Proceedings of the 2004 IEEE
Workshop on Information Assurance and Security
United States Military Academy, West Point, NY, 10–11 June 2004

Dimension Reduction Using Feature Extraction Methods for Real-time Misuse Detection Systems

Gopi K. Kuchimanchi, Vir V. Phoha, Kiran S. Balagani, Shekhar R. Gaddam

Abstract— **We present a novel signed Gain in Information (GI) measure for quantitative evaluation of gain or loss in information due to dimension reduction using feature extraction in misuse detection applications. GI is defined in terms of Sensitivity Mismatch Measure (Φ) and Specificity Mismatch Measure (Θ). 'Φ' quantifies information gain or loss in feature-extracted data as the change in detection accuracy of a misuse detection system when reduced data is used instead of untransformed original data. Similarly, 'Θ' quantifies information gain or loss as the change in the number of false alarms generated by a misuse detection system when feature-extracted data is used instead of original data. We present two neural network methods for feature extraction: (1) NNPCA and (2) NLCA for reducing the 41-dimensional KDD Cup 1999 data. We compare our methods with principal component analysis (PCA). Our results show that the NLCA method reduces the test data to approximately 30% of its original size while maintaining a GI comparable to that of PCA and the NNPCA method reduces the test data to approximately 50% with GI measure greater than that of PCA.**

Keywords— **Feature selection, real-time misuse intrusion detection, network security, component analysis, sensitivity mismatch measure, specificity mismatch measure.**

I. Introduction

Intrusion detection Systems (IDS) [1] have become popular tools for identifying anomalous and malicious activities in computer systems and networks. There are two types of IDS: (1) Misuse Detection Systems [2] that detect abnormal patterns in system usage by comparing them with known signature patterns and (2) Anomaly Detection Systems [3] that construct profiles of normal behavior and flag all deviations from estimated profiles as intrusions. Recently, a new class of IDS employing data mining techniques called Intrusion Detection using Data Mining (IDDM) [4] have gained popularity because of their abilities to automatically extract attack signatures, detect unseen anomalies, maintain high detection accuracies with low false alarm rates, and scale on large distributed datasets. A considerable subset of IDDMs [5], [6], [7] perceive misuse intrusion detection as a data partitioning problem in which data samples are classified as attacks or a non-attack. Such classifier based approaches employ wide range of machine learning techniques like neural networks [6], support vector machines [8], fuzzy logic [9], and others [10], [11] to identify intrusions. A detrimental aspect of deploying these techniques for real-time intrusion detection applications is their high time and space complexities, essentially due to the large dimensionality of the input feature space in which these techniques operate. The time and space complexities of most classifiers are exponential functions of their input vector size [12]. Moreover, demand for the number of samples for training the classifier grows exponentially with the dimension of the feature space. This limitation is called the 'curse of dimensionality'. Reducing feature space by extracting features that truly contribute to classification cuts preprocessing costs and minimizes the effects of the 'peaking phenomenon' in classification [13], thereby improving the overall performance of classifier based intrusion detection systems.

In this paper we present two neural network methods (NNPCA and NLCA) to reduce the dimensionality of TCP network traffic through feature extraction. We compare our methods with the traditional matrix method of Principal Component Analysis (PCA) and show that the NLCA and NNPCA perform better than PCA in eliminating redundant information and retaining essential causal and dynamic traits in the data that significantly contribute to decision making in classifier based misuse intrusion detection applications. To experimentally verify the effects of feature extraction on the detection accuracy and false positive rate of a classifier based IDS, we implement misuse detection systems using two classifiers: (1) the Non-linear classifier (NC), and (2) the CART decision tree classifier (DC) for machine discrimination of normal and intrusive network traffic. For our experiments we use the KDD Cup 1999 intrusion detection dataset prepared by Lee *et al.* [5]. The dataset contains 41 features representing selected measurements of normal and intrusive TCP sessions. Each labeled TCP session is either normal or a member of one of the 22 attack classes in the dataset. We report the performance of the classifiers in recognizing TCP session classes in four datasets: (1) the original KDD Cup 1999 dataset with 41 features, (2) the PCA reduced dataset, and (3) the NNPCA reduced dataset, and (4) the NLCA reduced dataset.

This project is supported in part by the Army Research Office under Grant No. DAAD 19-01-0646 and by the Louisiana Board of Regents under Grant No. LEQSF 2003-05-RD-A-17

The authors are with the Department of Computer Science, Louisiana Tech University, Ruston, LA.

Let 'd' be the original dimension of the feature space to be reduced. The problem of feature extraction can be formally stated as finding a function Ψ such that: (1) $\Psi : \Re^d \rightarrow \Re^k$, where $k < d$ is the dimension of the reduced feature subspace and (2) Ψ maximizes the Gain in Information (GI) measure. We define GI due to dimension reduction as the sum of Sensitivity Mismatch Measure (Φ) and Specificity Mismatch Measure (Θ). Let D denote the d-dimensional full data and K denote the k-dimensional reduced data obtained by $\Psi(D)$ using a reduction method M. The Sensitivity Mismatch Measure Φ gives the gain or loss of information in K, due to which the detection accuracy of an intrusion detection classifier increases or decreases when K is used as an alternative to D. The Sensitivity Mismatch Measure is given as $\Phi_{C_M} = \mu_C(K) - \mu_C(D)$, where $\mu_C(.)$ is the detection accuracy of classifier C on the data. If $\Phi_{C_M} > 0$, the reduced data 'K' is said to have *gained* information with classifier C. If $\Phi_{C_M} < 0$, the reduced data is said to have *lost* information. If $\Phi_{C_M} = 0$, K is said to have *retained* information. Similarly, the Specificity Mismatch Measure Θ gives the gain or loss of information in K, due to which the false positive rate of an intrusion detection classifier increases or decreases when K is used instead of D. The Specificity Mismatch Measure is given as $\Theta_{C_M} = \nu_C(D) - \nu_C(K)$, where $\nu_{C_M}(.)$ is the false positive rate of classifier C on the data. If the value of $\Theta_{C_M} > 0$, the data reduced by method M is said to have *gained* information with classifier C. If $\Theta_{C_M} < 0$, K is said to have *lost* information. If $\Theta_{C_M} = 0$, K is said to have *retained* information.

In addition to fast and effective reduction of dimensions in the feature space, feature extraction methods for real-time intrusion detection applications should consider changes in the behavior of background traffic workloads and should accommodate new features that may be discovered in near future. In this context, we introduce two properties essential to feature extraction for intrusion detection: (1) the *Adaptive Property* and (2) the *Scaling Property*. The first property refers to the ability of the method to adapt to variations in background network or host traffic workloads. This property essentially gives the extraction method the ability to handle burstiness (high variations in background traffic over varying time scales) [14], [15] and non-stationarity [16] inherent in internet traffic. This property also delays the overfitting-like phenomenon in which the approximation function Ψ no longer qualifies (over a period of time) to transform the data entering the intrusion detection classifier. Thus, the adaptive property contributes to the stability of the subsequent classification process. The second property refers to the ability to incorporate newly identified features while maintaining minimum retraining costs. PCA [17] is an efficient method to reduce dimensionality by providing a linear map of d-dimensional feature space to a reduced k-dimensional feature space.

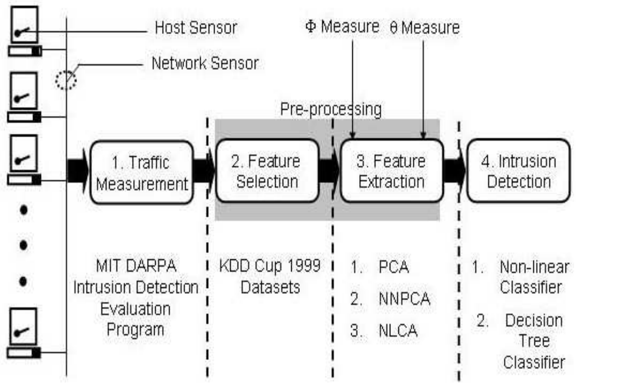

Fig. 1. Four stages of classifier based IDS architecture and their implementation details. The work in this paper contributes to the third stage using PCA, NNPCA, and NLCA methods and fourth stage using NC and DC.

However, PCA has high time complexity (which grows atleast quadratically with dimensions of data) and lacks the properties of adaptivity and scaling that are essential to real-time intrusion detection applications. Alternatively, the neural network approach to dimension reduction using NNPCA and NLCA give instantaneous response to input, adapt to new input patterns and scale to incorporate new features with minimal retraining. To experimentally verify the effects of feature extraction by PCA, NNPCA, and NLCA on the detection accuracy and false positive rate of a subsequent classification process, we implement two classifiers. The first classifier NC was chosen as a representative of conventional distance based statistical pattern recognition capable of yielding non-linear decision boundaries. The second classifier DC was chosen as a representative of nonmetric methods to classification. The results of classifying the original datasets and its low-dimensional counterparts obtained by PCA, NNPCA and NLCA show that the classifiers achieve equivalent and (in some cases) better detection accuracies and false positives using the reduced datasets.

The four stages of a classifier based intrusion detection system are illustrated in Figure 1. The first stage involves measurement of traffic workloads using host and network sensors. Sensors are software programs (e.g., sniffers capturing packets and software utilities monitoring system calls and cpu cycles) that monitor selected characteristics of the host/network traffic. Traffic measurement was done under DARPA Intrusion Detection Evaluation Program [18]. The second stage (preprocessing) involves two tasks. The first task of feature selection was undertaken by Lee *et al.* [5] and the datasets are available in [19]. In this paper we perform the second task of feature extraction by using the 10% subset of KDD Cup 1999 dataset for identifying key features that contribute to classifier based misuse detection. The dataset contains approximately 500,000 TCP sessions. Each session contains 41 selected measurements of TCP connections and is labeled either as an attack or a non-attack. The final stage involves attack detection for which we use NC and DC classifiers.

The rest of the paper is organized as follows. In Section II we describe feature extraction using component analysis and present the results of dimension reduction on KDD Cup 1999 data. In Section III we discuss Non-linear and CART decision tree classifiers. In Section IV we discuss the results of our work and conclude Section V.

II. Component Analysis

Component analysis is an unsupervised approach to find significant features in the data. In this section we discuss three component analysis techniques (1) PCA, (2) NNPCA, and (3) NLCA for reducing dimensionality of KDD Cup 1999 intrusion detection dataset. The dataset contains 41 features of TCP traffic. Each feature vector is labeled as an attack or a non-attack. There are 22 types of attacks in the dataset. We divide the dataset into training data subset (containing 25% of randomly selected representative samples from the original dataset) and reserve the rest for testing. On performing component analysis on the training subset, we obtain three compressed datasets in addition to the original dataset. Table I summarizes the four datasets later used for misuse detection using NC and DC.

TABLE I
Summary of datasets obtained after feature extraction.

Dataset Name	Reduced Dimensions	Method
ORIGDATA	41	None
PCADATA	19	PCA
NNPCADATA	19	NNPCA
NLCADATA	12	NLCA

A. PCA

Principal Component Analysis [17] reduces the dimensionality of data by restricting attention to those directions in the feature space in which the variance is greatest. In PCA, the proportion of the total variance accounted for by a feature is proportional to its eigenvalue. We perform PCA on 25% training data subset. Here, the goal is to reduce the cardinality of the dimensions (d) in the data where $d = 41$. We excluded two features (1) *number_of_outbound_commands* and (2) *is_host_login* as their values remained constant throughout the dataset, reducing d to 39. We compute the correlation matrix for the training dataset. Next, we compute the eigenvalues and sort them in decreasing order. The first eigenvalue e_1 corresponds to the first principal component, the second eigenvalue e_2 corresponds to the second principal component and so on. Table II shows the first 20 eigenvalues arranged in decreasing order. We use two tests: (1) Scree Plot test [17] and (2) Critical Eigenvalue test [17] as a test of hypothesis that k features are sufficient against the alternative that more than k features are required. The remaining $d-k$ features are assumed to contain noise or redundancy.

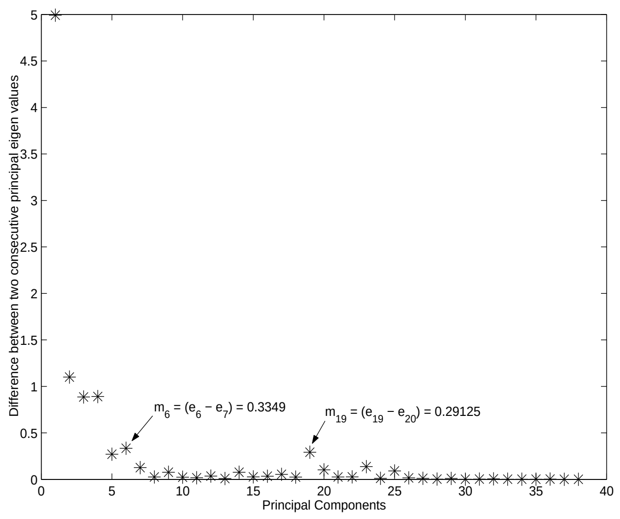

Fig. 2. Scree plot illustrates the differences in eigenvalues of the sixth and seventh (m_6), and nineteenth and twentieth (m_{19}) principal components.

In Scree Plot test, we plot the principal components against the differences m_i in successive sorted eigenvalues (e.g., $m_i = e_i - e_{i+1}$). The Scree plot in Figure 2 shows principal components plotted against the successive differences in eigenvalues shown in Table II. In this plot, the difference between successive eigenvalues decreases regularly from 4.9918 to 0.2707 for the first six principal components. Then there is an increase in difference between the sixth and seventh eigenvalues ($m_6 = e_6 - e_7 = 0.3349$), breaking the decreasing trend. The difference again decreases regularly from the seventh to nineteenth principal component. Then, there is a break in decreasing trend between the nineteenth and twentieth principal components ($m_{19} = e_{19} - e_{20} = 0.2912$), suggesting that either the first six components or the first nineteen components are the most significant components in the data. Since the Scree test suggests two sets of principal components, we perform Critical Eigenvalue test to verify the results of the Scree test. The Critical Eigenvalue test recommends selecting all principal components whose eigenvalues exceed the eigenvalue threshold, $\tau_e = d^{0.6}/15$, where d is the dimension of the original dataset. For our dataset, τ_e is 0.6005. The nineteenth principal component is the last principal component with an eigenvalue (approximately 0.74) that exceeds τ_e. Moreover, we have observed that the first nineteen principal components account for 94.91% of the total variation associated with all the thirtynine original variables, while the first six principal components account for only 62.57%. The analysis presented above shows that the number of significant features in the KDD Cup dataset is 19.

TABLE II
THE FIRST 20 FEATURES AND CORRESPONDING EIGENVALUES AND STANDARD DEVIATIONS (S.D). THE FIRST 19 FEATURES ARE SELECTED BASED ON RESULTS OF SCREE TEST AND CRITICAL EIGENVALUE TEST.

Feature Name	S. D	Eigen Value
src_bytes	51926	9.74
dst_bytes	29423	4.75
duration	773.65	3.65
is_guest_login	247.93	2.76
is_host_login	217.9	1.87
srv_diff_host_rate	106.64	1.60
diff_srv_rate	69.196	1.27
service	1.2974	1.14
flag	1.1499	1.11
protocol_type	0.9664	1.04
num_root	0.7784	1.02
hot	0.7769	1.00
num_compromised	0.6720	0.96
dst_host_same_srv_rate	0.4849	0.95
dst_host_count	0.4128	0.88
rerror_rate	0.3886	0.85
srv_count	0.3816	0.82
dst_host_srv_diff_host_rate	0.3815	0.76
count	0.3813	0.74
dst_host_same_src_port_rate	0.3812	0.45

Next, we find the actual features in KDD Cup 1999 datasets that correspond to the 19 significant principal components. Since the principal components explain almost all the variance in the data, we sort the 39 original features in the decreasing order of their standard deviation (shown in Table II) and select the first 19 features as the features that correspond to the 19 principal components. Hereafter we refer to the PCA-compressed intrusion detection dataset as PCADATA.

B. NNPCA and NLCA

In this section we present two neural network based component analysis techniques: (1) neural network principal component analysis (NNPCA), and (2) nonlinear component analysis (NLCA) for dimension reduction. Feedforward neural networks can be used to extract significant components [20]. Such networks are called autoassociative neural networks. The networks contain d nodes in input and output layers, corresponding to d-dimensions of the data to be reduced. One of the hidden layers in an autoassociator is called a bottleneck layer because the d-dimensional inputs pass through the k-dimensional bottleneck before reproducing the inputs. Therefore, dimension reduction occurs in the bottleneck layer. Nodes in the hidden and output layers use sigmoidal functions as activation functions. For our experiments with NNPCA and NLCA all samples in training, validation and testing datasets were normalized. The 25% training subset of KDD Cup 1999 dataset (with 127,437 tcp sessions) was split into training data (containing 80% of the 25% training subset) and validation data (containing 20% of the 25% training subset). The remaining 75% of KDD Cup 1999 data subset was used for testing.

Our NNPCA architecture initially consists of three layers with d units in each layer, corresponding to the 41 dimensions of the KDD Cup 1999 data. Each node in the network is associated with an individual learning parameter. The values of the learning parameters ranged from 0.1 to 0.5. Values between 0.0 to 1.0 were randomly chosen to initialize the weights. Dimension reduction was performed in two phases: (1) the Stopping Rule Estimation phase, and (2) the Bottleneck Layer Pruning phase. The first phase involves finding the early stopping criterion for the autoassociator. Early stopping criterion is represented in terms of two parameters: (i) the prediction accuracy over validation data (**A**), and (ii) the number of epochs (**E**) required to attain **A**. The idea behind finding the early stopping criterion is that training is allowed to continue sufficiently long to fit the structure of the underlying data, but not long enough to fit the noise. In this phase, we train the autoassociator using gradient descent rule and periodically (for every five epochs) check the prediction accuracy over the validation data. As the autoassociator is trained, the value of **A** increases until it reaches a saturation point, where it begins to drops. At this point the autoassociator training is stopped and the Stopping Rule Estimation parameters **A** and **E** are recorded. In our experiments we recorded a value of approximately 97% for **A** and 30 for **E**.

The second phase involves iterative pruning of the autoassociator model M_{41} with 41 nodes by eliminating nodes based on the variance in bottleneck node outputs over the validation dataset. The symbols and notations used in the procedure Bottleneck_Layer_Pruning are: (i) M_i is the autoassociator model with i nodes in the bottleneck layer, (ii) a_i is the prediction accuracy of M_i over the validation dataset, (iii) e_i counts the epochs while training the autoassociator model M_i, (iv) α_j is the learning parameter associated with j^{th} node in the bottleneck layer (initialized with values between 0.1 and 0.5), (v) Var_j is the variance in the outputs (obtained by passing the validation data) associated with j^{th} node in the bottleneck layer when model M_i reaches the stopping criteria, (vi) τ_{var}, is the variance threshold for pruning the bottleneck nodes, and (vii) δ is the learning parameter (α) increment factor. A formal procedure illustrating the Bottleneck Layer Pruning follows.

```
Procedure Bottleneck_Layer_Pruning($M_i$)
{
    for $j \leftarrow 1$ to i
        $Var_j \leftarrow 0$;
    end for
    // $\tau_{var}$ is set to 0.00001 in our experiments.
    while $Var_j \geq \tau_{var}$
        $a_i \leftarrow 0$;
        $e_i \leftarrow 0$;
        Assign random weights to $M_i$
        do
            Train $M_i$ using gradient descent
            rule.
            Calculate $a_i$ over the validation
            dataset.
            $e_i \leftarrow e_i + 1$;
        until $a_i = A \mid\mid e_i = E$
        for $j \leftarrow 1$ to i
            Calculate $Var_j$;
        end for
        if $Var_j$ = $min_{k=1:i}(Var_k)$
        // $\delta$ is set to 0.1 in our experiments.
            $\alpha_j \leftarrow \alpha_j + \delta$;
        end if
    end while
    if $Var_j \leq \tau_{var}$
        Prune node $j$;
    end if
    $i \leftarrow i - 1$;
    if $i > 0$
        Call Bottleneck_Layer_Pruning($M_i$);
    end if
}
```

NNPCA (and PCA) identify linear correlations among features. It is possible that there may be strong nonlinear relationships among features leading to better dimension reduction. To test for nonlinear relationships, we perform a nonlinear component analysis (NLCA) on normalized 25% training data subset. For NLCA, Kramer [21] suggests adding two hidden layers with nonlinear nodes to the NNPCA architecture. Lu and Hsieh [22] further simplify Kramer's five layered architecture with a four layer neural network with hyperbolic tangent activations and show that their simplified architecture alleviates overfitting. In our experiments we used a four layer neural network similar to [22], but with sigmoidal nodes in the hidden layers (i.e., bottleneck layer and decoding layer) to achieve bounded outputs. This neural network architecture was initialized with 10 nodes in the decoding layer (our experiments with higher number of nodes resulted in unstable architectures) and 41 nodes in the bottleneck layer. Dimension reduction was performed using the Stopping Rule Estimation and Bottleneck Layer Pruning as in NNPCA. The early stopping criterion parameters were recorded as $\mathbf{A} = 76.2214$ and $\mathbf{E} = 5$. Bottleneck Layer Pruning was performed (with τ_{var} set to 0.0001 and δ to 0.01) on 40 NLCA models.

Figure 3 shows the prediction accuracies over validation data by pruning the bottleneck layers in NNPCA and NLCA neural networks. The plot (in solid line) in Figure 3

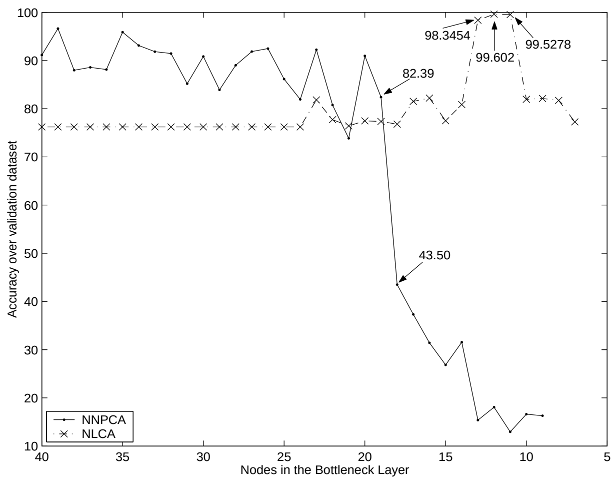

Fig. 3. Variation in prediction accuracies of NNPCA and NLCA networks over validation data. The solid lined plot shows prediction accuracies for NNPCA autoassociator models. The dashed plot shows prediction accuracies for NLCA neural networks.

shows that the prediction accuracies for NNPCA autoassociator models range from approximately 97% - 74% from M_{41} to M_{19}, which contains 19 nodes in the bottleneck layer. For the next model M_{18}, the prediction accuracy drastically drops to 43% and continues to drop thereafter, suggesting that the autoassociator model M_{19} is the optimal choice to generate the transformed dataset containing 19 components, each component corresponding to a single node in its bottleneck layer. We refer to this NNPCA transformed dataset as NNPCADATA hereafter. The dashed-line plot in Figure 3 shows that the prediction accuracy peaks for the NLCA neural network with 12 nodes in the bottleneck layer suggesting that there are 12 significant components in the data. Hereafter, we refer to the NLCA transformed dataset with 12 components as NLDATA.

III. Classifiers for Misuse Detection

We present two classifiers: (1) Non-linear classifier (NC), and (2) CART Decision Tree classifier (DC) to experimentally verify the effects of reducing the dimensions of the TCP traffic data on the detection accuracy and false positive rate. Classification was performed on four datasets summarized in Table I. For our experiments with NC and DC, we use 25% of KDD Cup 1999 data (and its low-dimensional counterparts) for training and the remaining 75% for testing.

A. Non-linear Classifier (NC)

NC is a classification technique in which the mean of a class is represented by the mean vector M. The standard deviation vector S defines the boundaries of the cluster in a z-dimensional euclidean space. Training samples are used to generate sample mean and standard deviation for each class C. The mean vector $M_i = \{m_1, m_2, \cdots, m_k, \cdots, m_z\}$ and standard deviation vector $S_i = \{s_1, s_2, \cdots, s_k, \cdots, s_z\}$

form the prototype for classes $C_{i=1to23}$ corresponding to normal and attack instances in the dataset. Here, m_k is the mean of the training samples for the k^{th} component of the dataset, and s_k is the standard deviation of that component. NC performs an independent test in each dimension. The membership of a sample $\overline{X} = \{x_i, x_2, \cdots, x_z\}$ to a class C is determined by the maximum number of dimensions in which the *class membership criterion* $\overline{X} \in C \Leftrightarrow |x_i - m_i| < ns_i \ \forall i$ is satisfied. Here n is a tunable parameter. We perform experiments on NC by varying the n-parameter values from 1 to 10.5 and present the results (in Section IV) in terms of detection accuracy and false positive rate.

B. Decision Tree Classifier (DC)

We use Classification and Regression Trees (CART) [23] as a second approach to build a misuse detection classifier. CART has become popular in a wide range of applications ranging from data mining to predictive modeling, due to its general framework that can be configured to produce different decision tree models for classification. The process of obtaining an optimal classification model consists of three stages [24]: (1)choosing the splitting criterion, (2) pruning, and (3) choosing complexity parameter. We use Gini Impurity as the splitting rule. The Gini impurity is a generalization of variance impurity, which is a measure of the variance associated with any two decision variables. The Gini impurity criterion is:

$$\textbf{Gini Impurity} = 1.0 - S = 1.0 - \sum_{i=1}^{k} P\left(J/T\right)^2$$

where S is the sum of squared probabilities $P(J/T)$ summed over all levels of dependent variables, J is the total number of classes (with 22 attack classes and a normal class) in the dataset and T is the number of nodes in the decision tree. In our experiments DC was trained to grow on the training data until it was not possible to grow any further. After the full tree was generated, it was pruned using the cost complexity pruning criterion given as: $MR+\alpha(Size)$, where MR is the misclassification rate at node t, given by $MR = 1 - \max_{j=1:J} P(j/t)$. MR measures the minimum probability of misclassification of a training pattern at node t and α is the penalty parameter that determines the largest possible tree with lowest complexity. We analyze the performance of four DC models obtained by varying the complexity parameter from 0.0 to 0.0002 and present results (in Section IV) in terms of detection accuracy and false positive rate.

IV. Results and Discussion

We present the performance of the Non-linear classifier (NC) and the decision tree classifier (DC) in terms of detection accuracy and false positive rate. The detection accuracy of an IDS is the percentage of attack samples detected as attacks. The false positive rate of an IDS is the percentage of normal samples detected as attacks. We analyze the Receiver Operating Characteristic (ROC) curves for each classifier based IDS over four datasets: (1) ORIGDATA, (2) PCADATA, (3) NNPCADATA, and (4) NLDATA and present our insights. We compare the performance of the three component analysis methods in terms of Gain in Information **GI** measure introduced in Section I.

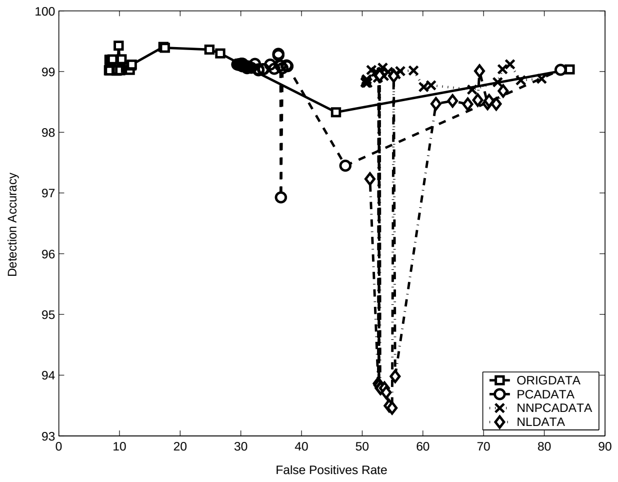

Fig. 4. ROC curves of NC over the four datasets. The curves shift toward the upper right corner when reduced datasets are used instead of original dataset.

Figure 4 shows ROC curves for NC on the four datasets. The detection accuracy and false positive rates plotted here were obtained by varying the values of the n-parameter of NC from 1 to 10.5. Our experiments showed that when n-parameter $\geq$ 8, the variation in average detection accuracy was almost constant (approximately 98%), while there was an alarming increase in false positive rates on all four datasets implying that the classifier itself is unusable as an IDS when n-parameter $\geq$ 8. Hence, we conducted our experiments on NC by varying the n-parameter only upto 10.5. The plot in Figure 4 shows that NC has reasonably good average detection accuracy 97.9304% over all four datasets, but shows an increase in false positive rates from ORIGDATA to NLDATA, with false positive rates of PCADATA and NNPCADATA being in between the other two datasets. To investigate whether the raise in false positives is because of the naivety of a classifier or because of the loss of information due to dimensionality reduction, we analyze the ROC curves of DC over the four datasets.

Figure 5 shows ROC curves for DC over the four datasets. The detection accuracies and false positive rates of DC were observed on four different decision tree models for each dataset. The models were obtained by pruning the complete decision tree until it shrinks to approximately 50% fo its original size. This reduction in size was achieved by varying the complexity parameter. The average detection accuracy of the DT classifier is 99.6438 over

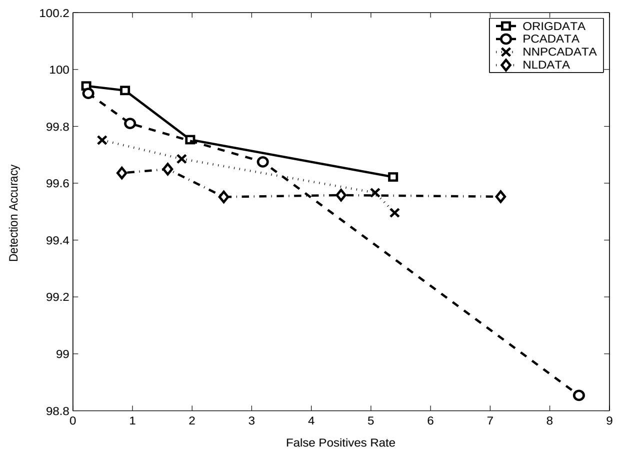

Fig. 5. ROC curves of DC over the four datasets. The curves show that detection accuracies and false positive rates over four datasets remain approximately same.

all the datasets. Varying the complexity parameter (0.0 to 0.000735) for experiments on ORIGDATA, the size of the decision tree models varied from 118 to 41 nodes with a minimum false positive rate of 0.2268 for the full decision tree model (with 118 nodes). For PCADATA, the size of the decision tree models varied from 113 to 41 nodes (by varying the complexity parameter from 0.0 to 0.000711) with a minimum false positives rate of 0.2609 for the decision tree model with 113 nodes. Similarly, for NNPCADATA the size of decision tree models varied from 190 to 75 nodes (by varying the complexity parameter from 0.0 to 0.0002) with a minimum false positive rate of 0.4922 for the decision tree with 190 nodes. For NLDATA, the size of decision tree models varied from 260 to 87 nodes (by varying the complexity parameter from 0.0 to 0.000145) with a minimum false positive rate of 0.8227 for decision tree with 260 nodes. Our experiments with the decision tree models on NLDATA showed that the average detection accuracy on five decision tree models was very high (99.589%) even after prunning the complete decision tree to one-third of its orginal size.

TABLE III

Minimum false positives rates (F.P.R.) with detection accuracies (D.A.) for NC and DC on the four datasets.

DATASET	F.P.R.		D.A.	
	NC	DC	NC	DC
ORIGDATA	8.2821	0.2268	99.0198	99.9428
PCADATA	29.4105	0.2609	99.1161	99.9167
NNPCADATA	50.5463	0.4922	98.8206	99.7516
NLDATA	51.2756	0.8227	97.2306	99.6359

Table III shows minimum false positive rates for NC and DC over four datasets. A row-wise observation of F.P.R values in Table III clearly indicate that the increase in false positive rates on reduced datasets (PCADATA, NNPCADATA, and NLDATA) as compared to the ORIGDATA for NC is due to the naivety of NC used in misuse detection applications. A column-wise inspection of F.P.R. values indicates that the growth trend in false positives (see the shift in ROC curves to upper right corner in Figure 4) drastically decreases on reduced datasets with DC (see Figure 5).

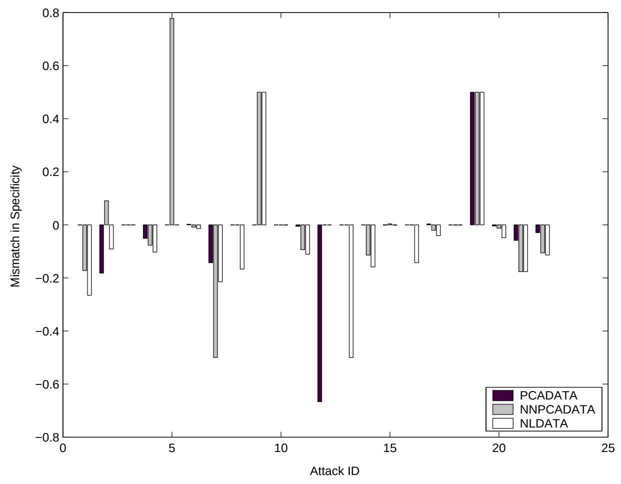

Fig. 6. A graphical interpretation of Specificity Mismatch Measure for DC. The attack list is available in UCI KDD archive. Attack IDs were given after arranging the attacks in alphabetical order.

We use the results of DC with relatively high detection accuracies and low false positive rates (shown in Table III) to show the Gain in Information *GI* (refer Section I) due to PCA, NNPCA, and NLCA transformations. Figure 6 illustrates the Specificity Mismatch Measure. The bars pointing upwards indicate a gain in information in terms of decrease in false positives due to dimension reduction. The bars pointing downwards indicate loss in information in terms of increase in false positives. The average Specificity Mismatch Measure over 22 attacks for PCA, NNPCA, and NLCA reduction methods were -0.000015, -0.00012, and -0.00027 respectively, indicating loss in information due to dimensionality reduction with all three methods. Figure 6 shows that PCA gains information for *spy* attack (with attack ID 19). NNPCA shows significant gains in information for *imap* (5), *multihop* (9) and *spy* attacks despite having less than 10 training instances for each of these attacks. NLCA shows gain in information for *multihop* and *spy* attacks.

Figure 7 illustrates the Sensitivity Mismatch Measure for 22 attacks over the three reduction methods. The bars pointing upward indicate gain in information in terms of increase in detection accuracies due to dimension reduction. The bars pointing downward indicate loss in information in terms of decrease in detection accuracies. The average Sensitivity Mismatch Measure over the 22 attacks for PCA,

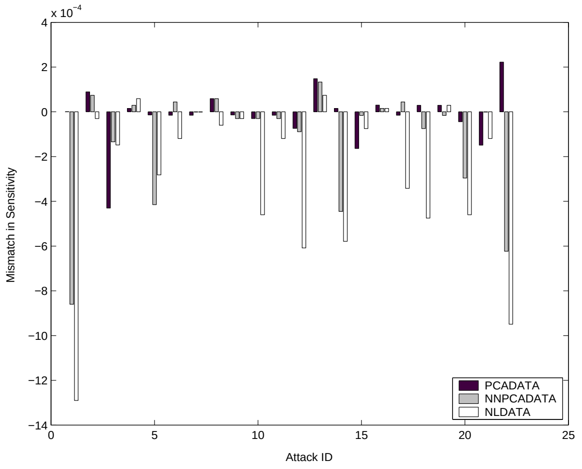

Fig. 7. A graphical interpretation of Sensitivity Mismatch Measure for DC.

and NLCA were -0.02829 and -0.05216 respectively, indicating loss in information due to dimensionality reduction. For NNPCA, the average Sensitivity Mismatch Measure was 0.0269 (a positive value) indicating gain in information. Figure 7 shows significant loss in information in NNPCA and NLCA reduced datasets for *back* (with attack ID 1), *imap*, *pod* (14), *teardrop* (20), and *warezmaster* (22) attacks. This loss is due to very less training instances (only 0.2694% of the total training instances) for these attacks.

The overall Gain in Information *GI* in data with DC due to PCA, NNPCA, and NLCA reductions were recorded as -0.02893, +0.026798, and -0.05243 respectively, showing that the data reduced using NNPCA method (NNPCA-DATA) gained information when compared to all other datasets over ORIGDATA with 41 features.

V. Conclusions

We have developed signed quantitative measures **GI**, Φ, and Θ to test the performance of feature extractors in intrusion detection applications. We use these measures to evaluate the performance of PCA, NNPCA, and NLCA feature extractors on misuse detection systems employing non-linear and decision tree classifiers. We show that the neural network feature extraction methods are more effective than PCA in reducing dimensions and retaining the causal dynamic information that is essential for maintaining high detection accuracy and low false positive rate in misuse detection systems. We are extending our work on quantitative measures to find optimal combinations of classifiers and feature extractors for intrusion detection systems.

References

[1] V. V. Phoha, *The Springer Dictionary of Internet Security*, vol. I. New York: Springer Verlag, June 2002.

[2] S. Kumar and E. H. Spafford, "A pattern matching model for misuse intrusion detection," in *Proceedings of the 17th National Computer Security Conference*, pp. 11–21, October 1994.

[3] H. S. Javitz and A. Valdes, "The sri ides statistical anomaly detector," in *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 316–326, IEEE Press, May 1991.

[4] T. Abraham, "IDDM: Intrusion Detection using Data Mining Techniques," Tech. Rep. DSTO-GD-0286, Defense Science and Technology Organization, Australia, 2000.

[5] W. Lee and S. J. Stolfo, "A Framework for Constructing Features and Models for Intrusion Detection Systems," *ACM Transactions on Information and System Security*, vol. 3, pp. 227–261, November 2000.

[6] Z. Zhang, J. Li, C. N. Manikopoulos, J. Jorgenson, and J. Ucles, "Hide: a hierarchical network intrusion detection system using statistical preprocessing and neural network classification," in *Proceedings of the 2001 IEEE Workshop on Information Assurance*, (New York), pp. 85–90, IEEE Press, June 2001.

[7] R. Agarwal and M. V. Joshi, "PNrule: A New Framework for Learning Classifier Models in Data Mining (A Case-Study in Network Intrusion Detection)," Tech. Rep. DSTO-GD-0286, Department of Computer Science, University of Minnesota, USA, 2000.

[8] S. Mukkamala, G. Janoski, and A. Sung, "Intrusion detection using neural networks and support vector machines," in *Proceedings of IEEE International Joint Conference on Neural Networks*, pp. 1702–1707, IEEE Press, 2002.

[9] J. Gomez and D. D. Gupta, "Evolving fuzzy classifiers for intrusion detection," in *Proceedings of the 2002 IEEE Workshop on Information Assurance*, (New York), IEEE Press, June 2001.

[10] I. Levin, "KDD-99 Classifier Learning Contest: LLSoft's Results Overview," *SIGKDD Explorations*, vol. 1, pp. 67–75, January 2000.

[11] D. Y. Yeung and C. Chow, "Parzen-Window Network Intrusion Detectors," in *Proceedings of the Sixteenth International Conference on Pattern Recognition*, vol. 4, pp. 385–388, August 2002.

[12] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, vol. I. New York: Wiley, 2002.

[13] A. K. Jain, R. P. W. Duin, and J. Mao, "Statistical Pattern Recognition: A Survey," *IEEE Transactions on Pattern Analysis and Mission Intelligence*, vol. 22, pp. 4–37, January 2000.

[14] M. E. Crovella and A. Bestavros, "Self-similarity in World Wide Web traffic: Evidence and possible causes," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 835–846, 1997.

[15] V. Paxson and S. Floyd, "Wide-area traffic: The failure of poisson modeling," *IEEE/ACM Transactions on Networking*, vol. 3, no. 3, pp. 226–244, 1995.

[16] J. Cao, W. S. Cleveland, D. Lin, and D. X. Sun, "On the non-stationarity of internet traffic," in *Proceedings of the ACM SIGMETRICS*, (Cambridge, MA), pp. 102–112, 2001.

[17] E. E. Cureton and R. B. D'Agostino, *FACTOR ANALYSIS An Applied Approach*, vol. I. London: Lawrence Erlbaum Associates, 1983.

[18] R. P. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K. R. Kendall, D. McClung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham, and M. A. Zissman, "Evaluating intrusion detection systems: the 1998 darpa off-line intrusion detection evaluation," in *Proceedings of the 2000 DARPA Information Survivability Conference and Exposition (DISCEX)*, vol. 2, (Cambridge, MA), pp. 12–26, 2000.

[19] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (UCI KDD Archive).

[20] P. Baldi and K. Hornik, "Neural networks and Principal Component Analysis: Learning from Examples Without Local Minima," *Neural Networks*, vol. 2, pp. 53–58, November 1989.

[21] M. A. Kramer, "Nonlinear Principal Component Analysis Using Autoassociative Neural Networks," *AIChE Journal*, vol. 37, pp. 233–243, February 1991.

[22] B. Lu and W. W. Hsieh, "Simplified nonlinear principal component analysis," in *3rd Conference on Artificial Intelligence Applications to the Environmental Science* , 2003.

[23] http://www.salford-systems.com.

[24] T. M. Mitchell, *Machine Learning*, vol. I. McGraw-Hill, 1997.