Computer Network Intrusion Detection using various Classifiers and Ensemble Learning

Ali H. Mirza
Department of Electrical and Electronics Engineering
Bilkent University, Ankara 06800, Turkey
mirza@ee.bilkent.edu.tr

***Abstract*—In this paper, we execute anomaly detection over the computer networks using various machine learning algorithms. We then combine these algorithms to boost the overall performance. We implement three different types of classifiers, i.e, neural networks, decision trees and logistic regression. We then boost the overall performance of the intrusion detection algorithm using ensemble learning. In ensemble learning, we employ weighted majority voting scheme based on the individual classifier performance. We demonstrate a significant increase in the accuracy through a set of experiments KDD Cup 99 data set for computer network intrusion detection.**

Keywords. Network intrusion, ensemble, anomaly, online learning, classification

I. Introduction

Anomaly detection [1] is of pivotal interest in the field of network intrusion detection, medical diagnosis, fraud detection etc. The basic assumption is that a huge amount of normal data originates from a particular distribution but unknown. Whereas, few unlikely and rare observations, i.e., anomalies, originates from different unknown distributions [2]. In some domains like network intrusion detection, anomaly detection is of prime importance. This is because, a single malicious attack, i.e., an anomaly is sufficient enough to override the system and may cause severe damage [3]. For example, an unusual and out of the routine traffic in a computer network depicts the presence of a network attack. Anomaly detection deals with identifying data points in the data that does not fit with the rest of the data [1]. In the era of big data, anomaly detection helps in identifying rare events and malicious data. Anomaly detection is a form of classification where we develop a model that predicts whether a data point belongs to a given particular distribution or not. The deviated data points are termed as anomalies or outliers. These anomalies may or may not belong to a particular distribution and are rare [2]. Typically, anomaly detection is an unsupervised learning technique, but the accuracy can be improved by using a semi-supervised version of anomaly detection [1], [2], [4]. The anomalies can be point, contextual and collective anomalies [1].

Bayesian approach is used in classification and regression problems for detecting outliers in the data [5], [6]. Some distribution based models like kernel or Parzen density estimator is also used for anomaly detection [7]. Parzen density estimator has a linear complexity which is quite slow for a very large amount of data [8], [9]. This method is highly unsuitable for high dimensional data.

978-1-5386-1501-0/18/$31.00

Kernel-based methods are commonly used for anomaly detection. Among the popular kernel based anomaly detection techniques, Hidden Markov Anomaly Detection (HMAD) [10], One class - Support Vector Machine (OC-SVM) [11] and Support Vector Data Description (SVDD) [12] are widely used. HMAD is used where there is a latent dependency in the input data sequence. Whereas, SVDD learns a hyperplane that encloses the majority of the data such that the outliers are outside the boundary of the hypersphere. On the other hand, OC-SVM attempts to learn a hyperplane that separates the data points from the origin with maximum margin.

In this paper, we employ three different machine learning algorithms that perform binary classification. We use neural networks, decision trees and logistic regression as our three classifiers. Then in order to boost the overall performance, we use ensemble learning approach. In ensemble learning, we employ majority voting technique to enhance the classification performance.

II. Problem Description

Given a computer network data, the task for the classifier is to learn a predictive model, i.e. a classifier, capable of distinguishing between legitimate and illegitimate connections in a computer network. The trained classifier protects a computer network from malicious attacks and unauthorized users. The attacks that may damage the system fall into four major categories:

- DOS - Denial Of Service
- R2L - Unauthorized access from a remote machine
- U2R - Unauthorized access to local super-user privileges
- Probing - Surveillance and Port Scanning

In short, the main task is Anomaly Detection where the illegitimate or bad connections are termed as anomalies or outliers. In this task, we have a multi-classification problem; the attacks come from four groups namely DOS, R2L, U2R and Probing. The fifth class is the normal class. We will perform binary classification whether the data is anomalous or not.

A. Data Description and Preprocessing

The data set [13] contains 41 variables in an ASCII formatted file. There are 5 Million data samples in the original data set. We are using a 10-percent corrected KDD Cup 99 data set. This reduced data set contains overall 0.5 million samples. We have divided it in the ratio of 60 percent-20 percent-20 percent,i.e., 0.3 Million (60 percent) is used for training, 0.1 Million for validation tasks, and remaining 0.1 Million for serve as unseen testing data. The preprocessing file extracts the variables in double and character strings, as the data set contains continuous and discrete variables. We perform one hot encoding on the data set. This preprocessed data set is then stored in the workspace, which is used for all the implementation and testing.

B. Principal Component Analysis

In a classification problem, feature extraction is treated as pre-processing technique. The aim is not only to reduce the computation complexity but also to obtain better classification accuracy. Feature extraction can be treated as a transformation of data, $Y = T(X), X \in R^D, T \in R^d$, and the transformation T, which reduces the dimensions of data from D to d, is obtained via optimization suitable objectives. Hence, the feature extraction is responsible for formulating suitable objectives and determining the corresponding optimal solution of T. Principal Component Analysis is performed to extract only 2 weighted features from original 41 features, making it computationally efficient data set. The eigenvalues and vectors are obtained via Singular Value Decomposition in sorted form. The singular value decomposition of $m \times n$ matrix Q is a factorization of the form $U\Sigma V^*$. The diagonal matrix Σ contains eigenvalues of the matrix. The columns of U and V are vectors corresponding to eigenvalues. We extract first 2 principal components and use this transformed featured data set in our experiments for training logistic regression, decision trees and neural networks classifiers. For comparison purposes, we show the effect of the number of features of the data set on the time taken for training, individual class performances and overall performance. We demonstrate this performance by using Logistic Regression classifier. Fig. 1 and Fig. 2 depicts the effect of feature reduction on the performance and computational complexity. For a number of PCA features equal to 2, we observe that we have a comparable high individual class performance and overall performance. For class 1, we have anomalies there, the class 1 and overall class performance deteriorates for a number of features equal to 3 and 4. After that, the performance again increases but at the cost of time taken for training. So, as a result, we use this transformed data set in order to reduce the computational complexity while maintaining a high efficiency.

III. Classifiers

We use three types of classifiers namely: Logistic Regression, Neural Networks and Decision Trees. The details of these classifiers are as follows:

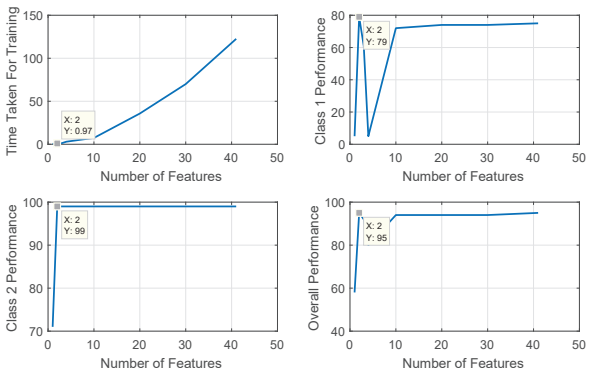

Fig. 1. Effect of feature reduction on performance and computational complexity.

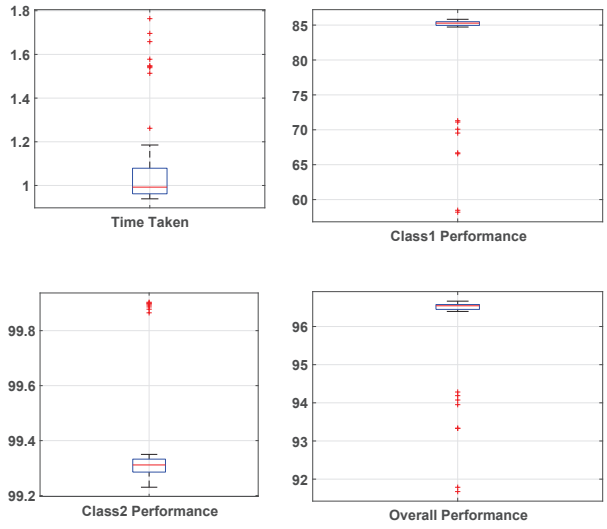

Fig. 2. 2-fold Validation of the feature Reduction.

A. Classifier # 1 : Logistic Regression

The first method used is the logistic Regression [14]. It is variant of linear regression used for the classification task. The sigmoid function is used to provide the conversion from a continuous variable into discrete occurrences. We have not used the ordinary equation formula. As we have millions of data and calculating the inverse and other multiplication can be inaccurate. We defined a loss function, and then using optimization gradient function, we minimized over loss function. The loss function used is the cross-entropy and is defined as:

$$J(\beta) = \frac{1}{n}\sum_{i=1}^{n}\Big[-y(i)\log\big(h_\beta(x(i))\big) - \big(1-y(i)\big)\log\big(1-h_\beta(x(i))\big)\Big] \quad (1)$$

TABLE I
TRAINING AND TESTING DATA ACCURACIES FOR NORMAL AND INTRUSION DETECTION USING LOGISTIC REGRESSION CLASSIFIER

Classes/Datasets	Training	Testing
Normal Connection	89.30	80.92
Intrusion Detection	98.96	99.86
Overall	96.66	96.13

Where the gradient is calculated as for each input feature vector as,

$$\frac{\partial}{\partial \beta_j} J(\beta) = \frac{1}{n} \sum_{i=1}^{n} \Big(h_\beta(x(i)) - y(i) \Big) x_j(i). \tag{2}$$

The sigmoid function used is defined as

$$h_\beta(x) = g(\beta^T x)$$
$$g(z) = \frac{1}{1 + \exp(-z)} \tag{3}$$

The cost function decreases with increase in a number of iterations, as the gradient tries to approach towards optimal minima. The classifier is trained over training data and then tested over testing data. For the simulations part, we are using PCA transformed data set that contains 2 principal components. The training and testing set are of $300,000 \times 2$ and $94,021 \times 2$ dimensions. The class-based and overall accuracies are shown in Table. I

The accuracy obtained describes the outcome of this classifier, as we want to be sure in the case when we give intrusion detection, that is the reason we have increased performance, in this case, we dont want to ignore any intruder.

B. Classifier # 2 : Neural Networks

We implemented single hidden layer multilayer perceptron [4] as our second method. We selected the corresponding network parameters as: number of neurons in hidden layer $= p = 3$ and learning rate = $\eta = 0.01$. Since we have binary classification task at our hand, we used Binary Cross Entropy as our loss function. The definition of a cross entropy function is as follows:

$$E = -\sum_{i=1}^{n_{class}} t_i \log(x_i) \tag{4}$$

where n_{class} is the total number of classes and x_i represents the softmax output of the i^{th} output layer of MLP. Here, t_i is the true label. We use stochastic gradient descent to learn the model parameters. We used $\eta = 0.01$ as the optimal learning rate value after doing cross-validation. We perform our experiments over a given number of epochs. As the data set is highly unbalanced, in order to train the network smoothly, we down-sampled the majority class, i.e., anomaly class in our case, so that we can get best results. Also, the KDD Cup data is highly rigorous, in order to obtain the best result out of the neural network, we selected the best subset of training data after down-sampling it. After obtaining such subset, we then validate our training and testing performance. We use

TABLE II
TRAINING AND TESTING DATA ACCURACIES FOR NORMAL AND INTRUSION DETECTION USING NEURAL NETWORKS CLASSIFIER

Classes/Datasets	Training	Testing
Normal Connection	92.72	91.73
Intrusion Detection	96.93	91.93
Overall	90.67	89.83

TABLE III
TRAINING AND TESTING DATA ACCURACIES FOR NORMAL AND INTRUSION DETECTION USING DECISION TREES CLASSIFIER

Classes/Datasets	Training	Testing
Normal Connection	91.08	88.71
Intrusion Detection	95.00	93.02
Overall	92.08	91.66

sigmoid as an activation function in the hidden layer and softmax function in the output layer. The gradient with respect to Hidden-Ouptut Layer and Input-Hidden weights is given as

$$\delta v_i = (t_i - x_i) * net \tag{5}$$

$$\delta w_j = \sum_{i=1}^{n_{class}} (t_i - x_i)(net)(w_{ji})(net_j(1 - net_j))x \tag{6}$$

where v_i is the hidden to i^{th} output layer weight coefficient matrix. Similarly, w_j is the input to j^{th} hidden layer weight coefficient matrix. Also net_j is the sigmoid output to the product of input and input-hidden weight matrix. The values for weight matrices are updated as follow

$$v_i = v_i + \eta \delta v_i \tag{7}$$

$$w_j = w_j + \eta \delta w_j \tag{8}$$

In the simulation setup, after cross-validation on parameters, we selected $p = 3, \eta = 0.01$ as our final best parameters. We randomized the data by shuffling the samples so that we can have an overall feasible and steady result. We observed that in our network, the error stops decreasing after 24 epochs. We show the cumulative cross-entropy error vs. number of epochs and report class-based accuracies, overall accuracy and confusion matrices for training and testing. Here again, we are using the PCA transformed version of the data. The training, validation and testing data sets are of $300,000 \times 2$,$100,000 \times 2$ and $94,021 \times 2$ dimensions respectively. The class-based and over all accuracies are shown in Table. II

C. Classifier # 3 : Decision Trees

As our third method of classification, we used decision trees to perform intrusion detection. The tree model we implemented is CART model [15]. Since the task at hand is to declare whether the current sample belongs to a normal class or not. We have two classes to classify using decision trees. We implemented the decision trees using pre-pruning approach. Among the popular pre-pruning approaches like a minimum limit on entropy, probability limit etc, we used a minimum number of samples in node criteria. This means that whenever the number of samples for a certain class decreases below

TABLE IV
TRAINING AND TESTING DATA ACCURACIES FOR NORMAL AND INTRUSION DETECTION USING ENSEMBLE LEARNING CLASSIFIER

Classes/Datasets	Training	Testing
Normal Connection	92.48	92.00
Intrusion Detection	99.73	99.94
Overall	97.53	96.14

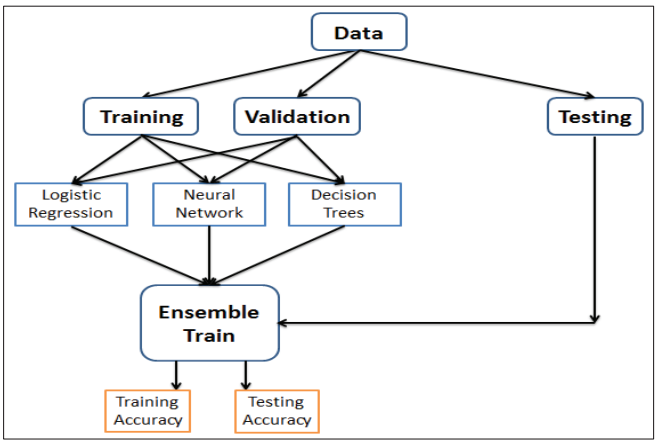

Fig. 3. Ensemble Learning Model.

a certain threshold (pre-pruning parameter), it is considered to be a pure node and stops further splitting. The splitting criterion is based on selecting the tree node that has the minimum entropy among all the possible splits calculated for all discrete and continuous features. Since the training data is highly unbalanced, we down-sampled the training data to construct a tree. After constructing the tree, we then used the whole unbalanced training data and testing data and compute the accuracies and confusion matrices. The class-based and over all accuracies are shown in Table. III

Decision Trees were more adaptable for class 0. It did not performed better on class 1. So in order to handle such incompetences in all implemented classification methods, we then decided to use Ensemble Learning.

IV. ENSEMBLE LEARNING

In order to boost the overall performance, we implemented ensemble learning technique [16]. We combined the results of all the classifiers in training and testing phase by extracting the most useful information out of all the classifiers. To extract the class 1, we gave more weight to neural networks output. To extract the class 0, we gave more weight to decision trees output As a whole, on the average, we gave 50 percent weight to the neural network, 20 percent weight to logistic regression and 30 percent weight to decision trees. Then we classified as class 0 if the value is less than 0.5 threshold and vice versa. The ensemble learning scheme is shown in Fig. 3. The class-based and over all accuracies are shown in Table. IV

V. CONCLUSION

We implemented three classifiers (logistic regression, neural networks, decision trees) for network intrusion detection. We reduced the dimensions of the dataset by using PCA. We observe that each classifier works for a particular class of decision trees favours more normal class. On the other hand, neural networks and logistic regression favours both classes. We boosted the overall performance by introducing ensemble learning. By designating certain weights to each of the classifiers, we decide whether the sample is anomalous or not. Since the main assumption of anomaly detection is that anomaly ratio should be very small. But in our data set, the anomaly rate is very high, i.e., approximately 80 percent. With this problem, we still managed to get good results by using ensemble learning.

REFERENCES

[1] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[2] M. Markou and S. Singh, "Novelty detection: a reviewpart 1: statistical approaches," *Signal processing*, vol. 83, no. 12, pp. 2481–2497, 2003.

[3] J. P. Anderson, "Computer security threat monitoring and surveillance," *Technical Report, James P. Anderson Company*, 1980.

[4] M. Markou and S. Singh, "Novelty detection: a reviewpart 2:: neural network based approaches," *Signal processing*, vol. 83, no. 12, pp. 2499–2521, 2003.

[5] C. M. Bishop, *Neural networks for pattern recognition*. Oxford university press, 1995.

[6] D. J. MacKay, *Bayesian methods for adaptive models*. PhD thesis, California Institute of Technology, 1992.

[7] V. Barnett and T. Lewis, *Outliers in statistical data*. Wiley, 1974.

[8] L. Tarassenko, P. Hayton, N. Cerneaz, and M. Brady, "Novelty detection for the identification of masses in mammograms," 1995.

[9] G. Ritter and M. T. Gallegos, "Outliers in statistical pattern recognition and an application to automatic chromosome classification," *Pattern Recognition Letters*, vol. 18, no. 6, pp. 525–539, 1997.

[10] N. Görnitz, M. Braun, and M. Kloft, "Hidden markov anomaly detection," in *International Conference on Machine Learning*, pp. 1833–1842, 2015.

[11] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[12] D. M. Tax and R. P. Duin, "Support vector data description," *Machine learning*, vol. 54, no. 1, pp. 45–66, 2004.

[13] S. D. Bay, D. F. Kibler, M. J. Pazzani, and P. Smyth, "The uci kdd archive of large data sets for data mining research and experimentation," *SIGKDD Explorations*, vol. 2, p. 81, 2000.

[14] S. Menard, *Applied logistic regression analysis*, vol. 106. SAGE publications, 2018.

[15] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[16] T. G. Dietterich, "Ensemble learning," *The handbook of brain theory and neural networks*, vol. 2, pp. 110–125, 2002.