Why Machine Learning Algorithms Fail in Misuse Detection on KDD Intrusion Detection Data Set

Maheshkumar Sabhnani and Gursel Serpen
Electrical Engineering and Computer Science Department
The University of Toledo
Toledo, OH 43606, USA

Abstract

A large set of machine learning and pattern classification algorithms trained and tested on KDD intrusion detection data set failed to identify most of the user-to-root and remote-to-local attacks, as reported by many researchers in the literature. In light of this observation, this paper aims to expose the deficiencies and limitations of the KDD data set to argue that this data set should not be used to train pattern recognition or machine learning algorithms for misuse detection for these two attack categories. Multiple analysis techniques are employed to demonstrate, both objectively and subjectively, that the KDD training and testing data subsets represent dissimilar target hypotheses for user-to-root and remote-to-local attack categories. These techniques consisted of switching the roles of original training and testing data subsets to develop a decision tree classifier, cross-validation on merged training and testing data subsets, and qualitative and comparative analysis of rules generated independently on training and testing data subsets through the C4.5 decision tree algorithm. Analysis results clearly suggest that no pattern classification or machine learning algorithm can be trained successfully with the KDD data set to perform misuse detection for user-to-root or remote-to-local attack categories. It is further noted that the analysis techniques employed to assess the similarity between the two target hypotheses represented by the training and the testing data subsets can readily be generalized to data set pairs in other problem domains.

Keywords: Intrusion detection, Machine learning, KDD data set, decision trees, cross validation

1. Introduction

The Knowledge Discovery in Databases (KDD) 1999 data set, which was introduced for the 1999 KDD Cup contest, has been recently utilized extensively for development of intrusion detection systems through a suite of pattern recognition and machine learning algorithms for four main attack categories: namely Probing, Denial of Service (DoS), User-to-Root (U2R), and Remote-to-Local (R2L). Such efforts, as reported in the recent literature, suggests relatively poor performance profile for at least part of the functionality spectrum for the attempted intrusion detection systems. Specifically, pattern recognition and machine learning algorithms trained with the KDD training data subset and tested on the KDD testing data subset failed to detect majority of U2R and R2L attacks within the context of misuse detection. In fact, not a single classification algorithm whose performance even comes close to an acceptable level, for all practical purposes, has been reported in the literature to date. Some of the researchers did try to explain the dismal performance figures obtained through reasons associated with training algorithms themselves, the training process itself or some other factor, but not the potential limitations of the data subsets themselves [1,2]. This observation points at an important issue to address since the KDD data set is one of few in the domain of intrusion detection and as such attracts significant attention from the researchers due to its well-defined and readily accessible nature [1,3,4,5,6,7,8,9]. It then becomes relevant and important to establish what can and what cannot be done with this data set.

This paper exposes the deficiencies and limitations associated with the representation of U2R and R2L attack categories in the KDD data set for misuse detection context and attempts to explain why the KDD data set should not be used to train pattern recognition or machine learning algorithms for these two attack categories. Multiple analysis techniques will be used to demonstrate, both objectively and subjectively, that the KDD training and testing data subsets represent dissimilar hypotheses[1] for the U2R and R2L attack categories within a misuse detection context. Hence, any pattern classification and machine learning algorithm that employs the KDD training data subset to learn the attack signatures[2] of these categories is very likely to demonstrate poor performance on the KDD testing data subset.

Section 2 discusses various U2R and R2L attacks present in the KDD training and testing data subsets while also presenting the distribution of attack patterns. Section 3 presents a literature survey, which highlights various algorithms applied in the recent past to the KDD data set for U2R and R2L attack categories. Section 4 describes each of the three analysis techniques employed to show why the KDD data set cannot be leveraged to train algorithms from the domain of pattern recognition and machine learning for U2R and R2L attack categories for misuse detection. Section 4.1 discusses the performance of C4.5 decision tree algorithm on the KDD training and testing data subsets. Section 4.2 discusses the cross-validation approach, which is applied to merged KDD training and testing data subsets using multiple classifier algorithms including a Binary Tree classifier, a Multi Layer Perceptron neural network classifier, and a Gaussian classifier. Section 4.3 presents a qualitative comparison of the various rules created through the C4.5 algorithm on the KDD training and testing data subsets. Finally, Section 5 concludes this paper by highlighting strengths and weaknesses of the KDD data set and suggesting its appropriate use within the context of intrusion detection.

[1] A hypothesis is the estimate of a target concept or function learned through a set of training examples of that target concept or function [22].
[2] An attack signature is a distinctive complex pattern used to detect system penetration, which may involve comparison of audit and log data from a variety of sources within the computing platform or infrastructure.

2. KDD Data Set: U2R and R2L Attack Categories

In 1998, the United States Defense Advanced Research Projects Agency (DARPA) funded an "Intrusion Detection Evaluation Program (IDEP)" administered by the Lincoln Laboratory at the Massachusetts Institute of Technology. The goal of this program was to build a data set that would help evaluate different intrusion detection systems (IDS) in order to assess their strengths and weaknesses. The objective was to survey and evaluate research in the field of intrusion detection. The computer network topology employed for the IDEP program involved two sub networks: an "inside" network consisting of victim machines and an "outside" network consisting of simulated real-world Internet traffic. The victim machines ran Linux, SunOS^{TM}, and Solaris^{TM} operating systems. Seven weeks of training data and two weeks of testing data were collected. Testing data contained a total of 38 attacks, 14 of which did not exist in the training data. This was done to facilitate the evaluation of potential IDSs with respect to their anomaly detection performance. Three kinds of data was collected: transmission control protocol (TCP) packets using the "tcpdump" utility, basic security module (BSM) audit records using the Sun Solaris^{TM} BSM utility, and system file dumps. This data set is popularly known as DARPA 1998 data set [10].

One of the participants in the 1998 DARPA IDEP [11], used only TCP packets to build a processed version of the DARPA 1998 data set [10]. This data set, named in the literature as KDD intrusion detection data set [12], was used for the 1999 KDD Cup competition, which allowed participants to employ it for developing IDSs. The KDD data set was consequently submitted to the University of California at Irvine "Knowledge Discovery in Databases" archive, and consists of approximately 5 million training and 0.3 million testing records. Both training and testing data subsets cover four major attack categories: Probing (information gathering attacks), Denial-of-Service (deny legitimate requests to a system), User-to-Root (unauthorized access to local super-user or root), and Remote-to-Local (unauthorized local access from a remote machine). Each record consists of 41 features [2], where 38 are numeric and 3 are symbolic, defined to characterize individual TCP sessions. Data mining techniques and domain knowledge were utilized to formulate features for different connections using the TCP packets [11,13].

A User-to-Root (U2R) attack is characterized by a process whereby any normal system user illegally gains access to the super-user privileges. Generally, a system defect or bug is exploited to execute a successful privilege transition from user level to root level. Buffer overflows are the most common type of attack mechanisms in this category. Other U2R attacks take advantage of root programs that do not manage temporary files in the system properly. Some U2R attacks occur because of an exploitable race condition in a single program, or two or more programs executing concurrently. Though these defects or bugs can be relatively easily patched, any new attacks with previously unknown mechanisms can result in insurmountable damage to the system, as the malicious user attains full control of the victim machine at the root level.

Generally most machines are accessible over the network through the Internet, but only authorized users are intended to be able to access the machines remotely. A Remote-to-Local (R2L) attack occurs when an attacker who does not have an account on the victim machine, gains local access as a user of the victim machine by sending network packets through standard protocols like TCP/IP or UDP. There are many ways in which an R2L attack can be executed. Buffer overflow vulnerabilities in some networking programs like `sendmail`, `imap`, or `named` can result in local access on the victim. Attacks like `dictionary`, `guest`, `ftp-write`, and

`Xsnoop` exploit system misconfigurations. Some attacks involve social engineering, an example being the `Xlock` attack, which is a Trojan horse program and used to initiate the screensaver on Solaris machines in order to capture the user's password and send it to the attacker.

The KDD training data subset has 52 U2R and 1126 R2L records, while the testing data subset has 228 U2R and 16189 R2L records. Four new U2R attacks are present only in the KDD testing data subset and records for these new attacks constitute around 80% of all U2R records in the testing data subset. Similarly, seven new R2L attacks are present only in the KDD testing data subset, and more than 60% of R2L records in the KDD testing data subset belong to these new R2L attacks.

For the two attack categories (U2R and R2L), many attacks and their records are present only in the testing data subset. Misuse detection is thus difficult, unless signatures show similarity between the attacks common to both the training and the testing data subsets and those new ones existing only in the testing data subset. It is therefore important to determine the similarity between attack signatures in the KDD training and testing data subsets to assess an upper bound on the expected misuse detection performance of a classifier algorithm on these data subsets.

3. Misuse Detection Performance for U2R and R2L Attack Classes: Literature Survey

A large number of pattern classification and machine learning algorithms have been applied to the KDD data set in order to develop intrusion detection systems for U2R and R2L attacks. This section presents a brief overview of these algorithms, their application to the KDD data set, and their performance results on the U2R and R2L attack categories.

C5.0 decision trees were employed by the winner of the 1999 KDD Cup, an intrusion detection competition [14]. The training process utilized 50 data subsets each having all records from both the U2R and the R2L attack categories, 4,000 records from the Probing category, 80,000 records from normal category, and 400,000 records from the DoS category. This was done to make sure that there were sufficient records present from each attack category to build decision tree models. For each of the above training data subsets, the researchers created ten C5.0 decision trees using error cost and boosting options. The final predictions were computed on top of the 50 predictions each obtained from one decision tree, by minimizing the conditional risk (sum of error-costs multiplied by class-probabilities). However, this classifier only detected 10% of the U2R records and 8% of the R2L records from the KDD testing data subset. It had a false alarm rate 0%.

Another machine learning algorithm applied to the KDD data set was Kernel Miner [2]. Kernel Miner is a data mining tool based on a global optimization model for classifying data and predicting the results of new cases using automatically generated decision trees (versus a single tree generated by the C4.5 algorithm). It does this by initially constructing locally optimal set of decision trees (called a decision forest) from which the optimal subset of trees (called the sub-forest) is selected for predicting new cases. Levin used a random sample of 10% of the KDD training data for the training data subset. A multi-class detection approach was used to detect different attack categories in the KDD data set. The final decision trees scored very high detection rates for all classes in the entire training data subset. Out of 311,029 test examples in the KDD data set, the classifier was able to correctly categorize 289,006 records, i.e. 92.92%. However, the classifier achieved detection rates of 11.84% and 7.32% for the U2R and the R2L attack categories, respectively. False alarm rates of 38.64% and 2.5% were generated for the

U2R and the R2L attack categories, respectively, on the entire testing data subset. Finally, the classifier was able to detect only 11.2% of the new attack examples present in the testing data subset: hence the tool failed at anomaly detection as well. ‘Over-fitting’ was suggested as one of the reasons for the dismal performance, where the number of records in the U2R and the R2L attack categories were significantly less in the training data subset than in the testing data subset.

Agarwal and Joshi [1] proposed a two-stage general-to-specific framework for developing a rule-based (*PN*-rule) classifier model on a data set that has widely different class distributions in the training data set. The proposed model consists of positive rules (*P*-rules) that predict the presence of the class, and negative rules (*N*-rules) that predict the absence of the class. In the first stage, *P*-rules are created using the training data to increase the probability of detection. After that, a set of *N*-rules is created to reduce the false-alarm rate for the learned classifier model. Finally all the *P*-rules and *N*-rules are applied in the sequence in which they were created, to the test data. The *PN*-rule method approach was applied on the KDD data set to develop a classifier model, which was able to detect only 6.6% of U2R attacks and 10.7% of R2L attacks in the KDD testing data subset. The false alarm rate was negligible. The testing data subset contained 17 new attack records not present in the training data subset. For known attacks (misuse detection only), *PN*-rule detected 23.1% of records in the U2R attack category and 28.9% of records in the R2L attack category, which is still very low for all practical purposes.

Yeung and Chow [9] proposed a novelty detection approach using non-parametric density estimation based on Parzen-window estimators with Gaussian kernels. Parzen [23] introduced a non-parametric approach for estimating the probability density function $p(\mathbf{x})$ from a set of n points represented by vectors $\mathbf{x}_i$, where $i=1,2,\ldots,n$, using kernel functions. The estimated probability function *p(***x***)* can be expressed as a mixture of radially symmetrical Gaussian kernels with common variance σ^2 as

$$p(\mathbf{x}) = \frac{1}{n(2\pi)^{d/2}\sigma^2} \sum_{i=1}^{n} \exp\left\{ -\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{2\sigma^2} \right\},$$

where *d* is the dimensionality of the feature space. Two sets of normal (attack free) data are required to build a model. The first set of normal data is used to build the non-parametric density estimation model *M*. The second set of normal data is used to define thresholds for the model. It is important to note that this non-parametric approach does not need training as it relies on estimating probability distribution. The model *M* is used to find whether test pattern **x** belongs to *M* using the Bayes decision rule. This novelty detection approach was employed to detect attack categories in the KDD data set. Symbolic features were represented by a group of binary-valued variables. The resulting feature vectors used to test this technique had 119 dimensions. 30,000 randomly sampled normal records from the KDD training data subset were used as training data to estimate the density of the model. Another 30,000 randomly sampled normal records (also from the KDD training data subset) formed the threshold determination set, which had no overlap with the training data subset. Their classifier detected 93.57% and 31.17% of U2R and R2L attack records, respectively, in the KDD testing data subset. The advantage of this technique is that no intrusion data is required: this effectively also introduces an anomaly detection capability. On the other hand, the disadvantage is that it cannot differentiate whether the intrusion belongs to a DoS or a U2R attack, etc. It can only indicate intrusive activity. Hence it cannot be used for multi-class detection, as required for the KDD data set. Furthermore, an increase in the false alarm rate can also be expected if the normal data is not a sufficiently good model.

Sabhnani [15] applied numerous pattern classification and machine learning algorithms on the KDD data set to develop intrusion detection systems. Numerous classifiers including a multilayer perceptron neural network, an incremental radial basis function neural network, a Gaussian classifier, a *K*-means classifier, a nearest neighbor classifier, a C4.5 decision tree, a Fuzzy ARTMAP classifier, a Leader cluster, and a Hyper sphere algorithm were developed. Each classifier model was built using the KDD training data subset and then evaluated using the KDD testing data subset. None of the classifiers were able to detect more than 25% of the attack records with acceptable false alarm rates.

The literature survey indicates that pattern recognition and machine learning algorithms trained and tested with the KDD data set demonstrate poor performance in terms of probability of detection versus the false alarm rate for the U2R and the R2L attack categories within the context of misuse detection. Although some researchers [1,2] suggested reasons for this failure, none have presented a conclusive and convincing argument to explain it to date. We believe certain intrinsic features of the KDD data set might explain why pattern classification and particularly trainable algorithms fail to demonstrate a significant detection rate for the U2R and R2L attack categories. The following sections explain why trainable algorithms that employ the KDD data set fail for the U2R and R2L attacks. Both objective and subjective analyses are used to demonstrate the deficiencies and limitations associated with the KDD data set.

4. Analysis of KDD Data Set

This section describes analyses demonstrating that the KDD training and testing data subsets represent dissimilar target hypotheses for the U2R and R2L attack categories. This characteristic violates the basic requirement for any trainable classifier algorithm to succeed.

Three separate procedures are applied for the analysis and the following sections elaborate on the procedures and results.

In Section 4.1, one of the procedures requires two scenarios to be studied: in one case, the original KDD training data subset is employed to induce a rule set through the C4.5 algorithm, whose performance is tested on the original KDD testing data subset. The second case involves utilizing the original KDD testing data subset as the training data and inducing a set of rules through the C4.5 algorithm. Performance of the decision tree is tested on the original KDD training data subset. These two cases collectively suggest whether or not the target hypotheses represented by the two KDD data subsets are similar.

In Section 4.2, cross-validation approach will be leveraged to evaluate if two data subsets represent similar target hypotheses. Two KDD data subsets will be merged to form a data superset which will be randomly partitioned into five equal-sized subsets or folds. Out of five folds, four will be used to develop classifiers and the fifth one will be employed to test performance. The idea is to assess if 80% of records in the merged superset (compared to all the records in the KDD training data subset) possesses sufficient information to develop a high-performance classifier.

Section 4.3 presents a qualitative analysis of knowledge represented by two KDD data subsets by comparing the two rule sets induced separately from each data subset. Since rule sets offer a convenient means for human interpretability of knowledge entailed by the associated data subset, a comparison between two rule sets is feasible to assess the similarity in knowledge, and consequently in target hypotheses represented by the two data subsets.

No preprocessing is performed on the KDD data set to ensure that the rules are created without any bias from scaling and other transformations, while noting that rules created from the original KDD data set will tend to be more intelligible.

4.1 C4.5 Decision Tree Algorithm Performance on KDD Data Set for U2R and R2L Attack Categories

The C4.5 algorithm [16] creates a decision tree from training data, while trying to maximize the probability of detection and reduce errors for each class in training data. In this study, two separate trees were created using the KDD data set. One decision tree was created using the KDD training data subset and the second one using the KDD testing data subset. The decision tree created using the KDD training data subset was tested on the KDD testing data subset and vice versa. After creating the decision tree models for the U2R and R2L attack categories, optimized rules were extracted using the `C4.5rules` utility, which is provided with the C4.5 decision tree software tool [17]. There were approximately 1000 nodes in the unpruned decision trees for the R2L models and 300 nodes for the U2R models.

Table 1 compares models for the U2R category with two classes, U2R and Not U2R. Model 1 used the KDD training data subset for creating the decision tree and Model 2 used the KDD testing data subset. Similarly, Table 2 compares models to detect two classes, R2L and Not R2L. The following notation is used in these tables: ‘KDD-R’ – KDD training data subset, ‘KDD-T’ – KDD testing data subset. Similar notation is used in Table 2.

	Model 1		**Model 2**	
	Training	**Testing**	**Training**	**Testing**
Data Subset	KDD-R	KDD-T	KDD-T	KDD-R
Number of Not-U2R Records	1074938	310797	310782	1073987
Not-U2R Record Detection Rate	99.99%	99.99%	99.99%	99.91%
Number of U2R Records	40	14	222	23
U2R Record Detection Rate	76.92%	6.14%	97.37%	44.23%

Table 1: Comparison of C4.5 Decision Tree Performance for the U2R Attack Category

	Model 1		**Model 2**	
	Training	**Testing**	**Training**	**Testing**
Data Subset	KDD-R	KDD-T	KDD-T	KDD-R
Number of Not-R2L Records	1073988	294720	292987	1060386
Not-R2L Record Detection Rate	99.99%	99.96%	98.79%	98.73%
Number of R2L Records	986	876	11859	43
R2L Record Detection Rate	98.70%	5.41%	73.25%	4.30%

Table 2: Comparison of C4.5 Decision Tree Performance for the R2L Attack Category

Tables 1 and 2 indicate that most of the non-attack records are detected with high probability. This result is expected because there are a large number of records in this class. For attack records, it can be seen that there are large variations between the performances of the two models. For example, Model 1 for U2R detects 76.92% of attack records in the training data subset and detects only 6.14% in the testing data subset. Model 2 for the U2R category detects 97.37% of attack records in the training data subset and detects only 44.23% in the testing data subset. A similar observation can be made for the R2L category. These results show that the model built from the KDD training data subset fails when tested on the KDD testing data subset. Similarly, the model built on the KDD testing data subset fails when tested with the KDD training data subset. This failure happens only when the attack-specific information or signature present in training and testing data subsets represent sufficiently dissimilar hypotheses for the target concept or function to be learned. If a classifier model shows promising results for training data, it should perform reasonably well on testing data if training and testing data subsets represent reasonably similar hypotheses.

When performance of the two models, Model 1 and Model 2, are evaluated within the same context, the results suggest that the U2R and R2L attack signatures in the KDD training and testing data subsets are not similar. This empirical finding is also justified by the way the KDD data set was created. One of the features of the KDD data set is that the testing data subset consists of many U2R and R2L attack signatures that are not present in the KDD training data subset. This feature exists to facilitate performance assessment of anomaly detection algorithms. However this feature is so prominently established in the KDD data set that traditional pattern recognition and machine learning algorithms are provided with diminished and (highly) distorted attack signatures in the training data subset, which fails to offer a reasonable framework for training. A fairly representative class samples are required to exist in the training data subset for pattern recognition and machine learning algorithms to achieve acceptable classification accuracy on the testing data subset.

4.2 Performance of Classifiers on Merged Data Set Using A Cross-Validation Approach

Recognizing the lack of similarity between attack signatures in the KDD training and testing data subsets, reported in Section 4.1, a new experiment was designed to further validate this empirical finding. In this experiment, the KDD training and testing data subsets were merged into one data set. This merged data set was partitioned into five subsets such that each subset contained approximately equal number of records, randomly selected, from different classes of the merged data set.

Five-fold cross-validation approach was applied to test the performance of the classifier algorithms on the merged data set. To build classifier models for the U2R attack category, the records were labeled as either U2R or Not U2R. The KDD training and testing data subsets were merged together and then re-sampled into five equal sized sets (called cross-validation folds). These folds had almost equal numbers of records from the U2R and the Not U2R classes. Models were then built using any four of the five folds of data and tested on the fifth fold. This means that the training data had 80% of records from the merged KDD data set and the testing data had 20%. Five sets of results, corresponding to the five folds, were obtained for each classifier algorithm. The same approach was used to create R2L classifier models. The motivation of using a cross-validation methodology was to determine whether or not 80% of the merged KDD data set (randomly sampled) contains sufficient information about the U2R and R2L attack categories to build high performance classifier models.

Three algorithms including a trainable neural network classifier (Neural), which is a multi layer perceptron network[18], a probabilistic classifier (Gaussian) [19], and a decision tree classifier (Binary Tree) [20] were tested using the cross-validation approach. These three algorithms represent a diverse set of algorithmic approaches to pattern classification. All three algorithms, as implemented in the LNKnet [21] pattern classification software, were applied to build classifiers for the U2R and R2L attack categories. The Binary Tree algorithm ensures that the classifier models created are not under or over trained, as could be the case with a multi layer perceptron network. Both the Neural and Gaussian classification algorithms have limitations with respect to the KDD data set. As there was no validation data subset, there was no precise stopping criterion for the Neural classifier. Due to uneven distributions of patterns in various attack classes, the Gaussian classifier might not be able to accurately model the KDD attack categories in the KDD data set. Neither of these limitations is applicable to the Binary Tree classifier algorithm.

For each classification algorithm, five-fold testing was used. Five models were built, each having one distinct fold for testing. For the Neural classifier, a three layer feed-forward neural network topology was instantiated. The unipolar sigmoidal transfer function with slope value of 1.0 was utilized for every neuron in both the hidden and the output layers. The learning algorithm was stochastic gradient descent with a mean squared error function. There were 41 neurons in the input layer (i.e., 41 features in the input pattern), 60 neurons in the hidden layer (an empirically determined value), and 2 neurons in the output layer (2 class detection). A constant learning rate of 0.1 along with a weight change momentum of 0.6 was applied. Randomly selected initial weights were used; they were uniformly distributed in the range [-0.1, 0.1]. Each epoch consisted of 200,000 patterns randomly selected from the training data subset. Gaussian classifier was built using a quadratic classifier having a separate tilted covariance matrix for each class to be detected. For the Binary Tree algorithm, a single feature was used for splitting the data subset among child nodes. The tree was expanded until no errors were found on the training data subset. No pruning of the decision tree was performed and the complete binary decision tree was used for testing.

For U2R attack detection, each fold consisted of 277,148 patterns not belonging to the U2R category and 56 patterns belonging to the U2R category. For the R2L attack category, each fold consisted of 273,766 patterns not belonging to the R2L category and 3,438 patterns belonging to the R2L category. Table 3 shows the result obtained by using these three classifier models for the U2R and Not U2R categories. These results present the average number of records detected in the testing data subset for each of the five models. Standard deviation values for the average values are shown in parentheses. For the R2L category, Table 4 shows equivalent results obtained for the three algorithms tested.

As shown in Table 3, the U2R class detection increased to almost 90% for any of the three algorithms tested. Similarly for the R2L attack category (Table 4), the detection rate was more than 97%, which represents highly superior performance. Gaussian classifier performed the best with 96.43% for the U2R category. For the R2L category, both the Neural and the Binary Tree algorithms performed well, with more than 99% detection rate. Little deviation was presented in the results (as shown in parentheses), when any of the five folds were used as testing, which suggests that the class patterns in each fold represent similar target hypotheses.

	No of Records Detected		**% of Records Detected**	
	Not-U2R	**U2R**	**Not-U2R**	**U2R**
Neural	276906 (172.04)	50 (2.59)	99.91 (0.039)	89.28 (4.622)
Gaussian	272534 (568.06)	54 (1.14)	98.34 (0.206)	96.43 (2.036)
Binary Tree	277057 (13.09)	49 (3.21)	99.96 (0.006)	87.50 (5.731)

Table 3: Cross Validation Results for the U2R Category: Mean and Standard Deviation Values

	No of Records Detected		**% of Records Detected**	
	Not-R2L	**R2L**	**Not-R2L**	**R2L**
Neural	271126 (101.745)	3419 (5.385)	99.04 (0.415)	99.44 (0.177)
Gaussian	266253 (449.215)	3368 (4.930)	97.26 (0.165)	97.96 (0.151)
Binary Tree	272174 (51.885)	3410 (4.930)	99.42 (0.882)	99.18 (0.168)

Table 4: Cross Validation Results for the R2L Category: Mean and Standard Deviation Values

These results demonstrate that classifier models trained using any four folds acquire sufficient information to achieve high detection rates if the fifth fold is employed for testing. Algorithms tested in the literature were able to achieve only approximately a detection rate of 10 to 20% for the U2R and the R2L attack categories. If KDD training and testing data subsets are merged and re-sampled with 80% of records in the training data subset and 20% of records in the testing data subset, the detection rates for the U2R and R2L attack categories rise to 90%. This clearly indicates that the original KDD training and testing data subsets represent dissimilar target hypotheses for the U2R and the R2L attack categories.

4.3 Qualitative Comparison of Rule Sets Induced from the KDD Training and Testing Data Subsets through C4.5

In this section, the decision trees, or equivalently the rule sets, induced by the C4.5 algorithm separately on the KDD training and testing data subsets are qualitatively compared for U2R and R2L attack categories. Observations are made in terms of features utilized, how many records are classified using the C4.5 rules, and whether the knowledge extracted from the training and the testing data subsets is "similar", i.e., whether the two data subsets represent similar target hypotheses.

The C4.5 decision tree algorithm creates rules based on the training data. Two decision trees or rule sets were created using the C4.5 algorithm, one using the KDD training data subset and the other using the KDD testing data subset. These two rule sets are compared qualitatively to assess "similarity" between them for both the U2R and the R2L attack categories.

All records in the KDD training and testing data subsets were relabeled as either belonging or not belonging to the U2R/R2L attack class. Table 5 shows the significant rules (which detect more than 80% of the U2R attack records) generated for the U2R attack category. Significant rules, which detect more than 85% of the R2L attack records, are presented in Table 6.

Table 5 indicates that for the U2R category, 24 out of 41 features were used by rules created using the KDD training and testing data subsets. Among these 24 features, many exist in either the rules created using the training data or the rules induced from testing data, but not both. There were a total of nine features (*logged_in*, *num_root*, *num_file_creations*, *num_compromised*, *num_access_files*, *count*, *same_srv_rate*, *dst_host_srv_diff_host_rate*, and *dst_host_same_srv_rate*) only present in the rules created using the KDD training data subset and not in those rules created using the KDD testing data subset. These features are not used by the rules created from the KDD testing data subset. Again there were a total of nine features (*is_hot_login*, *is_guest_login*, *dst_host_diff_srv_rate*, *flag*, *srv_count*, *dst_host_rerror_rate*, *rerror_rate*, *urgent*, and *dst_host_srv_rerror_rate*) only present in the rules induced from the KDD testing data subset. Hence out of 24 features used by rules created using both KDD data subsets, only six features (*dst_host_same_srv_port_rate*, *service*, *src_bytes*, *dst_bytes*, *hot*, and *root_shell*) were common to both sets of rules.

Similar observations are applicable for the R2L attack category. Table 6 indicates that for R2L attack category, a total of 17 out of 41 features were used by rules created using the KDD training and testing data subsets. Five features (*hot*, *dst_host_same_srv_port_rate*, *logged_in*, *dst_host_srv_diff_host_rate*, and *num_failed_login*) were only present in the rules created through the KDD training data subset and were not present in rules induced from the KDD testing data subset. Similarly, six features (*is_guest_login*, *count*, *dst_host_same_srv_rate*, *srv_count*, *srv_diff_host_rate*, and *dst_host_srv_count*) were only present in rules created by the

KDD testing data subset. The remaining six features (*duration, service, src_bytes, dst_bytes, same_srv_rate*, and *dst_host_diff_srv_rate*) were common for both rule sets.

Tables 5 and 6 suggest that the included feature sets from the KDD training and testing data subsets are not similar. Tables 5 and 6 also indicate that for the features that are common to both sets of rules created using the KDD training and testing data subsets, the range of values for which the rules assume the logic truth value is not similar. Although the features are common, the rules are applicable in different ranges leading to a conclusion that rule sets represent different knowledge from the domain. Hence, for the U2R and the R2L attack categories, any classifier algorithm trained on the KDD training (testing) data subset will not be able to detect a significant set of attacks in the KDD testing (training) data subset. This observation, once more, strongly suggests and validates that the two data subsets, the KDD training and testing, do not represent similar target hypotheses for the U2R and the R2L attack categories.

5. Conclusions

This paper demonstrated that the KDD training and testing data subsets represent dissimilar target hypotheses for the U2R and the R2L attack categories. In light of the extensive analyses performed in Section 4, it is reasonable to conclude that it is not possible for any trainable pattern classification or machine learning algorithm to demonstrate an acceptable level of misuse detection performance on the KDD testing data subset if classifier models are built using the KDD training data subset for the U2R and the R2L attack categories. Therefore, researchers are advised not to employ the KDD training and testing data subsets in this context. The KDD data set, on the other hand, might be attractive for anomaly detection algorithms since the testing data has substantial new attacks with signatures that are not correlated with similar attacks in the training data.

It is possible to generalize the findings in this paper by stating that any training and testing data subset pair can be validated for representing similar target hypotheses through the methodology proposed in this paper. This should be done in the event substantial attempts to develop classifiers or function approximators through a suite of trainable pattern classification or machine learning algorithms fail.

Acknowledgements

We gratefully acknowledge Mr. Alan Husted's valuable contribution in facilitating access to the computing resources required for this research project. This manuscript has been significantly improved in style through substantial efforts of anonymous reviewers.

References

1. R. Agarwal, and M. V. Joshi, PNrule: A New Framework for Learning Classifier Models in Data Mining (A Case-Study in Network Intrusion Detection), IBM Research Division Technical Report No. RC-21719, 2000.
2. I. Levin, KDD-99 Classifier Learning Contest LLSoft's Results Overview, *ACM SIGKDD Explorations* **1(2)** (2000), 67-75.
3. C. Elkan, Results of the KDD'99 Classifier Learning, *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Boston, MA* **1(2)** (2000), 63-64.
4. L. Ertoz, M. Steinbach, and V. Kumar, Finding Clusters of Different Sizes, Shapes, and Densities in Noisy, High Dimensional Data, Technical Report, 2002.
5. W. Fan, M. Miller, S. Stolfo , W. Lee, and P. Chan, Using Artificial Anomalies to Detect Unknown and Known Network Intrusions, *IEEE International Conference on Data Mining, San Jose, CA* (2001), 123-130.
6. J. Gomez, and D. Dasgupta, Evolving Fuzzy Classifiers for Intrusion Detection, *In Proceedings of the IEEE Workshop on Information Assurance, United States Military Academy, West Point, NY* (2001) 68-75.
7. M. V. Joshi, R. C. Agarwal, and V. Kumar, Mining Needles in a Haystack: Classifying Rare Classes via Two-Phase Rule Induction, *ACM SIGMOD Conference on Management of Data, Santa Barbara, CA* (2001), 91-102.
8. W. Lee, and S. Stolfo, A Framework for Constructing Features and Models for Intrusion Detection Systems, *ACM Transactions on Information and System Security*, **3** (2000), 227-261.

9. D. Y. Yeung, and C. Chow, Parzen-window Network Intrusion Detectors, *In Proceedings of the Sixteenth International Conference on Pattern Recognition, Quebec City, Canada* (2002), 385-388.
10. DARPA 1998 data set, http://www.ll.mit.edu/IST/ideval/data/1998/1998_data_index.html, cited August 2003.
11. W. Lee, S. J. Stolfo, and K. W. Mok, A Data Mining Framework for Building Intrusion Detection Models, *IEEE Symposium on Security and Privacy, Oakland, California* (1999), 120-132.
12. KDD 1999 data set, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, cited August 2003.
13. W. Lee, S. J. Stolfo, and K. W. Mok, Mining in a Data-Flow Environment: Experience in Network Intrusion Detection, *In Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA* (1999), 114-124.
14. B. Pfahringer, Winner of KDD'99 classifier learning contest, Australian Research Institute for Artificial Intelligence, http://www.ai.univie.ac.at/~bernhard/kddcup99.html, cited August 2003.
15. M. R. Sabhnani, and G. Serpen, Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context, *Proceedings of International Conference on Machine Learning: Models, Technologies, and Applications, Las Vegas, Nevada* (2003), 209-215.
16. J. R. Quinlan, C4.5: Program for Machine Learning, Morgan Kaufmann Publishing, 1992.
17. C4.5 Simulator, Developer Site: http://www.rulequest.com; Download code from: http://www.cs.uregina.ca/~dbd/cs831/notes/ml/dtrees/c4.5/tutorial.html, cited August 2003.
18. P. J. Werbos, Beyond Regression: New Tools for Prediction and Analysis in the Behavioural Sciences, Ph.D. dissertation, Committee on Applied Mathematics, Harvard University, Cambridge, MA (1974).
19. R. O. Duda, and P. E. Hart, Pattern Classification and Scene Analysis, New York: Wiley, 1973.
20. L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, Classification and Regression Trees, Belmont, California, Wadsworth, Inc., 1984.
21. LNKnet software, http://www.ll.mit.edu/IST/lnknet/index.html, cited August 2003.
22. T. Mitchell, Machine Learning, McGraw-Hill series in computer science, 1997.
23. E. Parzen, On Estimation of A Probability Density and Mode, *Annals of Mathematical Statistics*, **35** (1962), 1065-1076.

(service = telnet ∧ root_shell = 1 ∧
hot > 2 ∧ num_root <= 5)
➔ A U2R attack

(service = ftp_data ∧ logged_in = 1 ∧
dst_bytes > 2931 ∧
dst_host_srv_diff_host_rate <= 0.03)
➔ A U2R attack

(service = ftp_data ∧ logged_in = 1 ∧
dst_bytes > 2931 ∧ count > 1)
➔ A U2R attack

(src_bytes <= 533 ∧ num_file_creations > 0 ∧
num_access_files <= 0 ∧
dst_host_same_srv_port_rate > 0.11)
➔ A U2R attack

(dst_bytes <= 11972 ∧ root_shell = 1 ∧
hot <= 2 ∧ num_root > 1 ∧
num_file_creations > 0)
➔ A U2R attack

(num_file_creations > 0 ∧
same_srv_rate <= 0.41 ∧
dst_host_same_srv_rate > 0.89)
➔ A U2R attack

(num_compromised > 1 ∧
root_shell = 0 ∧ num_root > 0 ∧
dst_host_same_srv_port_rate > 0.13 ∧
dst_host_srv_diff_host_rate <=0.33)
➔ A U2R attack

(a) Training Data Subset

(service = other ∧ flag = REJ ∧
rerror_rate > 0.99 ∧ dst_host_rerror_rate <=0.19)
➔ A U2R attack

(service = telnet ∧ hot > 0 ∧ root_shell = 1 ∧
is_hot_login = 0 ∧
dst_host_same_srv_port_rate <= 0.02)
➔ A U2R attack

(service = other ∧ flag = SF ∧
dst_host_srv_rerror_rate > 0.61)
➔ A U2R attack

(srv_count <= 1 ∧ dst_host_diff_srv_rate <= 0.02 ∧
dst_host_rerror_rate <= 0.12 ∧
dst_host_srv_rerror_rate > 0.61)
➔ A U2R attack

(urgent > 0 ∧ root_shell = 1)
➔ A U2R attack

(service = ftp ∧ is_guest_login = 0 ∧
src_bytes <= 2203 ∧ dst_bytes > 195 ∧
is_hot_login = 0)
➔ A U2R attack

(service = ftp_data ∧ dst_bytes > 195)
➔ A U2R attack

(b) Testing Data Subset

Table 5: Significant C4.5 Rules to Detect U2R Attack Records in the KDD Data Set

(dst_bytes <= 3299 ∧ hot > 25)
➔ A R2L attack

(service = ftp_data ∧ src_bytes > 333 ∧
src_bytes <= 334)
➔ A R2L attack

(service = ftp_data ∧ same_srv_rate > 0.87 ∧
dst_host_same_srv_port_rate > 0.99 ∧
dst_host_srv_diff_host_rate > 0.11)
➔ A R2L attack

(logged_in = 1 ∧ duration > 3 ∧
src_bytes > 333 ∧ dst_bytes <= 63 ∧
dst_host_same_srv_port_rate > 0.99)
➔ A R2L attack

(num_failed_logins > 0 ∧
dst_host_diff_srv_rate <= 0)
➔ A R2L attack

(a) Training Data Subset

(service = private ∧ duration <= 2 ∧
src_bytes > 40 ∧ src_bytes <= 104 ∧
count <= 5)
➔ A R2L attack

(service = pop_3 ∧ src_bytes > 11 ∧
src_bytes <= 37)
➔ A R2L attack

(duration > 88 ∧ is_guest_login = 1)
➔ A R2L attack

(duration > 144 ∧ service = ftp_data)
➔ A R2L attack

(service = ftp_data ∧ src_bytes > 11 ∧
src_bytes <= 14 ∧ dst_host_srv_count <= 34)
➔ A R2L attack

(service = private ∧ dst_bytes > 125 ∧
srv_count <= 1 ∧ dst_host_srv_count <= 252 ∧
dst_host_diff_srv_rate <= 0.01)
➔ A R2L attack

(duration <= 2 ∧ src_bytes > 11 ∧
src_bytes <= 133 ∧ dst_bytes > 125 ∧
srv_count <= 3 ∧
dst_host_same_srv_rate <= 0.99)
➔ A R2L attack

(src_bytes > 11 ∧ src_bytes <= 133 ∧
dst_bytes > 52 ∧ srv_count <= 2 ∧
same_srv_rate > 0.58 ∧ srv_diff_host_rate <= 0.01 ∧
dst_host_diff_srv_rate > 0 ∧
dst_host_same_srv_port_rate <= 0)
➔ A R2L attack

(b) Testing Data Subset

Table 6: Significant C4.5 Rules to Detect R2L Attack Records in the KDD Data Set