Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context

Maheshkumar Sabhnani
EECS Dept, University of Toledo
Toledo, Ohio 43606 USA

Gursel Serpen
EECS Dept, University of Toledo
Toledo, Ohio 43606 USA

Abstract

A small subset of machine learning algorithms, mostly inductive learning based, applied to the KDD 1999 Cup intrusion detection dataset resulted in dismal performance for user-to-root and remote-to-local attack categories as reported in the recent literature. The uncertainty to explore if other machine learning algorithms can demonstrate better performance compared to the ones already employed constitutes the motivation for the study reported herein. Specifically, exploration of if certain algorithms perform better for certain attack classes and consequently, if a multi-expert classifier design can deliver desired performance measure is of high interest. This paper evaluates performance of a comprehensive set of pattern recognition and machine learning algorithms on four attack categories as found in the KDD 1999 Cup intrusion detection dataset. Results of simulation study implemented to that effect indicated that certain classification algorithms perform better for certain attack categories: a specific algorithm specialized for a given attack category . Consequently, a multi-classifier model, where a specific detection algorithm is associated with an attack category for which it is the most promising, was built. Empirical results obtained through simulation indicate that noticeable performance improvement was achieved for probing, denial of service, and user-to-root attacks.

Keywords: Computer security, Machine learning, Intrusion detection, misuse, KDD dataset, Probability of detection, False alarm rate

1. Introduction

Literature survey indicates that, for intrusion detection, most researchers employed a single algorithm to detect multiple attack categories with dismal performance in some cases. The set of machine learning algorithms applied in the literature constitutes a very small subset of what is potentially applicable for the intrusion detection problem. Additionally, reported results suggest that much detection performance improvement is possible. In light of the widely-held belief that attack execution dynamics and signatures show substantial variation from one attack category to another, identifying attack category specific detection algorithms offers a promising research direction for improved intrusion detection performance. In this paper, a comprehensive set of pattern recognition and machine learning algorithms will be evaluated on the KDD data set [1], which is one of few public domain such data, utilizes TCP/IP level information and embedded with domain-specific heuristics, to detect intrusions at the network level.

KDD dataset covers four major categories of attacks: Probing attacks (information gathering attacks), Denial-of-Service (DoS) attacks (deny legitimate requests to a system), user-to-root (U2R) attacks (unauthorized access to local super-user or root), and remote-to-local (R2L) attacks (unauthorized local access from a remote machine). KDD dataset is divided into labeled and unlabeled records. Each labeled record consisted of 41 attributes (features) [2] and one target value. Target value indicated the attack category name. There are around 5 million (4,898,430) records in the labeled dataset, which was used for training all classifier models discussed in this paper. A second unlabeled dataset (311,029 records) is provided as testing data [3]. Next, a brief and up-to-date literature survey of attempts for designing intrusion detection systems using the KDD dataset is presented.

Agarwal and Joshi [4] proposed a two-stage general-to-specific framework for learning a rule-based model (PNrule) to learn classifier models on a data set that has widely different class distributions in the training data. The PNrule technique was evaluated on the KDD testing data set, which contained many new R2L attacks not present in the KDD training dataset. The proposed model was able to detect 73.2% of probing attacks, 96.9% of denial of service attacks, 6.6% of U2R attacks, and 10.7% of attacks in R2L attack category. False alarms were generated at a level of less than 10% for all attack categories except for U2R: an unacceptably high level of 89.5% false alarm rate was reported for the U2R category.

Using Kernel Miner tool and the KDD data set [5], Levin created a set of locally optimal decision trees (called the decision forest) from which optimal subset of trees (called the sub-forest) was selected for predicting new cases. Levin used only 10% of the KDD training data randomly sampled from the entire training data set. Multi-class detection approach was used to detect different attack categories in the KDD data set. The final trees gave very high detection rates for all classes including the R2L in the entire training data set. The proposed classifier achieved 84.5% detection for probing, 97.5% detection for denial of service, 11.8% detection for U2R, and only 7.32% detection for R2L attack category in the KDD testing data set. False alarm rates of 21.6%, 73.1%, 36.4%, and 1.7% were achieved for probing, DoS, U2R and R2L attack categories, respectively.

Ertoz and Steinbach [6] used shared nearest neighbor (SNN) technique, which is particularly suited for finding clusters in data of different sizes, density, and shapes, mainly when the data contains large amount of noise and outliers. All attack records were selected from the KDD training and testing data sets with a cap of 10,000 records from each attack type: there are a total of 36 attack types from 4 attack categories. Also 10,000 records were randomly picked from both the training and the testing data sets. In total, around 97,000 records were selected from the entire KDD data set. After removing duplicate KDD records, the data set size reduced to 45,000 records. This set was then used to train two clustering algorithms: *K*-means and the proposed SNN technique, which were compared in terms of detection rates achieved. *K*-means algorithm, with number of clusters equal to 300, could detect 91.88% of probing, 97.85% of DoS, 5.6% of U2R, and 77.04% of the R2L attack records. The proposed SNN technique was able to detect 73.43% of probing, 77.76% of DoS, 37.82% of U2R, and 68.15% of the R2L records, while noting that no discussion on false alarm rates were reported by the authors. Results suggested that SNN performed better than *K*-means for U2R attack category. It is important to note that there was no independent testing data used. Also the results are evaluated on a relatively small portion of KDD dataset and not on complete KDD testing dataset. Hence the reported results show the performance of both algorithms on the training data set only, which is expected to result in high detection rates by default.

Yeung and Chow [7] proposed a novelty detection approach using non-parametric density estimation based on Parzen-window estimators with Gaussian kernels to build an intrusion detection system using normal data only. This novelty detection approach was employed to detect attack categories in the KDD data set. 30,000 randomly sampled normal records from the KDD training data set were used as training data set to estimate the density of the model. Another 30,000 randomly sampled normal records (also from the KDD training data set) formed the threshold determination set, which had no overlap with the training data set. The technique could detect 99.17% of probing, 96.71% of DoS, 93.57% of U2R, and 31.17% of R2L attacks in the KDD testing dataset as intrusive patterns: authors did not report any information on false alarm rates. As a significant limitation, it is important to note that this model detects whether a record is intrusive or not and not if the attack records belongs to a specific attack category.

Literature survey shows that, for all practical purposes, most researchers applied a single algorithm to address all four major attack categories. It is critical to question this approach that strives to identify a single algorithm that can detect attacks in all four attack categories. These four attack categories including DoS, Probing, R2L, and U2R, have distinctly unique execution dynamics and signatures [8], which motivates to explore if in fact certain, but not all, detection algorithms are likely to demonstrate superior performance for a given attack category. In light of this possibility, the study presented here will create detection models using a comprehensive set of pattern recognition and machine learning algorithms and select best performing (in terms of probability of detection and false alarm rates) algorithms for each attack category.

In the event that certain subset of detection algorithms do offer improved probability of detection coupled with false alarm rates for a specific attack category, a multi-classifier system will be attempted to improve the overall detection performance on the four attack categories as they exist in the KDD data sets.

2. Simulation study

This section will elaborate the methodology employed to test various algorithms on the KDD datasets. First required data preprocessing and tools employed will be discussed in Section 2.1. Cost matrix measure, which is employed to select best instances of a given classifier algorithm, will be presented in the following section. Parameter settings for various models will be described in Section 2.3. Results of testing classifier algorithms will be discussed in Section 2.4.

2.1. Data preprocessing and simulation tools

Attributes in the KDD datasets had all forms - continuous, discrete, and symbolic, with significantly varying resolution and ranges. Most pattern classification methods are not able to process data in such a format. Hence preprocessing was required before pattern classification models could be built. Preprocessing consisted of two steps: first step involved mapping symbolic-valued attributes to numeric-valued attributes

and second step implemented scaling. Attack names (like buffer_overflow, guess_passwd, etc.) were first mapped to one of the five classes, 0 for Normal, 1 for Probe, 2 for DoS, 3 for U2R, and 4 for R2L, as described in [9]. Symbolic features like protocol_type (3 different symbols), service (70 different symbols), and flag (11 different symbols) were mapped to integer values ranging from 0 to *N*-1 where *N* is the number of symbols. Then each of these features was linearly scaled to the range [0.0, 1.0]. Features having smaller integer value ranges like duration [0, 58329], wrong_fragment [0, 3], urgent [0, 14], hot [0, 101], num_failed_logins [0, 5], num_compromised [0, 9], su_attempted [0, 2], num_root [0, 7468], num_file_creations [0, 100], num_shells [0, 5], num_access_files [0, 9], count [0, 511], srv_count [0, 511], dst_host_count [0, 255], and dst_host_srv_count [0, 255] were also scaled linearly to the range [0.0, 1.0]. Two features spanned over a very large integer range, namely src_bytes [0, 1.3 billion] and dst_bytes [0, 1.3 billion]. Logarithmic scaling (with base 10) was applied to these features to reduce the range to [0.0, 9.14]. All other features were either boolean, like logged_in, having values (0 or 1), or continuous, like diff_srv_rate, in the range [0.0, 1.0]. Hence scaling was not necessary for these attributes.

The KDD 1999 Cup dataset has a very large number of duplicate records. For the purpose of training different classifier models, these duplicates were removed from the datasets. The total number of records in the original labeled training dataset is 972,780 for Normal, 41,102 for Probe, 3,883,370 for DoS, 52 for U2R, and 1,126 for R2L attack classes. After filtering out the duplicate records, there were a total of 812,813 records for Normal, 13,860 for Probe, 247,267 for DoS, 52 for U2R, and 999 for R2L attack classes.

The software tool LNKnet, which is a publicly available pattern classification software package [10], was used to simulate pattern recognition and machine learning models. The only exception was for developing decision tree classifier models. The C4.5 algorithm was employed to generate decision trees using the software tool obtained at [11]. All simulations were performed on a multi-user Sun SPARC™ machine, which had dual microprocessors, UltraSPARC-II, running at 400 MHz. System clock frequency was equal to 100 MHz. The system had 512 MB of RAM and Solaris™ 8 operating system.

2.2. Performance comparison measures

Identifying an optimal set of settings for the topology and parameters for a given classifier algorithm through empirical means required multiple instances of detection models to be built and tested on the KDD datasets. For instance, more than 50 models were developed using the multilayer perceptron algorithm alone. Hence a comparative measure is needed to select the best model for a given classifier algorithm. One such measure is the cost per example that requires two quantities to be defined: cost matrix and confusion matrix.

A cost matrix (*C*) is defined by associating classes as labels for the rows and columns of a square matrix: in the current context for the KDD dataset, there are five classes, {Normal, Probe, DoS, U2R, R2L}, and therefore the matrix has dimensions of 5×5. An entry at row *i* and column *j*, *C(i,j),* represents the non-negative cost of misclassifying a pattern belonging to class *i* into class *j*. Cost matrix values employed for the KDD dataset are defined elsewhere in [9]. These values were also used for evaluating results of the KDD'99 competition. The magnitude of these values was directly proportional to the impact on the computing platform under attack if a test record was placed in a wrong category.

A confusion matrix (*CM*) is similarly defined in that row and column labels are class names: a 5×5 matrix for the KDD dataset. An entry at row *i* and column *j*, *CM(i,j),* represents the number of misclassified patterns, which originally belong to class *i* yet mistakenly identified as a member of class *j*.

Given the cost matrix as predefined in [9] and the confusion matrix obtained subsequent to an empirical testing process, cost per example (*CPE*) was calculated using the formula,

$$CPE = \frac{1}{N}\sum_{i=1}^{5}\sum_{j=1}^{5} CM(i,j) * C(i,j)$$

where *CM* corresponds to confusion matrix, *C* corresponds to the cost matrix, and *N* represents the number of patterns tested. A lower value for the cost per example indicates a better classifier model.

Comparing performances of classifiers for a given attack category is implemented through the probability of detection along with the false alarm rate, which are widely accepted as standard measures.

2.3. Pattern recognition and machine learning algorithms applied to intrusion detection

Nine distinct pattern recognition and machine learning algorithms were tested on the KDD dataset. These algorithms were selected so that they represent a wide variety of fields: neural networks, probabilistic models, statistical models, fuzzy-neuro systems, and decision trees. An overview of how specific instances of these algorithms were identified as well as their intrusion detection performance on the KDD testing data set follows next.

2.3.1. Multilayer perceptron (MLP). Multilayer perceptron (MLP) [12] is one of most commonly used neural network classification algorithms.

The architecture used for the MLP during simulations with KDD dataset consisted of a three layer feed-forward neural network: one input, one hidden, and one output layers. Unipolar sigmoid transfer functions were used for each neuron in both the hidden and the output layers with slope value of 1.0. The learning algorithm used was stochastic gradient descent with mean squared error function. There were a total of 41 neurons in the input layer (41-feature input pattern), and 5 neurons (one for each class) in the output layer. Multiple simulations were performed with number of hidden layer nodes varying from 40 to 80 in increments of 10. Also for each simulation a constant learning rate (one of the four values 0.1,0 0.2, 0.3, and 0.4) was used along with 0.6 as the weight change momentum value. Different simulations had different learning rates that varied from 0.1 to 0.4 in steps of 0.1. Randomly selected initial weights were used that were uniformly distributed in the range [-0.1, 0.1]. Each epoch consisted of 500,000 samples having equi-probable records from each of the five output categories. Initially a total of 30 epochs were performed on the training dataset. Other simulations studied the effect of changing the number of training epochs: number of epochs was varied to 40, 50, 60, 100, etc. The final model, which scored the lowest cost per example value of 0.2393, consisted of 50 nodes in the hidden layer with a learning rate value equal to 0.1 and 60 epochs for training.

2.3.2. Gaussian classifier (GAU). Maximum likelihood Gaussian classifiers assume inputs are uncorrelated and distributions for different classes differ only in mean values. Gaussian classifier is based on the Bayes decision theorem [13].

Four distinct models were developed using the Gaussian classifier: quadratic classifier with diagonal covariance matrix, quadratic classifier with tilted covariance matrix, linear classifier with diagonal covariance matrix, and linear classifier with tilted covariance matrix. Linear discriminant classifier with full tilted matrix performed the best on the KDD testing dataset with cost per example value of 0.3622.

2.3.3. *K*-means clustering (K-M). *K*-means clustering algorithm [13] positions *K* centers in the pattern space such that the total squared error distance between each training pattern and the nearest center is minimized.

Using the *K*-means clustering algorithm, different clusters were specified and generated for each output class. Simulations were run having 2, 4, 8, 16, 32, 40, 64, 75, 90, 110, 128, and 256 clusters. Each simulation had equal number of clusters for each attack class. For number of clusters (*K*) that are not integer powers of 2, after generating *P* clusters (*P* being an integer power of 2) where $P>K$, the cluster centers having minimum variance among its patterns were removed one at a time until the clusters were reduced to *K*. An epoch consisted of presenting all training patterns in an output class for which centers are being generated. Clusters were trained until the average squared error difference between two epochs was less than 1%. During splitting, the centers were disturbed by $\mp 1\%$ of the standard deviation in each cluster so that new clusters are formed. A random offset of $\mp =1\%$ was also added during each split. The model that achieved the lowest cost per example value (0.2389) had 16 clusters in each class.

2.3.4. Nearest cluster algorithm (NEA). Nearest cluster algorithm [13] is a condensed version of *K*-nearest neighbor clustering algorithm. Input to this algorithm is a set of cluster centers generated from the training data set using standard clustering algorithms like *K*-means, E & M binary split, and leader algorithm.

Initial clusters were created using the *K*-means. Multiple simulations were performed using different set of initial clusters in each output class including 2, 4, 8, 16, 32, 40, 64, 75, 90, 110, 128, and 256 clusters using the *K*-means algorithm. Euclidean distance was used as the distance measure. Based on the minimum cost per test example, the nearest cluster model that performed the best had 32 clusters in each class during training. The model's cost per example for the KDD test data was equal to 0.2466.

2.3.5. Incremental radial basis function (IRBF). Incremental radial basis function (IRBF) neural networks [14] can also perform nonlinear mapping between input and output vectors similar to an MLP.

A total of six simulations were performed using the IRBF algorithm. Each simulation used initial clusters created using *K*-means algorithm: there were 8, 16, 32, 40, 64, and 75 clusters each in different output classes. Learning rate for change in weights was 0.1 and those for changes in hidden unit means and variance were 0.01. A separate diagonal covariance matrix was used for each hidden unit with minimum variance of 0.000001. A total of 10 epochs were performed on the training data. During each epoch, 500,000 records were randomly and uniformly sampled from each attack class. Mean squared error cost function was used for these simulations. Initial weights were randomly selected from a uniform distribution in the range of [-0.01, 0.01]. For each simulation using the IRBF, cost per example for the test dataset was calculated. The model with 40 hidden nodes for each class performed best with the cost per example value equal to 0.4164.

2.3.6. Leader algorithm (LEA). Leader algorithm [15] partitions a set of M records into K disjoint clusters (where $M \geq K$). A rule for computing distance measure between records is assumed along with a threshold value, *delta*. First input record forms the leader of first cluster. Each input record is sequentially compared with current leader clusters. If the distance measure between the current record and all leader records is greater than *delta*, a new cluster is formed with the current record being the cluster leader.

Different clusters were created for different output classes. Multiple simulations were executed with distance threshold value, *delta*, assuming values from the set {0.05, 0.2, 0.4, 0.6, 0.8, 1, 5, 10, 50, 70, 90, and 100}. Records from each class were randomly presented during training. As the value of *delta* was increased, the number of clusters in each output class kept reducing. Only one cluster per class was formed when *delta* was equal to 100. It was observed that the results were not reproducible as leader records for each cluster were dependent on the sequence in which the records were presented. The results varied tremendously for larger values of *delta*. Cost per example for test data was calculated for each simulation. The model that performed best had *delta* value equal to 0.05 and cost per example value equal to 0.2528.

2.3.7. Hypersphere algorithm (HYP). Hypersphere algorithm [16] [17] creates decision boundaries using spheres in input feature space. Any pattern that falls within the sphere is classified in the same class as that of the center pattern. Spheres are created using an initial defined radius. Euclidean distance between a pattern and sphere centers is used to test whether a pattern falls in one of the current defined spheres.

Multiple simulations were performed with different initial radius, e.g. 0.05, 0.2, 0.4, 0.5, 0.6, 0.8, 1.0, and 20.0. The total number of epochs performed was equal to 30. All patterns present in the training dataset were used during each epoch. Single nearest neighbor classification was used to classify patterns that were ambiguous or outside all hyperspheres created. A total of 1066 hyperspheres were formed after 30 epochs in all categories during training. The Hypersphere model that performed the best had initial radius equal to 0.05 and cost per example value equal to 0.2486.

2.3.8. Fuzzy ARTMAP (ART). Fuzzy ARTMAP (Adaptive Resonance Theory mapping) [18] algorithm is used for supervised learning of multidimensional data.

Fuzzy ARTMAP uses two ART's – ARTa and ARTb. ARTa maps features into clusters. ARTb maps output categories into clusters. There is a mapping from ARTa clusters to ARTb clusters which is performed during training. A constant vigilance value (distance measure) of 0.999 was used for ARTb in all simulations. Vigilance for ARTa was different for each simulation. Five simulations were performed using fuzzy ARTMAP algorithm. The vigilance values for ARTa during training were equal to 0.95, 0.9, 0.85, 0.75, and 0.65, respectively one for each simulation. And the corresponding testing vigilance for ARTa was equal to 0.9, 0.9, 0.8, 0.7, and 0.6, respectively. Different vigilance parameters can be set for each ART. Fuzzy ART parameters, *alpha* and *beta*, were set to 0.001 and 1.0 respectively. The parameter *alpha* is used to define the initial long term memory values in ART. The value of *alpha* should be set as small as possible. The value of *beta* can vary in the range [0.0, 1.0]. Smaller the value of *beta* larger is the effect of old weights during training. Generally *beta* is set to 1.0 so that changes in weights should be dependent on current input patterns. All inputs were normalized to the range [0.0, 1.0] prior to training. Inputs were complement-coded so that low feature values do not diminish during ART training. A total of 20 epochs were performed on the training data. During each epoch 200,000 records were sampled from each attack category given by Normal, Probe, DoS, U2R, and R2L with probabilities of 0.7, 0.08, 0.2, 0.05, and 0.05, respectively. The model that performed the best with respect to cost per example value had vigilance parameter equal to 0.95 and 0.9 for training and testing ARTa. It had 0.999 as vigilance parameter for ARTb both during training and testing. Cost per example for this model was equal to 0.2497.

2.3.9. C4.5 decision tree (C4.5). The C4.5 algorithm [C4.5 Simulator], developed by Quinlan [12], generates decision trees using an information theoretic methodology. The goal is to construct a decision tree with minimum number of nodes that gives least number of misclassifications on training data. The C4.5 algorithm uses divide and conquer strategy.

Initial window was set to 20% of the records present in the KDD training dataset. 20% of records present in the initial window size were added after each iteration and the tree was retrained. In all tests, at least two branches contained a minimum of two records. The cost per example achieved for the best decision tree classifier model was equal to 0.2396.

2.4. Classification performance of algorithms on KDD testing dataset

Best performing instances of all nine classifiers developed through the KDD training dataset in Section 2.3 were evaluated on the KDD testing data set. For a given classifier, its probability of detection and false alarm rate performance on a specific attack category was recorded. Simulation results are presented in Table 1. Both probability of detection (PD) and false alarm rate

(FAR) are indicated for each classifier algorithm and each attack category.

Table 1 shows that a single algorithm could not detect all attack categories with a high probability of detection and a low false alarm rate. Results also show that for a given attack category, certain algorithms demonstrate superior detection performance compared to others. MLP, GAU, K-M, NEA, and RBF detected more than 85% of attack records for probing category. For attack records in DoS category, MLP, K-M, NEA, LEA, and HYP scored a 97% detection rate. GAU and K-M, the two most successful classifiers for U2R category, detected more than 22% of attack records. In case of R2L category, only GAU could detect around 10% of attack records. Once the promising algorithms for each attack category are identified, the best algorithm can be chosen also considering the FAR values. Hence, MLP performs the best for probing, K-M for DoS as well as U2R, and GAU for R2L attack categories.

Table 1. PD and FAR for various algorithms

		Probe	DoS	U2R	R2L
MLP	*PD*	0.887	0.972	0.132	0.056
	FAR	0.004	0.003	5E-4	1E-4
GAU	*PD*	0.902	0.824	0.228	0.096
	FAR	0.113	0.009	0.005	0.001
K-M	*PD*	0.876	0.973	0.298	0.064
	FAR	0.026	0.004	0.004	0.001
NEA	*PD*	0.888	0.971	0.022	0.034
	FAR	0.005	0.003	6E-6	1E-4
RBF	*PD*	0.932	0.730	0.061	0.059
	FAR	0.188	0.002	4E-4	0.003
LEA	*PD*	0.838	0.972	0.066	0.001
	FAR	0.003	0.003	3E-4	3E-5
HYP	*PD*	0.848	0.972	0.083	0.010
	FAR	0.004	0.003	9E-5	5E-5
ART	*PD*	0.772	0.970	0.061	0.037
	FAR	0.002	0.003	1E-5	4E-5
C4.5	*PD*	0.808	0.970	0.018	0.046
	FAR	0.007	0.003	2E-5	5E-5

It is reasonable to state that the set of pattern recognition and machine learning algorithms tested on the KDD data sets offered an acceptable level of misuse detection performance for only two attack categories, namely Probing and DoS. On the other hand, all nine classification algorithms failed to demonstrate an acceptable level of detection performance for the remaining two attack categories, which are U2R and R2L.

Subsequent to the observation that for a given attack category, certain subset of classifier algorithms offer superior performance as empirically evidenced through results in Table 1, a multi-classifier design, where classifier components are selected among the best performing ones for a given attack category justifiably deserves further attention. This very issue will be elaborated in the next section.

3. Multi-classifier model

Results in Section 2 suggest that the performance can be improved if a multi-classifier model is built that has sub-classifiers trained using different algorithms for each attack category. Section 2 identified the best algorithms for each attack category: MLP for probing, K-M for DoS as well as U2R, and GAU for R2L. This observation can be readily mapped to a multi-classifier topology as in Figure 1. Table 2 shows the performance comparison of the proposed multi-classifier model with others in literature. Results in Table 2 suggest that the multi-classifier model showed significant improvement in detection rates for probing and U2R attack categories. Also the FAR was reasonably small for all attack categories.

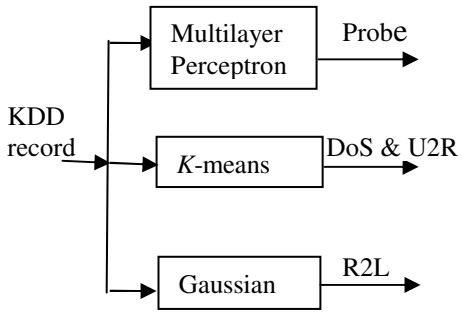

Figure 1. Multi-classifier model

Table 2. Comparative detection performance of Multi-classifier

		Probe	DoS	U2R	R2L
KDD Cup Winner	*PD*	0.833	0.971	0.132	0.084
	FAR	0.006	0.003	3E-5	5E-5
KDD Cup RunnerUp	*PD*	0.833	0.971	0.132	0.084
	FAR	0.006	0.003	3E-5	5E-5
Agarwal and Joshi	*PD*	0.73	0.969	0.066	0.107
	FAR	8E-5	0.001	4E-5	8E-4
Multi-Classifier	*PD*	0.887	0.973	0.298	0.096
	FAR	0.004	0.004	0.004	0.001

A second measure, the cost per example, can also be leveraged to elaborate further on the comparative performance assessment. Winner of the KDD-99 intrusion detection competition, who used C5 decision trees, obtained cost per example value of 0.2331.

Proposed multi-classifier model (created by selecting the best algorithms that could achieve the highest detection rates for each attack category) was able to achieve 0.2285 cost per example value on the KDD dataset test. This is better than that achieved by the KDD'99 Cup winner.

4. Conclusions

A simulation study was performed to assess the performance of a comprehensive set of machine learning algorithms on the KDD 1999 Cup intrusion detection dataset. Simulation results demonstrated that for a given attack category certain classifier algorithms performed better. Consequently, a multi-classifier model that was built using most promising classifiers for a given attack category was evaluated for probing, denial-of-service, user-to-root, and remote-to-local attack categories. The proposed multi-expert classifier showed improvement in detection and false alarm rates for all attack categories as compared to the KDD 1999 Cup winner. Furthermore, reduction in cost per test example was also achieved using the multi-classifier model. However, none of the machine learning classifier algorithms evaluated was able to perform detection of user-to-root and remote-to-local attack categories significantly (no more than 30% detection for U2R and 10% for remote-to-local category). In conclusion, it is reasonable to assert that machine learning algorithms employed as classifiers for the KDD 1999 Cup data set do not offer much promise for detecting U2R and R2L attacks within the misuse detection context.

5. References

[1] W. Lee, S. J. Stolfo, K. W. Mok, "A Data Mining Framework for Building Intrusion Detection Models", IEEE Symposium on Security and Privacy, Oakland, California, 1999a, pp. 120-132.

[2] 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*, San Diego, CA, 1999b, pp. 114-124.

[3] KDD data set, 1999; http://kdd.ics.uci.edu/databases/-kddcup99/kddcup99.html

[4] R. Agarwal, and M. V. Joshi, "PNrule: A New Framework for Learning Classifier Models in Data Mining", *Technical Report TR 00-015*, Department of Computer Science, University of Minnesota, 2000.

[5] I. Levin, "KDD-99 Classifier Learning Contest LLSoft's Results Overview", *SIGKDD Explorations, ACM SIGKDD*, January 2000, Vol. 1 (2), pp. 67-75.

[6] L. Ertoz, M. Steinbach, and V. Kumar, "Finding Clusters of Different Sizes, Shapes, and Densities in Noisy, High Dimensional Data", *Technical Report.*

[7] D. Y. Yeung, and C. Chow, "Parzen-window Network Intrusion Detectors", S*ixteenth International Conference on Pattern Recognition*, Quebec City, Canada, August 2002, pp. 11-15.

[8] K. Kendall, "A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems", *Master's Thesis*, MIT, Boston, MA, 1998.

[9] C. Elkan, "Results of the KDD'99 Classifier Learning", *SIGKDD Explorations, ACM SIGKDD*, Jan 2000.

[10] LNKnet software, *http://www.ll.mit.edu/IST/lknet/-index.html.* Cited November 2002.

[11] C4.5 simulator, Developer: *http://www.rulequest-.com*; Download code from: *http://www.cs.uregina.ca/~dbd/cs831/notes/ml/dtrees/c4.5 /tutorial.html.* Cited November 2002.

[12] P. Werbos, "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences", *PhD thesis*, Harvard University, 1974.

[13] R. O. Duda, and P. E. Hart, "Pattern Classification and Scene Analysis", *New York: Wiley*, 1973.

[14] F. M. Ham, "Principles of Neurocomputing for Science and Engineering", *McGraw Hill,* 1991.

[15] J. A. Hartigan, "Clustering Algorithms", New York: John Wiley and Sons, 1975.

[16] B.G. Batchelor, "Pattern Recognition: Ideas in Practice", *Plenum Press: New York*, 1978.

[17] Y. Lee, "Classifiers: Adaptive Modules in Pattern Recognition Systems", Cambridge, MA: MIT, Dept. of Electrical Engineering and Computer Science, 1989.

[18] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, & D. B. Rosen, "Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps", *IEEE Trans. on Neural Networks,* 1992, Vol. 3, pp. 698 – 713.