Classification Trees with Logistic Regression Functions for Network Based Intrusion Detection System

Article *in* IOSR Journal of Computer Engineering · June 2017

DOI: 10.9790/0661-1903044852

1 author:

Deeman Yousif Mahmood
Hasselt University
17 PUBLICATIONS **38** CITATIONS

SEE PROFILE

IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 19, Issue 3, Ver. IV (May - June 2017), PP 48-52
www.iosrjournals.org

Classification Trees with Logistic Regression Functions for Network Based Intrusion Detection System

Deeman Yousif Mahmood
(Computer Science Dept., College of Science/ University of Kirkuk, Iraq)

Abstract: *Intrusion Detection Systems considered as an indispensable field of network security to detect passive and anomaly activities in network traffics and packets. In this paper a framework of network based intrusion detection system has been implemented using Logistic Model Trees supervised machine learning algorithm."NSL-KDD" dataset which is an updated dataset from "KDDCup 1999" benchmark dataset for intrusion detection has been used for the experimental analysis using percent of 60% for training phase and the rest for testing phase. The testing and experimental results from the proposed structure shows that using two way functions which are classification with regression combined in Logistic Model Tree is very accurate in term of accuracy and minimum false-positive average with high true-positive average. Two classifications has been performed in the proposed model which are (Attack or Normal)*

Keywords: *Intrusion Detection System (IDS), Logistic Model Trees (LMT), Logistic Regression, NSL-KDD*

I. Introduction

An Intrusion Detection System (IDS) is a system that monitors and analyses network traffics to check for intrusive activities in the network and report events that does not coordinate the security criteria of the system administrator and within the recent years, attacks has been increased rapidly and drastically on networks and web applications which results in a wide interest of researchers for network intrusion detection systems [1]. IDSs are came into two categories: Signature based and Anomaly based, where the signature based seeks for previously-knowing patterns or samples of attacks, this model can detect and recognize only an attack with a precise matching behavior found versus previously stored patterns or samples knowing by signatures, while anomaly based is build based on origination a profile for normal activity of the system, anomaly based technique promotes itself by comprehension and aggregating information and familiarity about the system and decides the conduct of the security system in view of it [2]. Two essential types of IDS exists: host-based (HIDS) with network-based (NIDS), where the HIDS dwells on a specific host and searches for attack signs on that host, while NIDS dwells on a separate and distributed system that searches for network traffics, looking for attacks signs that cross that segment of the network [3]. To summarize, this paper presents firstly the intrusion detection system and the data set used for this research which is NSL-KDD data set, and on this basis, the proposition of this paper is to design a hypothesis approach for designing an accurate model for IDS in term of high detection rates and performance with keeping a low false alarm rate by using Logistic Model Trees (LMT).

II. Dataset

To evaluate the adequacy of the proposed IDS model, the "NSL-KDD" dataset has been utilized which is a modern issuance of "KDDCup99" dataset that is deemed as a basic benchmarking for evaluating intrusion detection rates. The dataset comprises of roughly 4,900,000 connection vector and each single vector consist of 41 features and class feature labeled to either normal or attack [4].

The attack types in this dataset come into four main classes:

- DOS: "Denial-of-Service".
- R2L: "Accessing from unauthorized remote machine".
- U2R: "Unauthorized access to privileges of root user".
- Probing: "Surveillance and other probing".

These features or attributes of each TCP/IP connection including target class appeared in Table 1 (KDD-CUP99 Features Description [5]).

Table 1: "NSL-KDDCup99" Dataset Features

#	Feature name	Description	Type Discrete Continuous
1	"Duration"	"Length (# of seconds) of the connection"	C
2	"protocol type"	"Type of the protocol, e.g. tcp, udp, etc."	D
3	"Service"	"Network service on the destination, e.g., http, telnet, etc."	D
4	"Flag"	"Normal or error status of the connection"	D
5	"src_bytes"	"# of data bytes from source to destination"	C
6	"dst_bytes"	"# of data bytes from destination to source"	C
7	"Land"	"1 if connection is from/to the same host/port; 0 otherwise"	D
8	"wrong_fragment"	"# of wrong fragments"	C
9	"Urgent"	"# of urgent packets"	C
10	"Hot"	"# of hot indicators"	C
11	"num_failed_logins"	"# of failed login attempts"	C
12	"logged in"	"1 if successfully logged in; 0 otherwise"	D
13	"num_compromised"	"# of compromised conditions"	C
14	"root_shell"	"1 if root shell is obtained; 0 otherwise"	D
15	"su_attempted"	"1 if su root command attempted; 0 otherwise"	D
16	"num_root"	"# of root accesses"	C
17	"num_file_creations"	"# of file creation operations"	C
18	"num_shells"	"# of shell prompts"	C
19	"num_access_files"	"# of operations on access control files"	C
20	"num_outbound_cmds"	"# of outbound commands in an ftp session"	C
21	"is_host_login"	"1 if the login belongs to the hot list; 0 otherwise"	D
22	"is_guest_login"	"1 if the login is a guest login; 0 otherwise"	D
23	"Count"	"# connections to the same host as the current one during past two seconds"	C
24	"srv_count"	"# of connections to the same service as the current connection in the past two seconds"	C
25	"serror_rate"	"% of connections that have SYN errors"	C
26	"srv_serror_rate"	"% of connections that have SYN errors"	C
27	"rerror_rate"	"% of connections that have REJ errors"	C
28	"srv_rerror_rate"	"% of connections that have REJ errors"	C
29	"same_srv_rate"	"% of connections to the same service"	C
30	"diff_srv_rate"	"% of connections to different services"	C
31	"srv_diff_host_rate"	"% of connections to different hosts"	C
32	"dst_host_count"		C
33	"dst_host_srv_count"		C
34	"dst_host_same_srv_rate"		C
35	"dst_host_diff_srv_rate"		C
36	"dst_host_same_src_port_rate"		C
37	"dst_host_srv_diff_host_rate"		C
38	"dst_host_serror_rate"		C
39	"dst_host_srv_serror_rate"		C
40	"dst_host_rerror_rate"		C
41	"dst_host_srv_rerror_rate"		C

III. Methodology

The intrusion detection model used in this research is consist of five parts: selection of dataset for training and testing phase, pre-processing of dataset, classification method, training and testing the system. The different phases of the implemented methodology is shown in the Figure 1.

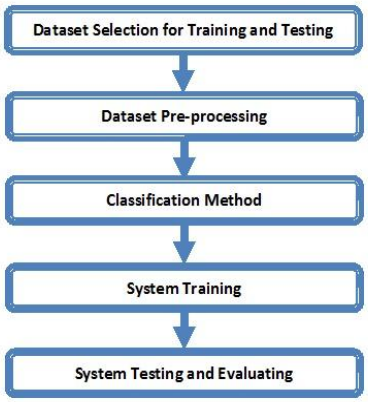

Figure 1: Phases of Implemented Methodology

3.1 Dataset Selection and Pre-Processing

"NSL-KDD" data set has been used as a benchmark dataset for training and testing the proposed system and the first part of analysis engine component of the IDS model is the pre-processing dataset. The pre-processing of dataset has a great importance as it results in increasing the efficiency of the detection for cases of training, testing, and classifying network activity into normal and abnormal. Pre-processing over the original "NSL-KDD" dataset is a substantial step to make the dataset suitable and appropriate for the IDS structure. Pre-processing of the dataset can be fulfilled by implementing:

- Nominal Features Transformation.
- Numeric Features Normalization.

3.1.1 Dataset Transformation

The "NSL-KDD" dataset consists of a huge amount of connection instances, while each connection pattern has 42 features with target class "Attack" or "Normal". These labelled connection patterns have to be converted and transformed from nominal or symbolic features in to numerical corresponding values to be in an appropriate input form for the model. An example: several text-based features required to be turned into numerical corresponding values, for this transformation: Table 2 will be used, and in this phase some vain data will be filtrated and adjusted.

There are sundry nominal patterns such as "HTTP", "TCP" and "SF", so that it is necessary to transform these nominal features to numeric corresponding values beforehand. For example, service type of "tcp is mapped to 1", "udp is mapped to 2" and "icmp is mapped to 3". Hence, keys in Table 2 will be used to convert and transform the nominal patterns of the dataset into the numeric corresponding values.

Table 2: Transformation Table for Different Values of Protocols, Flag and Services

Flag	"OTH"	1
	"REJ"	2
	"RSTO"	3
	"RSTOS0"	4
	"RSTR"	5
	"S0"	6
	"S1"	7
	"S2"	8
	"S3"	9
	"SF"	10
	"SH"	11
Protocol Type	"TCP"	1
	"UDP"	2
	"ICMP"	3
Service	All services	1 to 70

3.1.2 Dataset Normalization

Normalization is a fundamental step of dataset pre-processing which is enhancing the IDS performance when the used datasets for both training and testing are too large. The first step is to normalize continuous attributes, so that attribute values fall truly within a specified range of 0 to 1. In this model, Min-Max method of normalization has been used, using the following equation [6]:

$$x_i = \frac{v_i - \min(v_i)}{\max(v_i) - \min(v_i)}$$

Where, x_i is the normalized value, v_i is the factual value of that attribute where the maximum with minimum values are taken including all the values of that attribute and usually x_i is mapped to zero if the value of maximum record of the attribute is equal to the minimum one. An example of "NSL-KDD" dataset record is shown in Table 3.

Table 3: NSL-KDD Dataset Records

Original NSL-KDD Dataset Record
"0 Tcp http SF 181 5450 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 8 8 0 0 0 0 1 0 0 9 9 1 0 0.11 0 0 0 0 0"
Transformed and Normalized NSL-KDD Dataset Record
"0 1 33 10 0.28 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0.14 0 1 0 0.03 0 0 0 0 0"

3.2 Classification Method

Classification task is an important part of the constructed model for detecting intrusions, the classification model that has been used for this model is Logistic Model Trees (LMT).

This classifier is a product of combining the tree structure and the function of logistic regression to construct a single decision tree that has the function of logistic regression in the leaves [7], that is providing a model of piecewise linear regression which is a real valued function [8], while the ordinary decision trees that have constants at their leaves produced the piecewise constant model which is a function that has the same output vale for each input value [9]. The LogitBoost which is a boosting algorithm used to product the Logistic Regression model for every node of the constructed tree; the node is then splitting by the C4.5 criterion. The basic Logistic Model Trees induction algorithm uses the cross validation for finding the number of iterations of the LogitBoost that doesn't over fit the training data [10].
Figure 2 gives the pseudo code for the Logistic Model Trees algorithm [11].

```
LMT (examples)
{
        root = new Node ()
         alpha = getCARTAlpha (examples)
        root.buildTree (examples, null)
        root.CARTprune (alpha)
}
build Tree (examples, initialLinearModels)
{
        numIterations = CV_Iterations (examples, initialLinea Models)
        initLogitBoost (initialLinearModels)
        linearModels = copyOf (initialLinearModels)
        for i = 1...numIterations
                logitBoostIteration (linearModels, examples)
        split = findSplit (examples)
        localExamples = split.splitExamples (examples)
        sons = new Nodes[split.numSubsets ()]
         for s = 1...sons.length
                sons.buildTree (localExamples[s],nodeModels)
}
CV_Iterations (examples, initialLinearModels)
{
        for fold = 1...5
                InitLogitBoost (initialLinearModels) //split into training/test
        set
                train = trainCV (fold)
                test = testCV (fold)
                linearModels = copyOf (initialLinearModels)
                for i = 1...200
                        logitBoostIteration (linearModels,train)
                        logErrors[i] += error(test)
        numIterations = findBestIteration (logErrors)
        Return numIterations
 }
```

Figure 2: Logistic Model Trees (LMT) Pseudo Code

According to the explained algorithm above, following notes used to build the model:

- LogitBoost is used for building the logistic regression model for the root node by running on all the data.
- Iterations number to be used is set by "Five Folds Cross Validation", where in every fold, the LogitBoost is running through the training set overhead to the max digit of iterations which is (200).
- Iterations numbers that yield the minimum sum of errors over the testing set through all five folds will be applied to the LogitBoost model using all the data to construct the model for the parent or root node and build the logistic regression models for all other nodes of the tree.
- The data is splitting by the "C4.5" splitting criterion.
- LogitBoost will build the Logistic regression models at the lower levels or child nodes using the identical subsets of the data being used.
- As long as at minimal "15" samples are presented at one node with the of useful splitting criteria which is specified in the "C4.5" splitting scheme [10], then the operation of splitting and constructing the model is persistence and proceed in the same manner.
- The CART (Classification and Regression Trees) cross validation based on the pruning-algorithm is applied to the constructed tree [10,12].

3.3 System Training

The training phase of the implemented model is about feeding the algorithm by the dataset to construct the classification model which is based on classification tree with logistic regression. NSL-KDD data set has been used with a split percentage (60%) as the training set and the rest for the testing set and two class (Normal, Attack) classification has been performed.

3.4 System Testing and Evaluating

The testing phase is about applying the test set over the constructed model and test the system accuracy based on some criteria used in machine learning for calculating accuracy and the performance measurement that has been used to evaluate the proposed architecture is based on the following terms: "True Positive TP" for exactly recognized, "True Negative TN" for exactly denied, "False Positive FP" for incorrectly recognized, "False Negative FN" for incorrectly denied, accuracy, and error rate [13]. These standard metrics are extracted from a data structure knowing as the "Confusion Matrix" which is a particular table layout that visualize the performance of an algorithm as shown in Table 3.

Table 3: Confusion Matrix for Two Class Classification

	Predicted Class		
Actual Class	Activity	Attack	Normal
	Attack	TP	FN
	Normal	FP	TN

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{\#Correct\ Classification}{\#All\ Instances}$$

$$Error\ rate = 1 - Accuracy$$

IV. Results

The Logistic Model Trees algorithm is the basic classification algorithm for the constructed model and the results that has been obtained from the testing phase are summarized in Table 4. Test results computed and obtained using a computer running Windows 8 Pro, 64-bit Operating System, Intel® Core™ i7-3537U CPU @ 2.00 GHz and 8 GB of RAM, using Netbeans IDE 8.2 with Java J2SE packages.

Table 4: Results of Classification Model (LMT)

Parameter	**Value**
Accuracy	99.1962 %
Error Rate	0.8038 %
Average True Positive Rate	0.992
Average False Positive Rate	0.008

V. Conclusion

In this paper, an Intrusion Detection model based on Logistic Model Trees was proposed. The LMT algorithm can ensure the high classification accuracy and improve the detection rate. The LMT supervised classification algorithm has been used to determine the important parameters of this algorithm which is the classification via regression, and based on the experiments done in this work and the corresponding results obtained from the model it could emphasized that the Logistic Model Trees can ensure that the decision tree classifier via linear regression has high classification accuracy with low error rates.

References

[1] P. Kabiri, and A. Ghorbani, Research on Intrusion Detection and Response: A Survey, *International Journal of Network Security, 1(2)*, 2005, 84-102.

[2] Gaikwad, S. Jagtap, K. Thakare, and V. Budhawant, Anomaly Based Intrusion Detection System Using Artificial Neural Network and Fuzzy Clustering, *International Journal of Engineering Research & Technology (IJERT), 1(9)*, 2012,

[3] S. Sonawane, Sh. Pardeshi, and G. Prasad, A Survey on Intrusion Detection, *World Journal of Science and Technology, 2(3)*, 2012,

[4] Ch. Gu, and X. Zhang, A Rough Set and SVM Based Intrusion Detection Classifier, *Second International Workshop on Computer Science and Engineering, 18(2)*,2009.

[5] KDDCup99 Dataset Features, http://kdd.ics.uci.edu/databases/kddcup99/task.htm

[6] S. Knapskog, S. Gombault, and W. Wang, Attribute Normalization in Network Intrusion Detection, *IEEE 10th International Symposium on Pervasive Systems, Algorithms, and Networks (ISPAN), DOI: 10.1109/I-SPAN.2009.49*, 2009, 448-453.

[7] V. Mahesh, A. Kandaswamy, C. Vimal, and B. Sathish, ECG Arrhythmia Classification Based on Logistic Model Tree, *Journal of Biomedical Science and Engineering, 2(6)*, 2009, 405-411.

[8] Stanley, and William D., Technical Analysis And Applications With Matlab. Cengage Learning, *ISBN 1401864813*, 2004, p.143.

[9] C.Clapham, J.Nicholson , Oxford Concise Dictionary of Mathematics, Constant Function, *Addison-Wesley*, 2009, p. 175. Retrieved January 12, 2014.

[10] Sumner, Marc, E. Frank, and M. Hall,. Speeding up Logistic Model Tree Induction, *PKDD. Springer*, 2005, 675–683.

[11] N. Landwehr, M. Hall, and E. Frank, Logistic Model Trees, *Machine Learning, 59(1/2)*, 2005, 161-205.

[12] N. Fazakis, S. Karlos, S. Kotsiantis, and K. Sgrabas, Self-Trained LMT for Semi-supervised Learning, *Computational Intelligence and Neuroscience, 2016(2)*, 2016, 1-13.

[13] B. Tan, Y. Tan, and Y. Li, Research on Intrusion Detection System Based on Improved PSO-SVM Algorithm, *Chemical Engineering Transaction, 51*, 2016, 583-588.