Identifying Intrusions in Computer Networks using Principal Component Analysis and Random Forest

Md. Al Mehedi Hasan, Shamim Ahmad, Md. Khademul Islam Molla

Computer Science and Engineering, University of Rajshahi,
Rajshahi, Bangladesh.
Email: mehedi_ru@yahoo.com, shamim_cst@yahoo.com, khademul.cse@ru.ac.bd

Abstract

An Intrusion Detection System (IDS) is an important and necessary component in ensuring network security as well as in protecting network resources and infrastructures. In general, IDS deals with huge amount of data even for a small network, which may contain numerous irrelevant and redundant features. Extraneous features can make it harder to detect suspicious behavior patterns and cause higher resource consumption but produce poor detection rate. Therefore, feature reduction or extraction is an indispensable pre-processing step when mining huge datasets that can significantly improve the overall system performance. In this paper, we effectively apply principal component analysis (PCA) in introducing a process for feature extraction as well as for reduction. In this case, PCA reduces the number of features in order to decrease the complexity of the system and then, trained Random Forest is used to identify all kinds of attacks. We verify the effectiveness and the feasibility of the proposed IDS system by several experiments on KDD'99 dataset which are based on DARPA 98 dataset and provide labeled data for researchers working in the field of intrusion detection. The important deficiency in the KDD'99 data set is the huge number of redundant records as observed earlier. Therefore, we have derived a data set RRE-KDD by eliminating redundant record from KDD'99 train and test dataset, so the classifiers and feature extraction method will not be biased towards more frequent records. This RRE-KDD consists of both KDD99Train+ and KDD99Test+ dataset for training and testing purposes, respectively. Moreover, since, PCA requires that each data instance should be expressed as a vector of real numbers, therefore, in our work, each categorical features in the KDD is converted into required number of dummy variables which is meaningful from statistical viewpoint. The experimental results show that our proposed method not only reduces the number of the input features but also increases the classification accuracy

Keywords: Principal Component Analysis, Random Forest, KDD'99 dataset, RRE-KDD dataset, Variable Reduction, Feature Extraction.

1. Introduction

The internet and local area networks are growing larger in recent years. As a great variety of people all over the world are connecting to the Internet, they are unconsciously encountering the number of security threats such as viruses, worms and attacks from hackers [1]. Now firewalls, anti-virus software, message encryption, secured network protocols, password protection and so on are not sufficient to assure the security in computer networks, when some intrusions take advantages of weaknesses in computer systems to threaten. Therefore, Intrusion Detection Systems (IDSs) have become a necessary addition to the security infrastructure of most organizations [2]. Deploying highly effective IDS systems is extremely challenging and has emerged as a significant field of research, because, theoretically it is not possible to set up a system with no vulnerabilities [3]. Several machine learning (ML) algorithms, for instance, Neural Network [4], Genetic Algorithm [5], Support Vector Machine [2, 6], clustering algorithm [7] and more have been extensively employed to detect intrusion activities from large quantity of complex and dynamic datasets.

Current Intrusion Detection Systems (IDS) examine all data features to detect intrusion or misuse patterns [2, 6, 8]. Since the amount of audit data that an IDS needs to examine is very large even for a small network, therefore, analysis is getting difficult even with computer assistance because extraneous features can make it harder to detect suspicious behavior patterns [8, 9]. As a result, better IDS should be capable to reduce the amount of data to be processed too. This is very important if real-time detection is desired. Moreover, in the problem of high dimensional

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

feature space, some of the features may be redundant or irrelevant. Consequently, removing of these redundant or irrelevant features is very important; otherwise, they may deteriorate the performance of classifiers [10]. The feature extraction and reduction involve to find a subset of features to improve prediction accuracy or to decrease the size of the structure without significant compromising in prediction accuracy of the classifier based on only those extracted features [10].

Literature survey showed that, most of the researchers used randomly generated records or a portion of record from the KDD'99 dataset to develop feature extraction or reduction method and to build intrusion detection system [8, 11, 12] without using the whole train and test dataset. So, those findings will not reflect the actual extraction or reduction of features for classification. Although some researcher used the whole dataset but did not remove redundant records, which implies a limitation of having a chance of redundant record used for the same feature extraction or reduction and because of that, classification methods may be biased toward to the class that has redundant record [13, 14, 15]. These limitations motivated us to find out the actual reduction of feature for classification based on the whole train and test dataset of KDD'99 by removing redundant record. In this paper, we adapt Principal component analysis (PCA) to reduce dimensions of features, which can bring to a successful conclusion in intrusion detection. . We aim to filter out redundant information and significantly reduce number of computer resources, both memory and CPU time required to detect attacks.

However, PCA requires that each data instance should be expressed as a vector of real numbers. This is usually done by the using binary encoding, where, each categorical variable having possible m values should be replaced by (m-1) dummy variables. In [11, 12], PCA is used to reduce the 41 features to 5 and 8 but nothing is described how categorical features were encode. In [10] PCA also used to reduce the number of features from 41 to 23 and described that symbolic features are mapped to numeric values but did not clearly mention what approach was taken for that mapping purpose. In [14, 15, 16], PCA is used in reducing the number of features but nothing is described how symbolic features were encoded. Some researchers used only integer code to convert category features to numeric representation instead of using dummy variables which is not statistically meaningful way for this type of conversion [10, 17]. In our work, each categorical feature in the KDD is converted into required number of dummy variables which is meaningful from statistical viewpoint. Finally, this paper suggest Principal component analysis (PCA) as a reduction tool and Random Forest as a learning tool for the developed system, firstly we reduce the features and then apply the learning algorithm.

The remainder of the paper is organized as follows. Section 2 provides the description of KDD'99 dataset. We outline mathematical overview of PCA, RF and Proposed Intrusion Detection Model in Section 3. Experimental setup is presented in Section 4 and Preprocessing and RF model selection are drawn in Section 5 and 6 respectively. Feature Extraction from connection space is discussed in section 7. Finally, Section 8 reports the experimental result followed by conclusion in Section 9.

2. KDDCUP'99 Dataset

Under the sponsorship of Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory (AFRL), MIT Lincoln Laboratory has collected and distributed the datasets for the evaluation of researches in computer network intrusion detection systems [18]. The KDD'99 dataset is a subset of the DARPA benchmark dataset prepared by Sal Stofo and Wenke Lee [19]. The KDD data set was acquired from raw tcpdump data for a length of nine weeks. It is made up of a large number of network traffic activities that include both normal and malicious connections. The KDD99 data set includes three independent sets; ''whole KDD'', ''10% KDD'', and ''corrected KDD''. Most of researchers have used the ''10% KDD'' and the ''corrected KDD'' as training and testing set, respectively [17, 20]. The training set contains a total of 22 training attack types. The ''corrected KDD'' testing set includes an additional 17 types of attack and excludes 2 types (spy, warezclient) of attack from training set, so therefore there are 37 attack types that are included in the testing set, as shown in Table 1 and 2 . The simulated attacks fall in one of the four categories [2, 17, 21]: (a) Denial of Service Attack (DoS), (b) User to Root Attack (U2R), (c) Remote to Local Attack (R2L), (d) Probing Attack. A connection in the KDD-99 dataset is represented by 41 features, each of which is in one of the continuous, discrete and symbolic form, with significantly varying ranges [22, 23].

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

Classification of Attacks	Attack Name
Probing	Port-sweep, IP-sweep, Nmap, Satan
DoS	Neptune, Smurf, Pod, Teardrop, Land, Back
U2R	Buffer-overflow, Load-module, Perl, Rootkit
R2L	Guess-password, Ftp-write, Imap, Phf, Multihop, spy, warezclient, Warezmaster,

Table I. Attacks in KDD'99 Training dataset

Classification of Attacks	Attack Name
Probing	Port-Sweep, Ip-Sweep, Nmap, Satan, **Saint, Mscan**
DoS	Neptune, Smurf, Pod, Teardrop, Land, Back, **Apache2,Udpstorm, Processtable, Mail-Bomb**
U2R	Buffer-Overflow, Load-Module, Perl, Rootkit, **Xterm, Ps, Sqlattack**.
R2L	Guess-Password, Ftp-Write, Imap, Phf, Multihop, Warezmaster, **Snmpgetattack, Named, Xlock, Xsnoop, Send-Mail, Http-Tunnel, Worm, Snmp-Guess.**

Table II. Attacks in KDD'99 Testing dataset

2. 1 Inherent Problems and Criticisms against the KDD'99

Statistical analysis on KDD'99 dataset found important issues which highly affects the performance of evaluated systems and results in a very poor evaluation of anomaly detection approaches [24]. The most important deficiency in the KDD data set is the huge number of redundant records. Analyzing KDD train and test sets, Mohbod Tavallaee found that about 78% and 75% of the records are duplicated in the train and test set, respectively [25]. This large amount of redundant records in the train set will cause learning algorithms to be biased towards the more frequent records, and thus prevent it from learning unfrequent records which are usually more harmful to networks such as U2R attacks. The existence of these repeated records in the test set, on the other hand, will cause the evaluation results to be biased by the methods which have better detection rates on the frequent records.

To solve these issues, we have derived a new data set RRE-KDD by eliminating redundant record from KDD'99 train and test dataset (10% KDD and corrected KDD), so the classifiers will not be biased towards more frequent records. This RRE-KDD dataset consists of KDD99Train+ and KDD99Test+ dataset for training and testing purposes, respectively. The numbers of records in the train and test sets are now reasonable, which makes it affordable to run the experiments on the complete set without the need to randomly select a small portion.

3. Feature Extraction, Classification and Proposed Model

Principal component analysis (PCA) is a mathematical procedure that transforms a number of (possibly) correlated variables into a (smaller) number of uncorrelated variables called principal components [26]. On the other side, the Random Forest is an ensemble of unpruned classification or regression trees [27]. In this paper, we discuss a novel intrusion detection method based on Random Forest in a lower dimensional subspace which has been derived by PCA in order to reduce the computational complexity and memory significantly.

3.1 Principal component analysis

PCA technique reduces the dimensionality of the data while retaining as much as possible of the variation present in the original dataset [26]. The steps of principal component analysis are given below:

1. The first step is to obtain a set S with M vectors (connection records of KDD99Train+ dataset) where each record is an N dimension vector.

$$S = \{\Gamma_1, \Gamma_2, \Gamma_3, \dots\dots\dots, \Gamma_M\}$$

2. Second step is to obtain the mean vectors of connection records Ψ.

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

$$\Psi = \frac{1}{M}\sum_{i=1}^{M} \Gamma_i$$

3. Then find the difference Φ between the input vector and the mean vector

$$\emptyset_i = \Gamma_i - \Psi$$

4. The covariance matrix C has been obtained in the following manner

$$C = \frac{1}{M}\sum_{i=1}^{M} \emptyset_i \emptyset_i^T$$
$$= AA^T \qquad A = \{\emptyset_1, \emptyset_2, \emptyset_3 \dots\dots\dots, \emptyset_M\}$$

5. Compute the N eigenvalues of C: $\lambda_1 > \lambda_2 > \lambda_3 > \dots\dots\dots\dots > \lambda_N$

6. Find the N eigenconnection (eigenvectors of covariance matrix) from the covariance matrix, C: $v_1,\ v_2, v_3, \dots\dots\dots\dots, v_N$. Since C is symmetric, $v_1,\ v_2, v_3, \dots\dots\dots\dots, v_N$ form a basis and any vector Γ_i or actually $\emptyset_i$ can be written as a linear combination of the eigenvectors.

7. Project each of the original record into connection space (eigenspace). This gives a vector of weights representing the contribution of each eigenconnection to the reconstruction of the given connection record vector.

$$w_k = v_k^T(\Gamma_i - \Psi) \qquad \Omega_i^T = \left[w_1, w_2 w_3, \dots\dots\dots w_N\right]$$

where v_k is the k^{th} eigenconnection (eigenvector), w_k is the k^{th} weight (k^{th} new feature) in the vector $\Omega_i^T = [w_1, w_2 w_3, \dots\dots\dots w_N]$, and Ω_i is the new feature representation of Γ_i in the connection space (eigenspace).

It should be noted that, in the connection space (eigenspace), the new features are called principal components. In practice, the number of the principal components chosen depends on the precision we wish to reach [13]. For unsupervised cases of classifications, the dimensionality of the subspace k can be determined by

$$\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{N} \lambda_i} \geq \alpha$$

where α is the ratio of variation in the subspace to the total variation in the original space. But, in this paper, we used a supervised approach to choose the k, where the accuracy is found highest in testing dataset.

3.2 Random Forest

Consider the problem of separating the set of training vectors belong to two separate classes, $(x_1, y_1), (x_2, y_2), \dots\dots, (x_n, y_n)$ where $x_i \in R^p$ and $y_i \in \{-1, +1\}$ is the corresponding class label, $1 \leq i \leq n$. The main task is to find a classifier with a decision function f(x, θ) such that y = f(x, θ), where y is the class label for x, θ is a vector of unknown parameters in the function.

The Random forest is a well-known classifier and it has been applied broadly in many classifications problems. **It** generates many **classification** trees and each tree is constructed by a different bootstrap sample from the original data using a tree classification algorithm. After the forest is formed, a new object that needs to be classified is put down each of the tree in the forest for classification. Each tree gives a vote that indicates the tree's decision about the class of the object. The forest chooses the class with the most votes for the object.

The random forests algorithm (for both classification and regression) is as follows [28, 29]:

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

I. From the Training of n samples draw n_{tree} bootstrap samples.
II. For each of the bootstrap samples, grow classification or regression tree with the following modification: at each node, rather than choosing the best split among all predictors, randomly sample m_{try} of the predictors and choose the best split from among those variables. The tree is grown to the maximum size and not pruned back. Bagging can be thought of as the special case of random forests obtained when $m_{try} = p$, the number of predictors.
III. Predict new data by aggregating the predictions of the n_{tree} trees (i.e., majority votes for classification, average for regression).

There are two ways to evaluate the error rate. One is to split the dataset into training part and test part. We can employ the training part to build the forest, and then use the test part to calculate the error rate. Another way is to use the OOB error estimate. Because random forests algorithm calculates the OOB error during the training phase, we do not need to split the training data. In our work, we have used both ways to evaluate the error rate.

There are three tuning parameters of Random Forest: number of trees (n_{tree}), number of descriptors/predictors randomly sampled as candidates for splitting at each node (m_{try}) and minimum node size [29]. When the forest is growing, random features are selected at random out of the all features in the training data. The number of features employed in splitting each node for each tree is the primary tuning parameter (m_{try}). To improve the performance of random forests, this parameter should be optimized. The number of trees should only be chosen to be sufficiently large so that the OOB error has stabilized. In many cases, 500 trees are sufficient (more are needed if descriptor importance or intrinsic proximity is desired). There is no penalty for having "too many" trees, other than waste in computational resources, in contrast to other algorithms which require a stopping rule. Another parameter, minimum node size, determines the minimum size of nodes below which no split will be attempted. This parameter has some effect on the size of the trees grown. In Random Forest, for classification, the default is 1, ensuring that trees are grown to their maximum size and for regression, the default is 5 [29].

3.3 Proposed Intrusion Identification Model

This paper suggest principal component analysis (PCA) as a reduction tool and Random Forest as a learning tool for the developed system, firstly we extract and reduce the features and then apply the learning algorithm. This approach involves the following procedure:

I. Acquire an initial set of records (this set is called the training set). In this paper, we use the KDD99Train+ training dataset containing M=144585 connection records.

II. Calculate N eigenconnections (eigenvectors) from the training set, keeping only k (k<<N) eigenconnections that correspond to the highest eigenvalues. These N eigenconnections define the connection space. In our case, the value of N=115 instead of 41, detail explanation is given in section 5.

III. Calculate a set of weights which will be the new features or principal components in connection space by projecting the input connection record vector onto each eigenconnection.

IV. Once training and testing data is projected onto the new feature space, that is, connection space, apply Random Forest classifier to detect intrusions from the projected test dataset placed in the new feature space.

4. Dataset and Experimental Setup

Investigating the existing papers on the feature selection as well as extraction and anomaly detection which have used the KDD data set, we found that a subset of KDD'99 dataset has been used for training, testing and selecting/Extracting feature instead of using the whole KDD'99 dataset [8, 11, 12]. Moreover, existing papers anomaly detection mainly used two common approaches to apply KDD [25]. In the first, KDD'99 training portion is employed for sampling both the train and test sets. However, in the second approach, the training samples are randomly collected from the KDD train set, while the samples for testing are arbitrarily selected from the KDD test

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

set. In our work, we have used the whole dataset (Train and Test Data of KDD'99) by removing redundant records. The basic characteristics of the original KDD'99 and RRE-KDD (KDD99Train+ and KDD99Test+) intrusion detection datasets in terms of number of samples is given in Table 3. Although the distribution of the number of samples of attack is different on different research papers, in our previous work, we have found out the distribution of attack given in Table 3 [2, 6]. In our experiment, whole train (KDD99Train+) dataset has been used to train our classifier and the test (KDD99Test+) set has been used to test the classifier. All experiments were performed using Intel core i5 2.27 GHz processor with 4GB RAM, running Windows 7.

Dataset	Various Independent Sets	Normal	DoS	Probing	R2L	U2R	Total
Original KDD'99 Dataset	WholeKDD	972780	3883370	41102	1126	52	4898430
	10% KDD	97278	391458	4107	1126	52	494021
	KDD corrected	60593	229853	4166	16347	70	311029
RRE-KDD Dataset	KDD99Train+	87832	54572	2130	999	52	145585
	KDD99Test+	47913	23568	2678	3058	70	77287
	TrainSet (For Model Selection)	8784	5458	213	100	6	14561
	ValidationSet (For Model Selection)	8784	5458	213	100	6	14561

Table III. Number of Samples of Each Attack in Dataset

To select the best model in model selection phase, we have drawn 10% samples from the training set (KDDTrain+) to tune the parameters of Random Forest and another 10% samples from the training set (KDDTrain+) to validate those parameters, as shown in Table III.

5. Pre-processing

PCA is not able to process KDD99Train+ and KDD99Test+ dataset in its standard format. PCA requires each data instance should be presented as a vector of real numbers. Hence preprocessing is required prior to apply PCA in any feature extraction system. In the KDD-99 dataset, the features in columns 2, 3, and 4 in the KDD'99 dataset are the protocol type, the service type, and the flag, respectively. The value of the protocol type may be tcp, udp, or icmp; the service type could be one of the 66 different network services such as http and smtp; and the flag has 11 possible values such as SF or S2. Hence, as a part of preprocessing, the categorical features in the KDD dataset must be converted into a numeric representation. This is done by the usual binary encoding, where each categorical variable having possible m values is replaced with (m-1) dummy variables. Here a dummy variable is set to 1 (one) for a specific category and set to 0 (zero) for all other categories. After converting all categories to numeric values, we got 115 variables for each samples of the dataset.

However, some researchers used only integer code to convert category features to numeric representation instead of using dummy variables approach which is not statistically meaningful way for this type of conversion [10, 17]. The next step of pre-processing process is the scaling of the training data, i.e. normalizing of all features so that they have zero mean and a standard deviation of 1. This process prevents the numerical instabilities during the PCA calculation. Then, we used the same scaling approach on the test dataset.

Finally, attack names are mapped to one of the five classes namely Normal, DoS (Denial of Service), U2R (user-to-root: unauthorized access to root privileges), R2L (remote-to-local: unauthorized access to local from a remote machine), and Probe (probing: information gathering attacks).

6. Model Selection of RF

In order to generate highly performing classifiers capable of dealing with real data an efficient model selection is required. In this section, we present the experiments conducted to find efficient model for RF. To improve the detection rate, we optimize the number of the random features (m_{try}). We build the forest with different m_{try} (5, 6, 7,

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

10, 15, 20, 25, 30, 35, and 38) over the train set (TrainSet), then plot the OOB error rate and the time to build the classifier RF corresponding to different m_{try}. As Fig. 1 shows, the OOB error rate reaches the minimum when m_{try} is 7. Besides, increasing m_{try} increases the time to build the classifier. Thus, we choose 7 as the optimal value, which reaches the minimum of the OOB error rate and costs the least time among these values.

There are two other parameters in Random Forest: number of trees and minimum node size. In order to find the value for the number of trees (n_{tree}), we have used TrainSet for training and ValidationSet for testing and compared the OOB error rate with the independent test set (ValidationSet) error rate, for Random Forest as the number of trees increases; see Fig. 2. The plot shows that the OOB error rate tracks the test set error rate fairly closely, once there are a sufficient number of trees (around 100). Fig. 2 also shows an interesting phenomenon which is the characteristic of Random Forest: the test and OOB error rates do not increase after the training error reaches zero; instead they converge to their "asymptotic" values, which is close to their minimum.

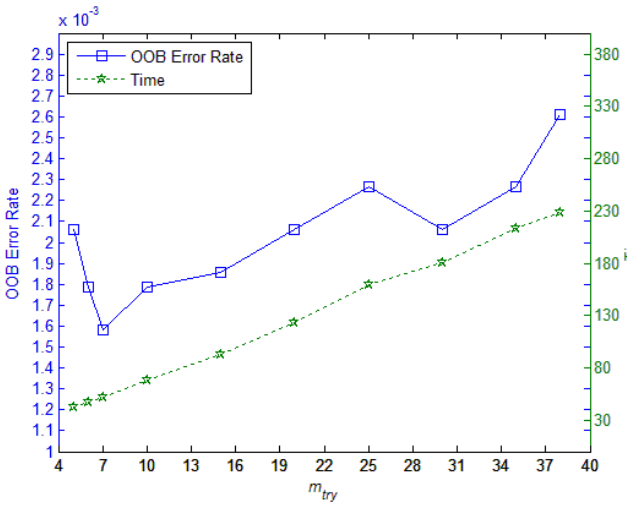

Figure 1: OOB Error Rates and required time to train for Random Forest Vs the number of m_{try}.

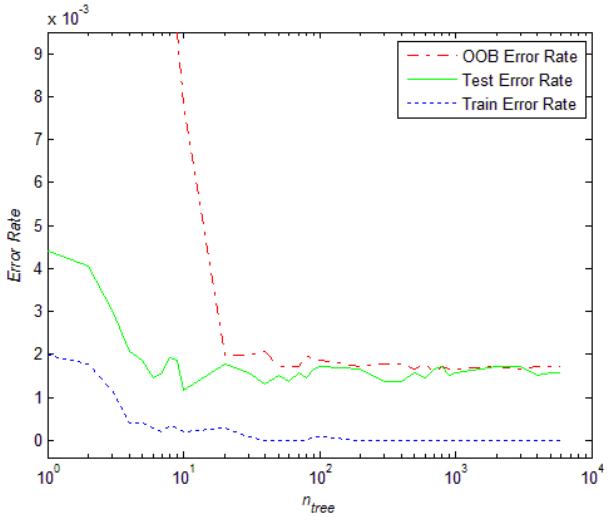

Figure 2: Comparison of Training, Out-of-Bag, and Independent Test Set Error Rates for Random Forest Vs the Number of Trees.

To determine the minimum node size, we have used TrainSet for training and compared the OOB error rate with varying the number of node size using n_{tree} = 100 and m_{try}= 7, as shown in Fig. 3. The plot shows that default value 1 (one) for classification gives the lowest OOB error rate.

7. Feature Extraction from Connection Space

In this paper, KDD99Train+ dataset has been used to find out the reduced feature space. There are 115 coordinates of each connection in KDD99Train+ are found after converting categorical features as explained in section 3. We have applied PCA to KDD99Train+ data set that generates N (here N=115) eigenconnections (eigenvectors) those form a connection space or feature space (eigen space) as well as their corresponding eigen values. After finding the eigenconnections, KDD99Train+ data has been projected on the formed connection space which, in turn, produced a new train dataset. Following the same manner, we have also projected the test data set KDD99Test+ and got a new test dataset on the connection space.

After getting the new train and test data set on the connection space, we have trained the RF classifier with different number of features with keping n_{tree}=100 and m_{try}= default value and tested the RF with the same number of features using test dataset as shown in Figure 4.

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

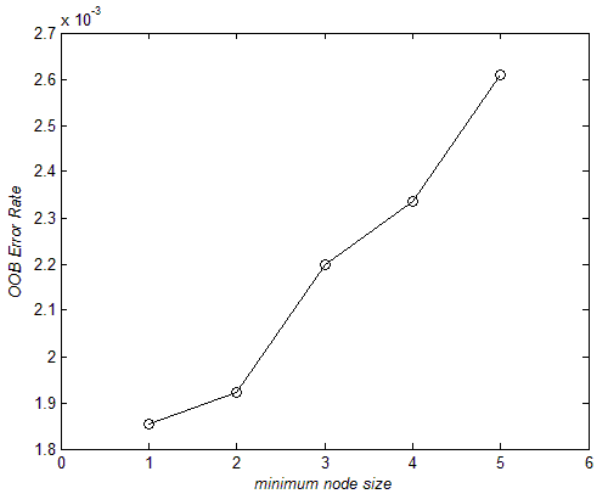

Figure 3: OOB Error Rates for Random Forest Vs the Number of Minimum Node size.

From figure 4, it is perceived that the classification accuracy has an increasing trend from the number of feature 5 to 25 and following that the accuracy started to decrease with some fluctuation. Observing this graph, we have considered three cases to analyze the accuracy: (i) first 5 features, (ii) first 10 features and (iii) first 25 features in the connection space. For every cases, we have tuned the m_{try} value following the procedure as defined in section 6 and got m_{try}=3, 3 and 6 for the first, second and third cases, respectively. We have also tuned the value of n_{tree} and got n_{tree} =100 for all cases by following the procedure described as in section 6.

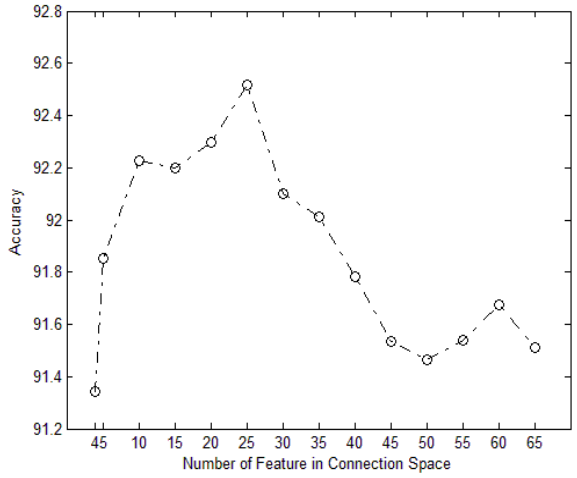

Figure 4: Classification Accuracy Vs Number of PCA Score

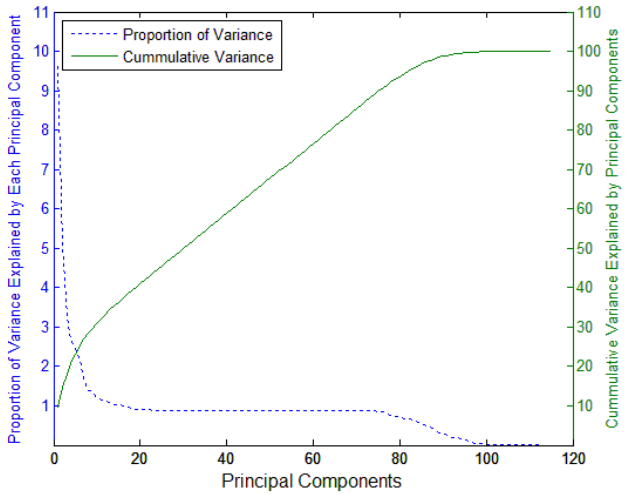

Figure 5: Variance Explained by the PCA

To give the detail analysis about the characteristics or contribution of each principal component, the cumulative variance of principal components and proportion of variance of each principal component are given in figure 5. This figures shows that first principal components provides the best explanation of the variance as expected, the degree of explanation of the following principal components decreases upto 25 principal components and then becomes almost steady state.

8. Results and Discussion

The final training and testing phase is concerned with the development and evaluation of the tuned RF model based on the optimal set of hyper-parameters found in the model selection phase [30]. After finding the set of parameters of RF with 41 features in original space and three cases (for the number of features 5, 10 and 25) in the connection

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

space (features space) as described in section 6 and 7 respectively, we have built the model using the whole train dataset (KDD99Train+) and finally, we have tested the trained model using the test dataset (KDD99Test+) both in the original space as well as in the connection space.

The found classification accuracy in both training and testing dataset are tabulated in Table 4. This table shows that all of the test accuracy for RF with selected features (5, 10, 25) in connection space is better than RF classifier with 41 variables in the original space. Although, it is expected that reducing feature requires less time to train the dataset than to train with the whole feature, however, in our case, RF with 5, 10, or 25 features took more time than RF with 41 variables. The reason behind it may be explained that the range of unique value of each variable in connection space is very much higher than that of it in the original space which, in turn, causes this computational complexity to build the RF classifier. In the connection space, the average number of the unique value of each variable is 144584, whereas the average number of unique value is 474.27 in the original space.

Table 4: Training and Testing Accuracy

Classifier	Training Accuracy	Testing Accuracy	Train Time (min)
RF with 41 Variables in Original Space	100	91.41	10.62
RF with 5 Variables in Connection Space	100	91.85	25.93
RF with 10 Variables in Connection Space	100	92.23	37.32
RF with 25 Variables in Connection Space	100	92.52	51.94

For the test dataset, the confusion matrix for each of the situations is given in Table 5, 6, 7 and 8 respectively. Going into more detail of the confusion matrix, it is seen that RF with selected features in the connection space performs better to detect all types of attacks than that of using RF classifier with 41 features in original space.

		Actual					
		Dos	Normal	Probing	R2L	U2R	%
Prediction	Dos	21795	55	185	0	0	98.91
	Normal	1609	46585	231	2649	67	91.09
	Probing	164	1272	2262	405	1	55.12
	R2L	0	1	0	4	1	66.67
	U2R	0	0	0	0	1	100
	%	92.48	97.22	84.47	0.13	1.43	

Table 5: Confusion matrix for Random Forest with 41 Variables in Original Data Space

We also considered the false positive rate and precision for each of the situation and these are shown in Table 9 and 10 respectively. The RF with reduced features in the connection space for almost all cases (5, 10, 25) gives higher precision and lower false positive rate than RF with all variables in original space.

		Actual					
		Dos	Normal	Probing	R2L	U2R	%
Prediction	Dos	22024	687	628	10	0	94.33
	Normal	1415	46685	376	2294	64	91.84
	Probing	129	202	1674	150	0	77.68
	R2L	0	339	0	603	2	63.88
	U2R	0	0	0	1	4	80
	%	93.45	97.44	62.51	19.72	5.71	

Table 6: Confusion matrix for Random Forest with 5 Variables in Connection Space

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

		Actual					
		Dos	Normal	Probing	R2L	U2R	%
Prediction	Dos	22206	587	471	13	0	95.40
	Normal	1217	47065	413	2696	61	91.47
	Probing	145	260	1794	142	0	76.63
	R2L	0	1	0	205	2	98.56
	U2R	0	0	0	2	9	81.82
	%	94.22	98.23	66.99	6.70	12.5	

Table 7: Confusion matrix for Random Forest with 10 Variables in Connection Space

		Actual					
		Dos	Normal	Probing	R2L	U2R	%
Prediction	Dos	22125	61	694	15	0	96.64
	Normal	1308	47516	384	2662	50	91.52
	Probing	135	335	1600	132	0	72.66
	R2L	0	1	0	249	6	97.266
	U2R	0	0	0	2	14	87.5
	%	93.88	99.17	59.75	8.14	20	

Table 8: Confusion matrix for Random Forest with 25 Variables in Connection Space

Classifier	Dos	Normal	Probing	R2L	U2R	Average False Positive Rate
RF with 41 Variables in Original Space	7.52	2.77	15.53	99.87	98.57	44.85
RF with 5 Variables in Connection Space	6.55	2.56	37.49	80.28	94.29	44.23
RF with 10 Variables in Connection Space	5.78	1.77	33.01	93.3	87.5	44.27
RF with 25 Variables in Connection Space	6.12	0.83	40.25	91.86	80	43.81

Table 9: False Positive Rate (%) of Random Forest for each of the Attack Types.

Classifier	Dos	Normal	Probing	R2L	U2R	Average Precision
RF with 41 Variables in Original Space	98.91	91.09	55.11	66.67	100	82.36
RF with 5 Variables in Connection Space	94.33	91.84	77.68	63.88	80	81.55
RF with 10 Variables in Connection Space	95.4	91.47	76.63	98.56	81.82	88.78
RF with 25 Variables in Connection Space	96.63	91.52	72.66	97.27	87.5	89.12

Table 10: Precision (%) of Random Forest for each of the Attack Types.

9. Conclusion

In this paper we have presented a Random Forest model for Intrusion Detection Systems (IDS) focusing on the improvement of the performance by reducing the number of input features using PCA. From the obtained experimental results, it is evident that the PCA based feature extraction and reduction process is very promising. In the real-world applications, the smaller number of feature are naturally expected in terms of both data management, reduced computational complexity and better detection accuracy. The results indicate that the ability of the RF classification with reduced features (5, 10 or 25 features in connection space) produce more accurate result than Random Forest classification with all features (41 features in original space). Research in feature extraction and intrusion detection using PCA and RF approach is still an ongoing area because of its good performance. The findings of this paper will be very useful for the research on feature reduction and classification. In addition to it, these findings also show how effectively PCA and RF can be used to maximize the rate of performance and to minimize the false positive rate.

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

References

1. Suebsing, Anirut, and Nualsawat Hiransakolwong. "Euclidean-based feature selection for network intrusion detection." In International Conference on Machine Learning and Computing, vol. 3, pp. 222-229. 2011.
2. Hasan, Md Al Mehedi, Mohammed Nasser, Biprodip Pal, and Shamim Ahmad. "Support vector machine and random forest modeling for intrusion detection system (IDS)." Journal of Intelligent Learning Systems and Applications 6, no. 1 (2014): 45.
3. Adebayo, O. Adetunmbi, Zhiwei Shi, Zhongzhi Shi, and Olumide S. Adewale. "Network anomalous intrusion detection using fuzzy-Bayes." In International Conference on Intelligent Information Processing, pp. 525-530. Springer US, 2006.
4. Cannady, James. "Artificial neural networks for misuse detection." In National information systems security conference, pp. 368-81. 1998.
5. Pal, Biprodip, and Md Al Mehedi Hasan. "Neural network & genetic algorithm based approach to network intrusion detection & comparative analysis of performance." In Computer and Information Technology (ICCIT), 2012 15th International Conference on, pp. 150-154. IEEE, 2012.
6. Wang, Qiang, and Vasileios Megalooikonomou. "A clustering algorithm for intrusion detection." In Defense and Security, pp. 31-38. International Society for Optics and Photonics, 2005.
7. Chen, Yuehui, Ajith Abraham, and Ju Yang. "Feature selection and intrusion detection using hybrid flexible neural tree." In International Symposium on Neural Networks, pp. 439-444. Springer Berlin Heidelberg, 2005.
8. Lee, Wenke, Salvatore J. Stolfo, and Kui W. Mok. "A data mining framework for building intrusion detection models." In Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on, pp. 120-132. IEEE, 1999.
9. Heba, F. Eid, Ashraf Darwish, Aboul Ella Hassanien, and Ajith Abraham. "Principle components analysis and support vector machine based intrusion detection system." In 2010 10th International Conference on Intelligent Systems Design and Applications, pp. 363-367. IEEE, 2010.
10. Shilpa lakhina1, Sini Joseph and Bhupendra verma."Feature Reduction using Principal Component Analysis for Effective Anomaly–Based Intrusion Detection on NSL-KDD", International Journal of Engineering Science and Technology, Vol. 2(6), 1790-1799, 2010.
11. Rupali Datti, Shilpa Lakhina, "Performance Comparison of Features Reduction Techniques for Intrusion Detection System", IJCST Vol. 3, Issue 1, 2012.
12. Yacine Bouzida, Fr´ed´eric Cuppens, Nora Cuppens-Boulahia and Sylvain Gombault. "Efficient Intrusion Detection Using Principal Component Analysis", In Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics, 2003.
13. Shailendra Singh, Sanjay Silakari and Ravindra Patel. "An efficient feature reduction technique for intrusion detection system" 2009 International Conference on Machine Learning and Computing, IPCSIT vol.3, 2011.
14. Venkatachalam, V., and S. Selvan. "Performance comparison of intrusion detection system classifiers using various feature reduction techniques." International journal of simulation 9, no. 1, pp. 30-39, 2008.
15. Vasan, K. Keerthi, and B. Surendiran. "Dimensionality reduction using Principal Component Analysis for network intrusion detection." Perspectives in Science, 2016.
16. M. Bahrololum, E. Salahi and M. Khaleghi. "Anomaly Intrusion Detection Design Using Hybrid Of Unsupervised And Supervised Neural Network", International Journal of Computer Networks & Communications (IJCNC), Vol.1, No.2, July 2009.
17. MIT Lincoln Laboratory, DARPA Intrusion Detection Evaluation, http://www.ll.mit.edu/CST.html,MA, USA. July, 2010.

https://sites.google.com/site/ijcsis/
ISSN 1947-5500