and detect intrusions, providing an easy yet effective solution to the security of enterprise network.

ACKNOWLEDGMENT

This work was supported by the Science and Technology Project of State Grid Corporation of China (Grant No.: 521304190004) (Project title: Research on Key Technologies of Integrated Power Network Security Simulation and Verification Environment).

REFERENCES

[1] McHugh, J. (2000). Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by Lincoln laboratory. ACM Transactions on Information and System Security (TISSEC), 3(4): 262-294. https://doi.org/10.1145/382912.382923

[2] Kemmerer, R.A., Vigna, G. (2002). Intrusion detection: a brief history and overview. Computer, 35(4): supl27-supl30. https://doi.org/10.1109/MC.2002.1012428

[3] Lee, W., Stolfo, S.J., Mok, K.W. (1999). A data mining framework for building intrusion detection models. In Proceedings of the 1999 IEEE Symposium on Security and Privacy (Cat. No. 99CB36344), pp. 120-132. https://doi.org/10.1109/SECPRI.1999.766909

[4] Kim, D.S., Nguyen, H.N., Ohn, S.Y., Park, J.S. (2005). Fusions of GA and SVM for anomaly detection in intrusion detection system. In International Symposium on Neural Networks, pp. 415-420. https://doi.org/10.1007/11427469_67

[5] Deng, H., Zeng, Q.A., Agrawal, D.P. (2003). SVM-based intrusion detection system for wireless ad hoc networks. In 2003 IEEE 58th Vehicular Technology Conference. VTC 2003-Fall (IEEE Cat. No. 03CH37484), 3: 2147-2151. https://doi.org/10.1109/VETECF.2003.1285404

[6] Bonissone, P.P. (1997). Soft computing: the convergence of emerging reasoning technologies. Soft Computing, 1(1): 6-18. https://doi.org/10.1007%2Fs005000050002

[7] Moezzi, S., Jalali, M., Forghani, Y. (2019). TWSVC+: Improved twin support vector machine-based clustering. Ingénierie des Systèmes d'Information, 24(5): 463-471. https://doi.org/10.18280/isi.240502

[8] Sani, R.A., Ghasemi, A. (2015). Learning a new distance metric to improve an SVM-clustering based intrusion detection system. In 2015 The International Symposium on Artificial Intelligence and Signal Processing (AISP), 284-289. https://doi.org/10.1109/AISP.2015.7123497

[9] Yang, Z., Karahoca, A. (2006). An anomaly intrusion detection approach using cellular neural networks. In International Symposium on Computer and Information Sciences, pp. 908-917. https://doi.org/10.1007/11902140_94

[10] Yang, Z., Karahoca, A., Yang, N., Aydin, N. (2008). Network intrusion detection by using cellular neural network with tabu search. In 2008 Bio-inspired, Learning and Intelligent Systems for Security, 64-68. https://doi.org/10.1109/BLISS.2008.29

[11] Mitchell, R., Chen, R. (2013). Adaptive intrusion detection of malicious unmanned air vehicles using behavior rule specifications. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 44(5): 593-604. https://doi.org/10.1109/TSMC.2013.2265083

[12] Hu, W., Gao, J., Wang, Y., Wu, O., Maybank, S. (2013). Online adaboost-based parameterized methods for dynamic distributed network intrusion detection. IEEE Transactions on Cybernetics, 44(1): 66-82. https://doi.org/10.1109/TCYB.2013.2247592

[13] Fung, C.J., Zhang, J., Boutaba, R. (2012). Effective acquaintance management based on bayesian learning for distributed intrusion detection networks. IEEE Transactions on Network and Service Management, 9(3): 320-332. https://doi.org/10.1109/TNSM.2012.051712.110124

[14] Aslahi-Shahri, B.M., Rahmani, R., Chizari, M., Maralani, A., Eslami, M., Golkar, M.J., Ebrahimi, A. (2016). A hybrid method consisting of GA and SVM for intrusion detection system. Neural Computing and Applications, 27(6): 1669-1676. https://doi.org/10.1007/s00521-015-1964-2

[15] Manzoor, I., Kumar, N. (2017). A feature reduced intrusion detection system using ANN classifier. Expert Systems with Applications, 88: 249-257. https://doi.org/10.1016/j.eswa.2017.07.005

[16] Cannady, J. (1998). Artificial neural networks for misuse detection. In National Information Systems Security Conference, 26: 443-456.

[17] Ryan, J., Lin, M.J., Miikkulainen, R. (1998). Intrusion detection with neural networks. In Advances in Neural Information Processing Systems, 943-949.

[18] Ke, G., Hong, Y.H. (2014). The research of network intrusion detection technology based on genetic algorithm and bp neural network. In Applied Mechanics and Materials, 599: 726-730. https://doi.org/10.4028/www.scientific.net/AMM.599-

601.726

[19] Rumelhart, D.E., Hinton, G., Williams, R. (1986). Parallel distributed processing: Explorations in the microstructure of cognition. vol. 1.

[20] Sarle, W. (1997). Neural network FAQ, part 1 of 7: Introduction, periodic posting to the usenet newsgroup comp. ai. neuralnets. ftp://ftp. sas. com/pub/neural/FAQ. html.

[21] Das, K.J. (2000). Attack development for intrusion detector evaluation. Doctoral dissertation, Massachusetts Institute of Technology.

[22] Haykin, S. (2010). Neural Networks: A Comprehensive Foundation. 1999. Mc Millan, New Jersey, 1-24.

[23] Lawrence, S., Giles, C.L. (2000). Overfitting and neural networks: Conjugate gradient and backpropagation. In Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium, 1: 114-119. https://doi.org/10.1109/IJCNN.2000.857823