Detecting network intrusions using multi-class Logistic Regression and Correlation-based feature selection

Taha Ait tchakoucht[1], Mostafa Ezziyyani[1]

[1] Mathematics and Applications department, UAE, Faculty of Sciences and Techniques, Tangier, Morocco
{t.aittchakoucht, m.ezziyyani}@ fstt.ac.ma

Abstract. Because they're facilitating life, using computers and other intelligent devices associated with internet has become vital in those days. Banking transactions, education, trade marketing, texting ...are all daily and important operations that relies on such technology. Information systems that handle those operations must be kept secure from any intrusive activity. To help ensure that, we must take into consideration several subjects such as access control by managing confidentiality, integrity and availability, as well as deploying detection and prevention tools and mechanisms that help preparing for and dealing with attacks. In this perspective, we propose a network intrusion detection model based on multiclass logistic regression (MLR) and Correlation-based feature selection (CFS). Results will be discussed with respect to NSL-KDD Dataset, and compared to other techniques based on various classification methods.

Keywords: Intrusion detection system; Logistic Regression; Attack; NSL-KDD

1 Introduction

Intrusion detection system is a set of tools destined for traffic analysis in order to capture any abnormal or malicious activity that threatens the continuity of service, and alert security administrators for possible reactions.

Two families of intrusion detection are to mention; Misuse detection and Anomaly detection. The misuse detection approach consists of recognizing attacks that follow intrusion patterns already recognized and reported by experts [11]. Misuse detection systems are vulnerable to intruders who use new patterns of behavior or who mask their illegal behavior to deceive the detection system. Anomaly detection methods were developed to overcome this issue. Anomaly detection is based on the behavior of the user and / or application [10]. A third approach would be the hybrid intrusion detection which combines both techniques in the same time [12]. Often, these approaches are based on Data-mining techniques. Data-mining Techniques is a set of techniques for the extraction of motifs from large data sets, combining statistical and machine learning methods with database management. Those techniques involve learning association rules, cluster analysis, classification and regression.

An IDS could also be whether a Network-based IDS (NIDS) or Host-based IDS (HIDS). The objective of HIDS is to focus on a single machine while he NIDSs look at the packets that pass through the whole network to determine if an attack occurs. In Anomaly-based approach, they consist of establishing a network profile that separates between normal and abnormal activity.

The remainder of this paper is organized as follows: Section (II) represents some related work. Section (III) explains our re-search methodology. In Section (IV), experimental results and performance comparison are discussed. Finally, Section (V) de-scribes conclusion and opens to some perspectives.

2 Related work

In their application, IDSs are based on machine learning techniques, such as support vector machine, Artificial neural network), K-means algorithm, Naive bayes, decision trees…

In [1], Quinlin introduces a machine learning model C4.5, an extension version of ID3 algorithm, based on decision trees where he discusses several topics like tree construction, pruning as a remedy against high variance, conversion to rules, problems related to missing attribute values. The system is widely used, but still suffers from some limitations like the bias in favor of rectangular regions.

When using Naïve Bayes methods, the most usual problem faced is that one of handling continuous variables. One solution considers discretizing or assuming that the data are generated by a single Gaussian. According to [2], this assumption can be vio-lated in some domains, and instead of it, they're using statistical methods for non-parametric density estimation. As a result, the system generalizes better than the former approach. This approach to Bayesian induction bears some similarities to other re-search in machine learning and statistics.

SVMs are also popular machine learning techniques. One example which is widely used is LIBSVM [3].

3 Proposed Model

3.1 NSL-KDD

NSL-KDD[4] is a dataset proposed to solve some of the inherent problems of the KDD'99 [5]. Although, this reformed version of KDD dataset is still suffering from certain problems mentioned by McHugh[6] and may not be a perfect representation of exist-ing real networks, due to the lack of public datasets for network based IDSs, it still can be applied as an effective benchmark da-taset to help compare different intrusion detection models.

The NSL-KDD dataset has the following advantages over the original KDD dataset:

- It does not include redundant records in the train set, so the classifiers will not be biased towards more frequent records.
- There are no duplicate records in the proposed test sets; therefore, the performance of the learners are not biased by the methods which have better detection rates on the frequent records.
- The number of selected records from each difficulty level group is inversely proportional to the percentage of records in the original KDD dataset. As a result, the classification rates of distinct machine learning methods vary in a wider range, which makes it more efficient to have an accurate evaluation of different learning techniques
- The number of records in the train and test sets is reasonable, which makes it affordable to run the experiments on the complete set without the need to randomly select small portion. Consequently, evaluation results of different research works will be consistent and comparable.

The dataset is built on 41 features including:

1. **Basic features:** this category encapsulates all the attributes that can be extracted from a TCP/IP connection.
2. **Traffic features:** this category includes features that are computed with respect to a window interval and is

divided into two groups:

a) *"same host" features:* examine only the last 100 connections having the same destination host as the cur-rent connection, and calculates statistics related to protocol behavior, service, etc...

b) *"same service" features:* examine only the last 100 connections that have the same service as the current connection.

3. **Content features:** unlike most of the DOS and Probing attacks, the R2L and U2R attacks don't have any intrusion fre-quent sequential patterns. This is because the DOS and Probing attacks involve many connections to some host(s) in a very short period of time; however, the R2L and U2R attacks are embedded in the data portions of the packets, and normally involves only a single connection. To detect these kinds of attacks, we need some features to be able to look for suspicious behavior in the data portion, e.g., number of failed login attempts. These features are called content features.

3.2 Preprocessing

Converting symbolic data to numerical data. In addition to the improvements presented in NSL-KDD we perform some preprocessing to prepare an appropriate dataset for usage in the algorithm.

Indeed, there are some categorical features that need to be converted to numerical data such as, *protocol_type*, *Service* (service type) and *Flag* (connection status flag). *protocol_type* counts 3 values, *Service* regroups 70 values and *Flag* is precisely an 11 values feature. So, we replaced every categorical value with a reference integer

We also have attributed a reference number to each class type including the normal class, as shown in Table. 1

Table 1. CLASSES AND REFERENCES.

Class	Reference number	Label
Normal	**1**	Normal
Probe	**2**	Ipsweep, portsweep, nmap, satan, saint, mscan
DOS	**3**	smurf, teardrop, pod, back,land, apache2, udpstrom, mailbomb, processtable, neptune
R2L	**4**	dictionary, ftp_write, guess_password, imap, named, sendmail, spy, xlock, xsnoop, snmpgetattack, httptunnel, worm, snmpguess, multihop, phf, warezclient, warezmaster
U2R	**5**	perl, ps, xterm, rootkit, loadmodule, eject, buffer_overflow, sqlattack

Figures 1 and 2 respectively show 2 original NSL-KDD examples with symbolic data and their converted version with numerical data

0,tcp,ftp_data,SF,491,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,150,25,0.17,0.03,0.17,0.00,0.00,0.00,0.05,0.00,normal

0,icmp,eco_i,SF,18,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,1,16,1.00,0.00,1.00,1.00,0.00,0.00,0.00,0.00,ipsweep

Fig. 1 Data from original dataset

0,2,17,10,491,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,150,25,0.17,0.03,0.17,0.00,0.00,0.00,0.05,0.00,1

0,1,11,10,18,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,1,16,1.00,0.00,1.00,1.00,0.00,0.00,0.00,0.00,2

Fig. 2 Data from Fig.1 after conversion

Mean normalization and feature scaling. One important operation in a machine learning problem with a set of features is to proceed to mean-normalization and fea-ture scaling. We usually encountering a situation where the features vary in different ranges of values. Here for example, in one hand we have the *protocol_type(integer)* varying from 1 to 3. In the other hand, the *duration(real)* varies from 0 to 58329. This will cause the algorithm to converge slowly. Therefore, we have to make sure that the features are on a similar scale, and make them have approximatively zero mean.

In order to do so, we replace x_i with $\frac{xi-\mu i}{si}$. **(1)**

Where x_i the feature value, μ_i the mean value of feature x_i over all training examples,and s_i the standard deviation..

Dimensionality Reduction. For this purpose, we are using the Correlation-based feature selection (CFS) [9] which belongs to the category of filters;

Feature-class and feature-feature correlations are computed. Then, we search for the best dataset in the subsets space.

The CFS's feature subset evaluation function**:**

$$Ms = \frac{\mathrm{kRfc}}{\sqrt{\mathrm{k} + \mathrm{k}(\mathrm{k}-1)\mathrm{Rff}}}. \quad \textbf{(2)}$$

Rfc: Mean feature-class correlation.

Rff: Average feature-feature inter-correlation

Algorithm. To compute the cost value, we are using the following cost function:

$$\mathrm{Cost}(\theta) = -\frac{1}{\mathrm{m}}\sum_{\mathrm{i}=0}^{\mathrm{m}} \mathrm{y}^{(\mathrm{i})} \log\left(\mathrm{h}_\theta\left(\mathrm{x}^{(\mathrm{i})}\right)\right) + \left(1-\mathrm{y}^{(\mathrm{i})}\right)\log\left(1-\mathrm{h}_\theta\left(\mathrm{x}^{(\mathrm{i})}\right)\right) + \frac{\alpha}{2\mathrm{m}}\sum_{\mathrm{j}=1}^{\mathrm{n}} \theta_{\mathrm{j}}^2 \quad \textbf{(3)}$$

$$h_\theta(x) = sigmoid\,(z) \text{ and } z = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n$$

$x^{(i)}$ refers to the i^{th} example in the dataset. $x_1, x_2, \ldots x_n$ are the selected features. h_θ is the hypothesis function (predictor) algorithm. $y^{(i)}$ is the i^{th} example's label value which is whether 0 or 1. m the number of examples in the dataset. n the number of selected features, Θ is parameter vector which size is n+1, and α is regularization parameter.

To minimize that cost function, we first compute its partial derivatives:
.

$$\frac{\partial Cost(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}).x_j^{(i)} \qquad j = 0$$

$$\frac{\partial Cost(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}).x_j^{(i)} + \frac{\alpha}{m}\theta_j \qquad j \geq 1$$

(4)

The cost value and the gradient (partial derivatives) will be passed to the minimization function fmincg [7]. Overall, we employ the following algorithm for Multiclass Logistic Regression MLR. (Syntax is a mixture of Matlab and algorithmic's + vectorized representation)

```
program MLR (Output)
KDDTrain20 Dataset [4]
const lambda:= 1;

var      X: Real % Training set

         y: Real % class labels vector

        m, theta, theta0, grad, cost, hypothesis: Real;

       % theta:the weights vector. cost: the cost
function. grad: partial derivatives of 'cost'.
hypothesis: hypothesis function h_θ

  begin
    cost := 0;

    m = length(y);

    grad := zeros(size(theta));

    theta0 := [0 ; theta(2:end)];

    hypothesis := sigmoid(X*theta);

    cost := (1/2*m)*sum(-y.*log(hypothesis)-(1-
y).*log(1- hypothesis))+...
```

```
(lambda/2*m)*(theta0'*theta0);

grad := (1/m)*X'*( hypothesis -y)+(lambda/m)*theta0;

cost := fmincg(cost,grad) [7]

Use the new Θ resulting of the minimal cost value
and recompute  'hypothesis'.

Compare the new 'hypothesis' and y to evaluate
accuracy on training set and predict classes for new
examples test set 'KDDTest+ and KDDTest-21').


end
```

4 Experimental Results

To evaluate the model, we selected 11 features using feature selection. We applied the first 20% examples of the KDDTrain+ as the training set.

We then experimented our trained model on 2 datasets, the KDDTest+ and KDDTest-21. For comparison, we have selected the widely used J48 decision tree machine learning technique [1] and Naïve Bayes [2] from the Weka [8] collection with the default values as the input parameters for these methods.

In order to have an optimal system, we used a validation set of 12597 records which is 10% of the original dataset, and focused the processing on the parameter α;

Normally if α is too small we fall into a high variance problem (overfitting). If it is too big the system will suffer from high bias (underfitting). eight models are tested. Every model refers to a specific value of α . Figure 3 represent those models;

Remark 1. Bigger values of α result in an increase in the $\mathbf{Cost}_{\mathbf{VALIDATION}}(\boldsymbol{\theta})$. We chose the model whose $\mathbf{Cost}_{\mathbf{VALIDATION}}(\boldsymbol{\theta})$ is minimal, which is ($\boldsymbol{\alpha}$ =1 and $\mathbf{Cost}_{\mathbf{VALIDATION}}(\boldsymbol{\theta})$= 1.78e5). We then trained our system according to that model and tested it on KDDTest+ and KDDTest-21.

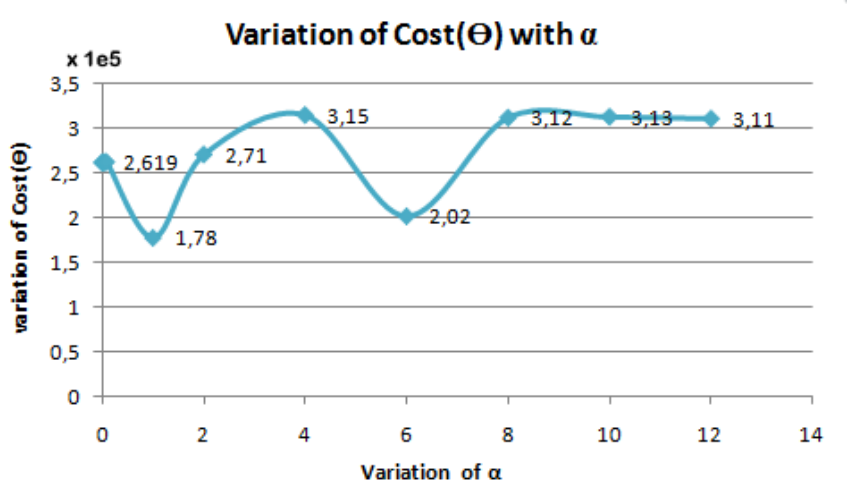

Fig. 3. Variation of Cost with α

To have a good insight on how the system is behaving on new sets of examples especially when treating the rare classes (skewed data), we used the Precision/Recall and FScore metrics;

$$Precision = \frac{\#\ True\ positives}{\#True\ positives + \#false\ positives} \quad (5)$$

$$Recall = \frac{\#\ True\ positives}{\#\ False\ Negatives + \#\ True\ positives} \quad (6)$$

$$FScore = \frac{2.Precision.Recall}{Precision + Recall} \quad (7)$$

We denote TP:True Positives, FP:False Positives, FN:False Negatives, P:Precision , R:Recall and FS:FScore

Tables 2 and 3 represent confusion matrix respectively on KDDTest+ and KDDTest-21.

Table 2. Confusion Matrix On KDDTest+

	Normal	Probe	DOS	R2L	U2R
Normal	9400	143	167	1	0
Probe	244	1701	476	0	0
DOS	1906	46	5506	0	0
R2L	2616	153	118	0	0
U2R	56	5	6	0	0
FP-rate	**34%**	**16,8%**	**12%**	-	-

FN-rate = **21,4%**

Table 3. Confusion Matrix On KDDTest-21

	Normal	Probe	DOS	R2L	U2R
Normal	1935	25	192	0	0
Probe	747	349	1306	0	0
DOS	1894	34	2414	0	0
R2L	2856	14	16	1	0
U2R	61	1	4	1	0
FP-rate	**74%**	**17,5%**	**38,6%**	-	-

FN-rate = **46,9%**

Table 4 represent the FScore indicator for all classes on both KDDTest+ and KDDTest-21.

Table 3. Fscore on both KDDTest+ and KDDTest-21

	KDDTest+			**KDDTest-21**		
	P	R	**FScore**	P	R	**FScore**
Normal	0,66	0,97	**0,79**	0,26	0,90	**0,40**
Probe	0,83	0,70	**0,76**	0,83	0,15	**0,25**
DOS	0,88	0,74	**0,80**	0,61	0,56	**0,58**
R2L	0	0	**0**	0,5	~0	**~0**
U2R	-	-	**-**	-	-	**-**

Remark 2. FScore show good detection performance of Normal,Probe and DOS classes(respectively 0.79 , 0.76 and 0.80) on KDDTest+.
Remark 3. Results on KDDTest-21 are medium, something due to the characteristics of this Dataset (It contains instances that not all of the 21 Machine learners could detect),leading to high values of False positive and False negative rates which affects negatively the Precision and the Recall metrics and consequently the FScore.
Remark 4. Large number of R2L and U2R instances is considered as normal behaviour; 90% of R2L instances and 83% of U2R attack instances with respect to KDDTest+. 98.9% of R2L instances and 91% of U2R instances with respect to KDDTest-21. This happens because of the similarities between R2L, U2R and Normal instances and the lack of R2L and U2R training records (only 209 R2L and 11 U2R in the 20%KDDTrain+, whilst there are 2887 R2L and 67 U2R instances on each test set); As a result, the system hardly detected some R2L attacks and couldn't detect U2R attacks.

Figures 5 and 6 show performance comparison with [1] and [2]

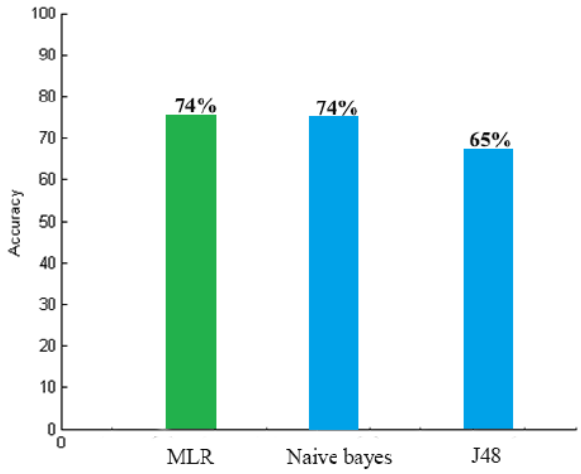

Fig. 5. Performance Comparison with respect to KDDTest+

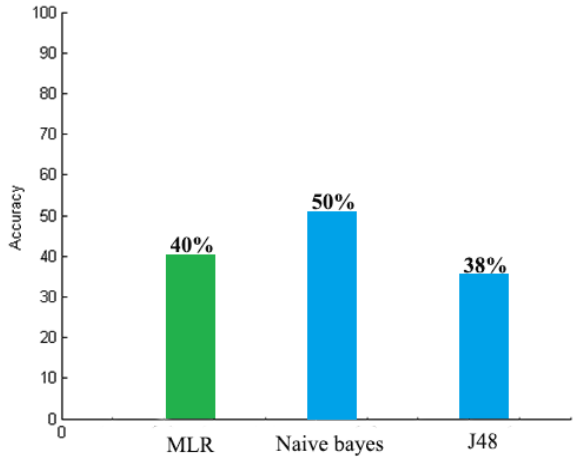

Fig. 6. Performance Comparison with respect to KDDTest-21

5 Conclusion and Outlook

In this article, we presented a NIDS based on Multiclass Logistic Regression. The model consists of several steps going from preprocessing to detection. Usually, the accuracy of classifiers on the original KDDTest of KDD99'is relatively high given that this test set contains skewed data. The accuracy results of classifiers on KDD99' cannot reflect the ability of the classifier. However, even if the accuracy rate is generally medium on KDDTest+ and KDDTest-21, those two test sets are good indicator of classifiers capability.

The method resulted in good accuracy rates compared with other widely used machine learning techniques. The model can be enhanced using additional/alternative machine learning techniques, with more targeted feature engineering and refined prepro-cessing, in order to :

- Reduce the false positives rate(FP).
- Reduce the false negatives rate(FN).
- Decrease the training costs.
- Increase performance detection on R2L and U2R attack types.

References

1. J.Quinlan, C4.5: Programs for machine learning.Morgan Kaufmann, (1993)
2. G. John and P. Langley, "Estimating continuous distributions in Bayesian classifiers," in proceedings of the 11th Conference on Uncertainty in Artificial Intelligence, pp 338-345 (1995)
3. C. Chang and C. Lin.,"LIBSVM : a Library for Support Vector Machines", (2001) Software at http://www.csie.ntu.edu.tw/ cjlin/libsvm
4. NSL-KDD: available on http://www.unb.ca/research/iscx/dataset/iscx-NSL-KDD-dataset.html , (2009)
5. KDD CUP 1999, available on: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
6. J. McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln Laboratory",ACM Transations on Information and system security, vol. 3, no. 4, pp. 262–294 (2000)
7. C. E. Rasmussen, fmincg minimization function. http://learning.eng.cam.ac.uk/carl/code/minimize/
8. "Waikato environment for Knowledge analysis (Weka) and Using Weka in Matlab" http://www.mathworks.com/matlabcentral/fileexchange/50120-using-weka-in-matlab
9. M. A. Hall and L. A. Smith, "Feature Subset Selection: A Correlation Based Filter Approach", University of Waikato, (1997)
10. Denning DE, Edwards DL, Jagannathan R, Lunt TF, Neumann PG, "A prototype IDES: A real-time intrusion detection expert system". Technical report, Computer Science Laboratory, SRI International, Menlo Park
11. M. Roesch, "Snort — lightweight intrusion detection for networks". In Proceedings of the 13th Systems Administration Conference, pp 229 – 238, Seattle, WA, USA, Usenix Association, , (1999)
12. Jagannathan R, Lunt TF, Anderson D, Dodd C, Gilham F, Jalali C, Javitz HS, Neumann PG, Tamaru A, Valdes A, "System Design Document: Next-generation intrusion-detection expert system (NIDES)". Technical report, Computer Science Laboratory, SRI International, Menlo Park, (1993)