THE PRACTICE ON USING MACHINE LEARNING FOR NETWORK ANOMALY INTRUSION DETECTION

YU-XIN MENG

Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong
E-MAIL: ymeng8@student.cityu.edu.hk

Abstract:

Machine learning is regarded as an effective tool utilized by intrusion detection system (IDS) to detect abnormal activities from network traffic. In particular, neural networks, support vector machines (SVM) and decision trees are three significant and popular schemes borrowed from the machine learning community into intrusion detection in recent academic research. However, these machine learning schemes are rarely employed in large-scale practical settings. In this paper, we implement and compare machine learning schemes of neural networks, SVM and decision trees in a uniform environment with the purpose of exploring the practice and issues of using these approaches in detecting abnormal behaviors. With the analysis of experimental results, we claim that the real performance of machine learning algorithms depends heavily on practical context. Therefore, the machine learning approaches are supposed to be applied in an appropriate way in terms of the actual settings.

Keywords:

Machine learning; Neural network; Support vector machine; Decision tree; Intrusion detection

1. Introduction

According to the Microsoft security intelligence report [1] from January through June 2010, the infection trends are still increasing on average around the world. The threats such as malware software (e.g. Trojans, worms) and potentially unwanted software are subsistent key challenges with regard to the security of computers and networks. To mitigate this problem, intrusion detection systems (IDS) [4, 5] have been widely deployed into network environment aim to identify unwanted software and services.

In traditional, intrusion detection systems fall into two general types: host IDS (HIDS) and network IDS (NIDS) according to the data source. The HIDS is used to log and analyze system events within a single host with the purpose of discovering abnormal executions, while the function of NIDS is monitoring and analyzing network traffic in order to identify suspicious activities. Furthermore, network intrusion detection system (NIDS) can be broadly classified as *misuse detection* and *anomaly detection* based on the use of detection techniques. The misuse detection (also called signature-based detection or rule-based detection) [14, 15] detects attacks depending heavily on the pre-defined rules that give precise descriptions of known exploits and attacks. On the contrary, anomaly detection system [16, 17] begins with establishing a normal behavior model and then finds out deviations from that profile, which is expected to have the ability to discover unknown attacks.

The basic assumption for the anomaly detection system is that malicious activity reveals characteristics not observed for normal usage [6]. In other words, the basic problem in anomaly detection is an issue of classification that how to differentiate normal and abnormal activities effectively and efficiently. To settle the problem, machine learning schemes are extensively applied and developed in current intrusion detection community to improve the performance of anomaly detection. In particular, neural networks [8], support vector machines [7], decision trees [10] are popular and significant schemes utilized in anomaly detection systems to enhance the classification performance and speed. In addition, many other machine learning approaches (e.g. genetic algorithms [9], information theory [18]) are used to boost the classification process and consolidate the theory base.

Neural networks often refer to artificial neural networks (ANN) in computing science that consist of interconnecting artificial neurons that simulate the properties of biological neurons. The most important and crucial property of ANN is to learn and adjust coefficients automatically in the neural networks according to data inputs and outputs. In the training phase, the types of neural network schemes can be classified as *supervised training algorithms* and *unsupervised training algorithms*. Supervised training networks learn the desired data output for a given input, while the unsupervised training networks learn unlabeled output (without explicating desired output). After training phase, ANN can establish connections between input and output nodes with coefficient or weight to be used to determine new input data during the test phase.

Support vector machines (SVM) are a set of related supervised learning methods that utilized to analyze data and recognize patterns by mapping input feature vectors into a higher dimensional feature space. The standard SVM is a non probabilistic binary linear classifier that builds a model by given a set of marked training data that used to category the new input data. Intuitively, a good separation is achieved by the hyperplane (high dimensional space) that has the largest distance to the nearest training data points with the purpose of reducing generalization error.

Decision tree is a tree-like model to classify the given datasets using the values of its attributes. A node of a decision tree shows an attribute and its relevant edges. Each edge connects two nodes or a node and a leaf. A leaf is explicated with a decision value to determine the class of the input data. In intrusion detection, the main issue of decision tree is that how to select the best attributes in partitioning the datasets with effectiveness. A decision tree can be constructed by pre-classified data in training phase, and then used to classify new input data in terms of decision values during test phase.

Though the machine learning schemes are ad hoc in the academic research of anomaly intrusion detection, it is rarely employed in practical settings (e.g. large scale, operational environment) [19]. We argue that the false alarm rate, high training cost and fluctuant capability are the reasons for such situation. In this paper, we construct a unified experimental environment to compare the machine learning approaches of neural networks, support vector machines and decision trees and analyze their performance results with the purpose of exploring the practical issues on using these machine learning schemes in network anomaly detection systems.

The rest of the paper is organized as follows. In section 2, we begin with a brief discussion of KDD99 datasets and feature selection. We then make comparisons by using neural networks, support vector machines and decision trees plus an analysis of the experimental results in section 3. At last, we briefly summarize our work in section 4.

2. Datasets and feature selection

To provide a uniform environment for the experiment, we use an open source tool: WEKA (Waikato Environment for Knowledge Analysis) version 3.6 [22] as the experimental platform to realize machine learning schemes so as to avoid the effects on the experimental results that may be caused by the distinct implementations of algorithms.

2.1. KDD dataset

KDD Cup 1999 Data (KDD99) [11] is the dataset used in the experiment to evaluate machine learning schemes. This dataset derives from the DARPA packet traces and includes a wide variety of intrusions simulated in a military network environment. In practice, we acknowledge that this dataset is decade old and has many criticisms [12, 13] for current research. But we believe that it is still adequate for our experiment which aims to reflect the performance of distinct machine learning approaches in a general way and find out relevant issues.

In addition, the full KDD99 dataset (uncompress the file *kddcup.data.gz* [11]) has 4,898,431 records and each record contains 41 features. Due to the computing power, we do not use the full dataset of KDD99 in the experiment but a 10% portion (uncompress the file *kddcup.data_10_percent.gz*) of it. This 10% KDD99 dataset contains 494,069 records (each with 41 features) and 4 categories of attacks. The details of attack categories and specific types are shown in Table1.

Table 1. Attack categories and types in 10% kdd99 dataset

number	Four Attack categories			
	Probe	*DoS*	*U2R*	*R2L*
# 1	ipsweep	back	buffer overflow	ftp write
# 2	nmap	land	loadmodule	guess passwd
#3	portsweep	neptune	perl	imap
#4	satan	pod	rootkit	multihop
#5	-	smurf	-	phf
#6	-	teardrop	-	spy
#7	-	-	-	warezclient
#8	-	-	-	warezmaster

According to Table1, there are four attack categories in 10% KDD99 dataset (also in full KDD99 dataset):

(1) *Probing:* scan networks to gather deeper information

(2) *DoS:* denial of service

(3) *U2R:* illegal access to gain super user privileges

(4) *R2L:* illegal access from a remote machine

Each of the attack categories contains some specific attack types. For example, *DoS* has 6 specific attack types (e.g. back, land, neptune), *R2L* has 8 specific attack types (e.g ftp write, guess passwd, imap). There are totally 22 specific attack types within the 10% KDD99 dataset, while the full KDD99 dataset has 39 specific attack types. Although the number of specific attack types is different between 10% KDD99 dataset and full KDD99 dataset, we believe that there are no negative effects on our evaluation purpose.

2.2. Feature selection

The attribute (also called *feature*) in the dataset is a key element that can affect the performance results of machine learning schemes. There are 41 features of each record in the dataset which belong to 4 main categories: TCP connection basic characteristic, TCP connection content characteristic, time-based network traffic and host-based network traffic. The full features are shown in Table2. Intuitively, some features are insignificant to the training of machine learning algorithms as well the improvement of detection rate. Thus, feature selection [24] is an essential method to select a subset of relevant features for building robust learning models.

Table 2. Feature categories in kdd99 dataset

Four categories of features	
TCP connection basic characteristic (1~9)	*duration*, *protocol_type*, *service*, *flag*, *src_bytes*, *dst_bytes*, *land*, *wrong_fragment*, *urgent*
TCP connection content characteristic (10~22)	*hot*, *num_failed_logins*, *logged_in*, *num_compromised*, *root_shell*, *su_attempted*, *num_root*, *num_file_creations*, *num_shells*, *num_access_files*, *num_outbound_cmds*, *is_hot_login*, *is_guest_login*
Time-based network traffic (23~31)	*count*, *srv_count*, *serror_rate*, *srv_seror_rate*, *rerror_rate*, *srv_rerror_rate*, *same_srv_rate*, *diff_srv_rate*, *srv_diff_host_rate*
Host-based network traffic (32~41)	*dst_host_count*, *dst_host_srv_count*, *dst_host_same_srv_rate*, *dst_host_diff_srv_rate*, *dst_host_same_src_port_rate*, *dst_host_srv_diff_host_rate*, *dst_host_serror_rate*, *dst_host_srv_serror_rate*, *dst_host_srv_serror_rate*, *dst_host_rerror_rate*, *dst_host_srv_rerror_rate*

In table2, we give a number to each feature according to their sequences. For instance, *duration* is 1, *protocol_type* is 2 and *hot* is 10. Moreover, intending to provide a wider and deeper comparison result in the experiment, we choose three attribute evaluation methods and four search methods from WEKA 3.6 for feature selection.

Evaluation methods:

(1) *CfsSubsetEval:* evaluates the worth of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them.

(2) *InfoGainAttributeEval:* evaluates the worth of an attribute by judging the information gain with respect to the class.

(3) *GainRatioAttributeEval:* evaluates the worth of an attribute by judging the gain ratio with respect to the class.

Search methods:

(1) *BestFirst:* searches the space of attribute subsets by greedy hillclimbing augmented with a backtracking facility.

(2) *GreedyStepwise:* performs a greedy forward or back- ward search through the space of attribute subsets.

(3) *Ranker:* ranks attributes through their individual evaluations.

(4) *GeneticSearch:* performs a search using the simple genetic algorithm described in Goldberg.

What is more, in order to illustrate the effects caused by the numbers of values in the classification attribute (the label attribute), we begin with a feature selection with a 23-class classifier (corresponding to 22 specific attack types plus a normal type in 10% dataset) in Table3. Then, we aggregate the 22 specific attack types into 4 categories (Probe, DOS, U2R and R2L) so as to form a 5-class (plus a normal type) classifier and the results of relevant feature selection are shown in Table4.

Table 3. feature selection results with 23-class classifier

Search methods	Evaluation methods		
	CfsSubsetEval	*InfoGainAttributeEval*	*GainRatioAttributeEval*
BestFirst	2,3,4,5,6,7,8,14,23,30,36	-	-
GreedyStepwis	2,3,4,5,6,7,8,14,23,30,36	-	-
GeneticSearch	2,3,4,5,6,7,8,10,12,19,23,29,30,31,33,36,38	-	-
Ranker	-	5,23,3,24,36,2,33,35,34,30,29,4,6,38,25,39,26,12,32,37,31,40,41,27,28,1,10,13,8,22,16,19,17,11,14,7,18,9,15,21,20	8,7,13,2,11,4,10,26,25,12,3,30,39,38,36,9,5,29,14,6,35,34,33,23,22,37,24,32,40,27,41,31,28,18,1,17,16,19,15,21,20

According to the results in Table3 and Table4, it is true that the class numbers in classification attribute will affect the final results of feature subset selection. For example, there are 11 features {2,3,4,5,6,7,8,14,23,30,36} selected under the combination of *CfsSubsetEval* and *BestFirst* in Table3. While the number reduces to 8 features {3,6,12,14,23,31,32,37} in Table4 with the same combination. Furthermore, the ranking sequences are greatly different by combining *InfoGainAttr- ibuteEval* and *GainRatioAttributeEval* with *Ranker* (note that the search method of *Ranker* cannot be used with *CfsSubsetEval*). But it is not often true that the less value numbers in the classifier, the less numbers in the feature subsets. We provide an example in Table5, which the classifier has only 2 class (normal type and attack type). With the combination of *GeneticSearch* and *CfsSubsetEval*, the number of features in this subset is more than that in Table4. (*All the evaluation and search methods are used with default settings in WEKA.*)

Table 4. Feature selection results with 5-class classifier

Search methods	Evaluation methods		
	CfsSubsetEval	*InfoGainAttributeEval*	*GainRatioAttributeEval*
BestFirst	3,6,12,14,23, 31,32,37	-	-
GreedyStepwis	3,6,12,14,31, 32	-	-
GeneticSearch	2,5,6,11,12,22 ,23,30,31,37	-	-
Ranker	-	5,23,3,6,12,24,36, 32,2,37,33,35,34,31 ,30,29,38,39,25,4, 26,1,40,41,27,28,10 ,22,16,19,13,17,11, 8,14,18,9,15,7,21, 20	12,11,6,14,22,9,37, 3,32,31,5,2,1,17,23, 36,16,18,19,10,15, 24,38,35,25,33,39, 34,30,26,4,41,40,29 ,27,13,28,8,7,21,20

Table 5. Feature selection results with 2-class classifier

Search methods	Evaluation methods		
	CfsSubsetEval	*InfoGainAttributeEval*	*GainRatioAttributeEval*
BestFirst	6,12,23,31,32	-	-
GreedyStepwis	6,12,23,31,32	-	-
GeneticSearch	1,2,6,7,8,12, 15,23,31,32, 36,37	-	-
Ranker	-	5,23,3,6,36,12,24,2, 32,37,33,35,31,34, 29,30,39,38,26,25,4 ,1,41,40,10,16,28, 19,13,17,8,22,18,15 ,11,14,7,21,27,9,20	12,6,37,32,31,3,2,5, 36,23,1,19,16,17,24 ,15,18,39,26,34,38, 35,33,25,30,29,4,41 ,10,22,8,13,40,11, 14,7,28,21,27,9,20

3. Experiment on machine learning schemes

In this section, we perform schemes of neural networks, support vector machines and decision trees that have been realized in WEKA environment and applied them into distinct feature subsets to provide multi-angular comparison results. In addition, the computing power of the experimental envir- onment is based on the capability of Intel Core(TM) 2 Duo CPU with 2.8GHz processor plus 4.0 G RAM in a 32-bit operating system.

3.1 scheme selection and experimental results

Each kind of schemes of neural networks, support vector machines and decision trees has lots of specific algorithms. In the experiment, we only choose three algorithms provided by WEKA to make the comparisons. The selected algorithms are as follows.

RBFNetwork (Neural network): implement a normalized Gaussian radial basis function network.

SMO (support vector machine): implement sequential minimal optimization algorithm for training a support vector classifier. This SVM scheme has the ability to deal with multi classification problem.

J48 (decision tree): generate a pruned or unpruned C4.5 decision tree.

In the experiment, we begin with evaluating the perfor- mance of RBFnetwork, SMO and J48 on a dataset of 98,804 instances (consist of normal: 19268, DoS: 79207, U2R: 17, R2L: 1 and Probe: 311) that are randomly selected from the 10% KDD99 dataset in order to reduce the burdens of computing. The results are illustrated in Table6 and Table7 with a 2-class classifier and a 5-class classifier respectively. With regard to the combination of *InfoGainAttributeEval*, *GainRatioAttributeEval* with *Ranker*, we choose the top12 features to form the subset respectively. (*All the algorithms are used with default settings in WEKA.*)

As the J48 has the best performance of detection rate in Table6 and Table7. We further implement it to 10% KDD99 dataset with the same feature subsets in Table8.

Furthermore, we compare the detection rates of Probe, DoS, U2R and R2L on 10% KDD99 dataset in Table9 plus an additional condition of *no feature selection*.

Table 6. Results on 2-class dataset (98804 instances)

10-folder Cross validation	RBFnetwork	SMO	J48
	Accuracy(%)	*Accuracy(%)*	*Accuracy(%)*
BestFirst+CfssubsetEval	98.3897	99.2602	99.582
GeneticSearch+ CfssubsetEval	97.3048	99.6326	99.9231
Ranker+InfoGainAttributeEval	99.244	99.6731	99.9777
Ranker+GainRatioAttributeEval	95.0194	99.579	99.9757

Table 7. Results on 5-class dataset (98804 instances)

10-folder Cross validation	RBFnetwork	SMO	J48
	Accuracy(%)	*Accuracy(%)*	*Accuracy(%)*

10-folder Cross validation	RBFnetwork	SMO	J48
	Accuracy(%)	*Accuracy(%)*	*Accuracy(%)*
BestFirst+CfssubsetEval	99.3998	99.5385	99.7561
GeneticSearch+CfssubsetEval	99.3857	99.3209	99.9312
Ranker+InfoGainAttributeEval	99.3968	99.7733	99.9767
Ranker+GainRatioAttributeEval	94.0994	99.5193	99.8239

Table 8. J48 results on 10% kdd99 dataset (494021 instances)

10-folder Cross validation	J48		
	23-class Accuracy(%)	*5-class Accuracy(%)*	*2-class Accuracy(%)*
BestFirst+CfssubsetEval	99.9439	99.6371	99.4087
GeneticSearch+CfssubsetEval	99.931	99.8935	99.8994
Ranker+InfoGainAttributeEval	99.9387	99.9597	99.969
Ranker+GainRatioAttributeEval	99.6298	99.73	99.9609

Table 9. J48 detection rates on 10% kdd99 dataset with 5-class classifier

10-folder Cross validation	J48 Detection rate (%)			
	Probe	*DoS*	*U2R*	*R2L*
No feature selection	99.34	99.99	53.85	96.43
BestFirst+CfssubsetEval	67.2	99.95	67.27	92.28
GeneticSearch+CfssubsetEval	99.1	99.97	32.69	96.36
Ranker+InfoGainAttributeEval	99.2	99.95	53.85	96.45
Ranker+GainRatioAttributeEval	74.39	99.97	63.46	95.83

3.2 Analysis and discussion

In our initial experiments (see Table6 and Table7), the performance of J48 is better than the other two algorithms over all feature subsets and all the three algorithms achieve a very high detection rate more than 94% in general. We then implement the J48 algorithm to 10% KDD99 dataset aim to compare the results with different classifiers (see Table8). Intuitively, the results are improved with the increase of class (from 5-class to 23-class) for the evaluation method of *Cfs- subsetEval*, but are decreased with the other two methods of *InfoGainAttributeEval* and *GainRatioAttributeEval*. Finally, a comparison of detection rate of 4 attack categories is given in Table9. The detection rates for DoS and R2L are high and consistent, while are not stable for Probe and U2R.

The results in the experiment reflect some challenges on using machine learning schemes in intrusion detection.

Fluctuant capability: the detection performance of J48, SMO and RBFnetwork is high in total, but the capability of such rates are not stable and fluctuant with different feature subsets, as well with the distinct classifiers. For example, the detection rates of U2R (from 32.69% to 67.27%) and Probe (from 67.2% to99.34%) in Table9 are changed a lot with different methods of feature selection (subsets are distinct). Moreover, the real results of feature selection are not always better than those of no feature selection. In Table9, the detection rate of Probe attack is up to 99.34% without selecting features, yet the rate is only 67.2% with the feature selection of BestFirst and CfssubsetEval.

False alarm rate (also called false positive rate): due to the fluctuant capability of machine learning schemes, the false alarm rates are fluctuant as well. Though the false alarm rates in our experiment are acceptable, it is not the actual case in the real environment. The cost of errors in large-scale operational settings may very high since the analysis time and man power are so expensive that if a benign activity is reported as an intrusion. What is more, the excessive false positive reports may be generated under an environment with large network traffic even with a low false alarm rate.

Difficulties of Training: intuitively, the more and larger training datasets, the more precise detection for the anomalies. However, the dataset needs to be properly labeled (to decide instances to be normal or abnormal) in the training phase. Unfortunately, it is very hard to achieve this goal in practical environment because of the enormous network traffic and expensive man power. In addition, the time for the training is distinct with different algorithms. In our experiment, the large dataset asks for more training time such as neural networks and SVM than decision trees.

Discussion: The main purpose of our work is to explore the practice on using machine learning schemes in anomaly intrusion detection. We acknowledge that our experimental results are only a patch of applications of machine learning schemes due to page limit, but the results do actually reflect some key issues. What is more, some other research results [2, 20, 21, 23] are complementary to our efforts. In addition, a better detection performance can be achieved by the use of ensemble method of utilizing the fusion of different base classifiers [3]. Through the analysis results in our work, we express the key idea that the machine learning schemes are supposed to be implemented by understanding the system targets and as well trading off the balance between the costs and accuracy according to real scenarios.

4. Conclusions

In this paper, we perform a wider and deeper experiment to compare the performance of machine learning schemes as neural networks, support vector machines and decision trees with the purpose of showing the practice and exploring the issues on using such kind of approaches in network anomaly intrusion detection. Moreover, the effects of feature selection are discussed and analyzed as well. Through the experimental results, we claim that the machine learning approaches are fit to identify anomalies by proper training but the performance may be variable in terms of different algorithms. In particular, the fluctuant capability, the cost of false alarm rate and the difficulties of training data are key factors and challenges to encumber the wide use of machine learning schemes in real operational environment. Thus, the machine learning schemes should be applied in an appropriate way to enhance detection performance.

References

[1] Microsoft Security Intelligence Report 2010, www.microsoft.com/sir (access on April 2011).

[2] Mukkamala. S., Janoski. G., Sung. A., “Intrusion detection: support vector machines and neural networks”, Proceedings of the IEEE International Joint Conference on Neural Networks, pp. 1702–1707. St. Louis, MO, 2002.

[3] Chan, A.P.F., Ng, W.W.Y., Yeung, D.S., Tsang, E.C.C., "Comparison of different fusion approaches for network intrusion detection using ensemble of RBFNN", Proceedings of IEEE International Conference on Machine Learning and Cybernetics, pp.3846-3851 Vol. 6, 18-21 Aug. 2005.

[4] Paxson. V., “Bro: A System for Detecting Network Intruders in Real-Time”, Computer Networks 31(23-24), 2435–2463, 1999.

[5] K. Scarfone, P. Mell, “Guide to Intrusion Detection and Prevention Systems (IDPS)”, NIST Special Publication 800-94, Feb 2007.

[6] D. E. Denning, “An Intrusion-Detection Model”, IEEE Transactions on Software Engineering, vol. 13, no.2, pp. 222-232,1987.

[7] W. Hu, Y. Liao, and V. R. Vemuri, “Robust Anomaly Detection Using Support Vector Machines,” Proceedings of International Conference on Machine Learning, 2003.

[8] Z. Zhang, J. Li, C. Manikopoulos, J. Jorgenson, and J. Ucles, “HIDE: a Hierarchical Network Intrusion Detection System Using Statistical Preprocessing and Neural Network Classification,” IEEE Workshop on Information Assurance and Security, 2001.

[9] C. Sinclair, L. Pierce, and S. Matzner, “An Application of Machine Learning to Network Intrusion Detection,” Proceedings of Computer Security Applications Conference, 1999.

[10] S. R. Safavian, and D. Landgrebe, “A survey of decision tree classifier methodology”, IEEE Transactions on Systems, Man and Cybernetics, vol. 21, pp. 660-674, June 1991.

[11] “The UCI KDD Archive: KDD Cup 1999 Data Set,” http://archive.ics.uci.edu/ml/datasets/KDD+Cup+1999+Data .

[12] J. McHugh, “Testing Intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratories”, ACM Transactions on Information and System Security, vol. 3, no. 4, pp. 262–294, November 2000.

[13] M. V. Mahoney and P. K. Chan, “An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection”, Proceedings of Recent Advances in Intrusion Detection, 2003.

[14] G. Vigna, R.A. Kemmerer, “NetSTAT: a network-based Intrusion Detection Approach”, Proceedings of Annual Computer Security Applications Conference, pp. 25 –34, 1998.

[15] W. Lee, S. J. Stolfo, K. Mok, “A Data Mining Framework for Building Intrusion Detection Models”, Proceedings of IEEE Symposium of Security and Privacy, pp. 120-132, 1999.

[16] A. Valdes, D. Anderson, “Statistical Methods for Computer Usage Anomaly Detection Using NIDES”, Technical report, SRI International, January 1995.

[17] A.K. Ghosh, J. Wanken, F. Charron, “Detecting Anomalous and Unknown Intrusions Against Programs”, Proceedings of IEEE Annual Computer Security Applications Conference, pp. 259 – 267, 1998.

[18] W. Lee and D. Xiang, “Information-Theoretic Measures for Anomaly Detection,” Proceedings of IEEE Symposium on Security and Privacy, 2001.

[19] Sommer, R., Paxson, V., “Outside the closed world: On using machine learning for network intrusion detection”, Proceedings of IEEE Symposium on Security and Privacy, 2010.

[20] Peddabachigari S., Abraham A., Thomas J., “Intrusion Detection Systems Using Decision Trees and Support Vector Machines”, International Journal of Applied Science and Computations, Vol.11, No.3, pp.118-134, 2004.

[21] Dong Ling Tong and Robert Mintram, “Genetic Algorithm-Neural Network (GANN): a study of neural network activation functions and depth of genetic algorithm search applied to feature selection”, International Journal of Machine Learning and Cybernetics, Vol. 1, No. 1-4, pp. 75-87, 2010..

[22] WEKA, http://www.cs.waikato.ac.nz/ml/weka/.

[23] Mukkamala. S. and Sung. A. H., “Feature Selection for Intrusion Detection Using Neural Networks and Support Vector Machines”, Journal of the Transportation Research Board of the National Academics, pp. 33-39, 2003