Data Fusion and Cost Minimization for Intrusion Detection

Devi Parikh, *Student Member, IEEE*, and Tsuhan Chen, *Fellow, IEEE*

***Abstract*—Statistical pattern recognition techniques have recently been shown to provide a finer balance between misdetections and false alarms than the more conventional intrusion detection approaches, namely misuse detection and anomaly detection. A variety of classical machine learning and pattern recognition algorithms has been applied to intrusion detection with varying levels of success. We make two observations about intrusion detection. One is that intrusion detection is significantly more effective by using multiple sources of information in an intelligent way, which is precisely what human experts rely on. Second, different errors in intrusion detection have different costs associated with them—a simplified example being that a false alarm may be more expensive than a misdetection and, hence, the true objective function to be minimized is the cost of errors and not the error rate itself. We present a pattern recognition approach that addresses both of these issues. It utilizes an ensemble of a classifiers approach to intelligently combine information from multiple sources and is explicitly tuned toward minimizing the cost of the errors as opposed to the error rate itself. The information fusion approach dLEARNIN alone is shown to achieve state-of-the-art performances better than those reported in the literature so far, and the cost minimization strategy dCMS further reduces the cost with a significant margin.**

***Index Terms*—Cost minimization, data fusion, dCMS, dLEARNIN, pattern recognition for intrusion detection.**

I. Introduction

With networking technology evolving so rapidly, more attacks on computer networks are carried out by exploiting unknown weaknesses or bugs always contained in system and application software [1]. Hence, computer security has been receiving a lot of attention in recent years. Two approaches to intrusion detection are conventionally used [2]—one is misuse detection and the other is anomaly detection.

Misuse detection is an attack signature-based approach that utilizes a detailed description of the sequence of actions performed by the attacker. This approach detects a series of actions as an attack only if it matches a previously seen attack signature identically. Hence, if a new attack is made, the system fails to recognize it. This is certainly suboptimal since new attacks and new attack variants are constantly being developed. More general signatures would reduce these misdetections but also give high false alarm rates. Hence, due to the requirement of low false alarm rates, such signature-based approaches are prevalent.

Anomaly detection is based on modeling the normal activity of the computer system. In this case, a traffic pattern whose profile deviates from this model is detected as an intrusion (i.e., any anomalous network activity is classified as an attack). This approach is general enough to detect new attacks with low false alarm rates provided that the model accurately represents its normal working condition, and that any attack and only an attack against the system involves its abnormal use. Unfortunately, the acquisition of profiles of normal activity is not an easy task [3]. The audit records used to produce the profiles of normal activity may contain traces of intrusions leading to misdetections, and also activities of legitimate users often deviate from their normal profile as modeled, leading to high false alarm rates.

The aforementioned discussion points out that the two intrusion detection approaches are usually formulated in terms of explicit matching paradigms [3]. All activities not matching the normal profiles are classified as an attack by anomaly detection approaches and only those activities matching one of the attack signatures are classified as attacks by misuse detection approaches. While such matching is effective when the patterns being classified exhibit a regular and repeatable structure, this is not the case for network traffic. Moreover, for most current intrusion detection systems, the development of matching rules for anomaly and misuse detection relies on the experience and intuition of human experts [4] which is highly subjective. As a consequence, such rules can hardly adapt to the high variability of normal activities or to the number of novel attacks constantly being developed.

These difficulties in conventional intrusion detection systems lead researchers to apply statistical pattern recognition approaches [5], where statistical models for normal traffic and attack traffic are automatically built, simultaneously with the appropriate matching rules. The main motivation for using pattern recognition approaches for the development of advanced intrusion detection systems is their generalization capability, which can support the recognition of intrusions that have not been seen previously and have no previously described pattern [2]. This formulation of the intrusion detection problem combines the advantages of a signature-based and anomaly-based intrusion detection system. A technical report on intrusion detection technology, where commercial and research products are briefly reviewed, provides a discussion on the challenges to develop effective intrusion detection systems [4]. In particular, it has been pointed out that advanced research issues on

Manuscript received October 12, 2007; revised June 17, 2008. Published August 13, 2008 (projected). This work was supported in part by the National Science Council, Taiwan, R.O.C., through the International Collaboration on Advanced Security Technology (iCAST) and in part by a National Science Foundation (NSF) Graduate Research Fellowship. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Lang Tong.

The authors are with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: dparikh@cmu.edu; tsuhan@cmu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2008.928539

intrusion detection systems should involve the use of pattern recognition approaches for the following three main reasons: 1) generalization from a representative set of examples allows detecting new types of intrusion; 2) attack signatures can be extracted automatically from labeled traffic data, thus allowing us to overcome the subjectivity of human interpretation of intrusive behavior, the latter being employed in many current intrusion detection systems; the intrusion detection system can adapt to new threats.

The problem formulation of intrusion detection in a pattern recognition framework has been provided in [2]. There are two observations we make about intrusion detection from a pattern recognition perspective.

1) Intrusion detection has much to gain from combining information from multiple sources, which is what human experts rely on, however, in an intelligent way. The most common categories of information are [6]: 1) intrinsic features that include general information about the connection, such as the duration, type, protocol, flag, etc.; 2) traffic features that encapsulate statistics related to past connections similar to the current one (e.g., number of connections with the same destination host); and 3) content features which contain information about the data content of packets, such as errors reported by the operating system or root-access attempts.
2) Different errors in classifying network traffic have different costs associated with them; for instance, the cost of a false alarm may be much higher than false detection, or more specifically, the cost of mistaking a certain attack type as normal traffic may be more than mistaking another attack type as normal.

In this paper, an approach to intrusion detection in computer networks that addresses both of these characteristics is presented. We present dLEARNIN, which utilizes an ensemble of classifiers approach that combines information from different sources of information. Also, we utilize a cost minimization strategy dCMS, to gear the final classification decision toward minimizing the cost of the errors, the true objective function, and not the error rate itself.

The rest of this paper is organized as follows. In Section II, we discuss some related work in the literature on using pattern recognition approaches for intrusion detection. We also specifically discuss the use of multiple classifier approaches for intrusion detection as well as the cost considerations made so far in intrusion detection literature. We also contrast our proposed approaches dLEARNIN and dCMS to other existing algorithms in pattern recognition literature for both tasks. Sections III and IV present the proposed algorithms dLEARNIN and dCMS, and results on a publicly available data set are provided in Section V. Section VI surveys some potential areas for future work, followed by conclusions in Section VII.

II. Related Work

A. Pattern Recognition Approaches to Intrusion Detection

The use of pattern recognition approaches to intrusion detection have been considered ever since the early years of intrusion detection system development. In particular, the application of neural networks for intrusion detection systems has been investigated by a number of researchers. Neural networks for intrusion detection have first been introduced as an alternative to the techniques in the intrusion detection expert system to model user behavior [7]. In particular, the typical sequence of commands executed by each user is learned. An approach to anomaly detection based on neural networks is proposed in [8]. A neural-network model designed to perform both anomaly and misuse detection has been proposed in [9]. The training set is made up of strings of events captured by the base security module that is a part of many operating systems. If the training set is made up of strings related to normal behavior, neural networks act as an anomaly detector. On the other hand, if strings captured during an attack session are included in the training set, the network model can be modified to act as a misuse detection system. Instead of audit data, traffic statistics are also used as features for misuse detection [10]. Traffic features at different levels of abstraction have been used, from packet data [11] to very high level features, such as the security level of the source and destination machines, occurrences of suspicious strings, etc. [12]. This shows that pattern recognition techniques based on neural networks are apt to provide a solution to some open issues in intrusion detection system development (e.g., the automatic extraction of normal and attack signatures from data and the ability to detect attacks not known at training time).

In addition, the extensive evaluation of pattern classification techniques carried out on a sample data set of network traffic performed during the KDD'99 [13] conference points out the feasibility of a pattern recognition approach to intrusion detection [12]. This dataset is a version of the standard set of data to be audited, as collected through the 1998 DARPA Intrusion Detection Evaluation Program at the MIT Lincoln Labs [14], whose objective was to survey and evaluate research in intrusion detection. The data set includes a wide variety of intrusions simulated in a military network environment. A review of the performance values of several pattern recognition algorithms on this data set has been provided in [15]. Apart from neural networks, the algorithms applied include Gaussian classifier, K-means clustering, nearest cluster algorithm, incremental radial basis functions, fuzzy ARTMAP, decision trees, etc. [15].

B. Multiple Classifiers Approaches

In most pattern recognition approaches applied to intrusion detection, classification is performed in the feature space made up of all the features available to detect the considered attack classes. It is easy to see that classifiers working in such a monolithic feature space can suffer from the curse of dimensionality, due to the large number of features necessary for effective intrusion detection and to the limited amount of training data that can be collected, especially for attack classes. It is important to realize that features with very different meanings, related to different characteristics of network traffic, are used in intrusion detection. It is well known that it can be very difficult for an individual classifier to effectively process features that have very different semantic meanings. We do need these diverse sources of information for effective intrusion detection, as is used by

human experts, however, combining them all in a high-dimensional feature space is not the answer. Instead, we should consider subsets of features that come from a common source, train classifiers on these subsets independently, and combine them in a meaningful fashion. Most methods for combining classifier outputs are based on the assumption that the outputs of different classifiers are independent. This assumption cannot be verified in practice, though in many real cases, it may reasonably hold. In particular, it may reasonably hold for the problem at hand, since the three types of features described earlier are related to independent connection characteristics.

Several multiple classifier approaches have been applied to intrusion detection. In some works, classifiers trained on different network services (such as ftp, mail, etc.) [16] or trained to detect different types of attacks [17] were combined. However, these do not address the issue of operating in a space of high dimensionality with features with different semantic meanings concatenated. Other works attempt to combine classifiers trained on different features divided based on hierarchical abstraction levels [11] or the type of information contained [3]. Our proposed approach is similar and combines classifiers trained on different sources of information. Most previous works, however, employ fairly straightforward fusion schemes to combine classifiers. Our proposed ensemble of a classifiers-based approach, on the other hand, uses a more advanced learning algorithm called dLEARNIN, inspired strongly by Learn++ [18], [20] which, in turn, has been shown to perform more effective data fusion than standard approaches.

Several approaches in pattern recognition have been developed for data fusion, for which ensemble-based approaches constitute a relatively new breed of algorithms. Traditional methods are generally based on probability theory (Bayes theorem, Kalman filtering), or decision theory such as the Dempster–Schafer (DS) theory and its many variations. The majority of these algorithms has been developed in response to the needs of military applications, most notably, target detection and tracking [21]–[23]. Ensemble of classifiers-based approaches seek to provide a fresh and more general solution for a broader spectrum of applications. Such approaches include simpler combination schemes, such as majority vote, threshold voting, averaged Bayes classifier, maximum/minimum rules, and linear combinations of posterior probabilities [24], [25]. More complex data-fusion schemes are also widely used, including ensemble-based variations of DS, template matching, neural and fuzzy systems, and stacked generalization [26]–[33]. Another related approach to data fusion using classifier combination schemes is input decimation: the use of different feature subsets in multiple classifiers [32], [33]. Input decimation can be useful in allowing different modalities, such as Fourier coefficients and pixel averages, to be naturally grouped together for independent classifiers [32]. Input decimation can also be used to lower the dimensionality of the input space by weeding out features that do not carry strong discriminating information [33].

Learn++ has been shown to be a useful addition to this list of techniques [18], [20] and provides a more general structure capable of using a variety of different classifier architectures, and has the ability to combine their outputs for the following:

1) a stronger overall classifier;
2) incremental learning;
3) multisensor data fusion.

In this paper, we demonstrate the use of dLEARNIN, a version of Learn++ modified for intrusion detection, as a strong classifier, and more important, to achieve effective data fusion for intrusion detection; however, it could be applied for adaptive incremental learning as well, which is very relevant to a network traffic scenario.

C. Cost Minimization Approaches

Different costs are associated with different errors in intrusion detection. However, most of the pattern recognition and machine-learning algorithms applied to intrusion detection have not been geared toward minimizing the cost of the errors. Ad-hoc methods are often used to pick classifiers or algorithm parameters until low cost is achieved; however, the algorithms do not explicitly employ any strategies to directly minimize the cost.

Apart from intrusion detection, several other applications, such as biometric recognition, etc. have different costs associated with different errors. However, most often, unprincipled ad-hoc techniques are used to deal with these costs. Slightly more principled techniques include using the receiver operating characteristic (ROC) curves or precision-recall curves and choosing a desired operating point. However, this works only for two class problems, such as verification problems in biometric recognition [34] and there are no fully developed extensions of ROC curves to multiple classes [35]. Algorithms, such as cost-sensitive Adaboost [36], have also been proposed; however, if the involved costs vary, they require retraining the entire classification system.

Our proposed cost minimization strategy dCMS, briefly introduced in [37], provides a principled approach that

1) can deal with multiple class problems;
2) is a simple post-training step and, hence, does not require retraining of classifiers if the costs involved vary;
3) is classifier independent and can be applied to any classification system that provides a confidence score in its classification.

In this paper, we demonstrate the use of dCMS to minimize the true object function for intrusion detection scenarios, the classification error costs, as opposed to the error rate itself.

III. Data Fusion: dLEARNIN

As stated before, intrusion detection naturally lends itself into a data-fusion scenario where it is beneficial to combine information from multiple sources. Our proposed algorithm to do so, dLEARNIN, is inspired by Learn++ [18]. Learn++ trains a multiclass ensemble of classifiers for each source of information, and further combines these ensembles of classifiers to achieve data fusion. dLEARNIN uses a slightly different approach. It uses a hierarchical ensemble of a classifiers-based algorithm, as shown in Fig. 1. We train a binary classification system for each C class—where one of the classes may be the normal traffic, and other classes may be the different types of attacks. Each classification system combines information from the multiple sources of information in a similar fashion as Learn++, and

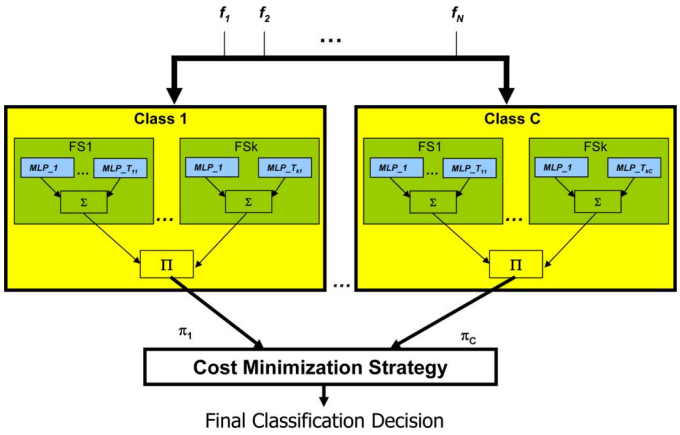

Fig. 1. Layout of the proposed classification system dLEARNIN: a hierarchical ensemble of classifiers.

these C classification systems are further combined. The details are explained next. More differences between dLEARNIN and Learn++ will be pointed out as we proceed.

A. Learning

We will explain the learning algorithm in detail for the classification system corresponding to one class, which is repeated for all classes. Let the different sources of information (feature sets) available be denoted as FS_k, $k = 1, 2, \ldots, K$, where K is the total number of data sources. An ensemble of classifiers is trained for each individual feature set and then combined. This is because the ensemble of classifiers has been shown to provide higher classification accuracies than single classifiers [19]. The learning algorithm adds a new classifier to the ensemble at each iteration. The following algorithm is the one to learn these ensemble of classifiers corresponding to each feature set, which is repeated for all feature sets.

The inputs to the algorithm are 1) the training data S_k comprised of m_k instances $x_i \in R^{N_k}$ along with their correct labels $y_i = +, -$ or $y_i = 1, 0$, $i = 1, \ldots, m_k$, where N_k is the number of features in the feature set FS_k; 2) the validation data V_k comprised of l_k instances along with their correct labels $y_i = +, -$, $i = 1, \ldots, l_k$; 3) a supervised classification algorithm BaseClassifier, generating individual classifiers (henceforth, hypotheses); and 4) an integer T_k, the number of classifiers to be generated in the ensemble corresponding to FS_k. The suffix k will be dropped here on unless ambiguous. The BaseClassifier can be any supervised classifier, such as an MLP, a support vector machine, or a decision tree; the only restriction is that it should provide higher than 50% accuracy on the validation data (better than random). MLPs were used here. In general, as T_k increases, the performance increases and then flattens out. On rare occasions, due to overfitting artifacts, the performance may decrease if T_k increases beyond a certain range. However, this is not characteristic of boosting algorithms, such as Adaboost, Learn++, and dLEARNIN, especially due to several modifications made to Learn++ incorporated in dLEARNIN. The value of T_k can be picked using cross-validation.

During the tth iteration, the BaseClassifier is trained on a selected subset of the training data to generate hypothesis h_t. This training subset TR_t is drawn from the training data according to a distribution D_t, which itself is obtained by normalizing a set of weights w_t maintained on the training data. The distribution D_t determines which instances of the training data are more likely to be selected into the training subset TR_t. Unless *a priori* information indicates otherwise, this distribution is initially set to be uniform, by initializing $w_1(i) = (1)/(m) \quad \forall i = 1, \ldots, m$, giving equal probability to each instance to be selected into the first training subset. At each subsequent iteration t, the weights previously adjusted at iteration $t-1$ are normalized to ensure a legitimate probability distribution D_t

$$D_t(i) = \frac{w_t(i)}{\sum_{i=1}^{m} w_t(i)} \quad \forall i = 1, \ldots, m. \tag{1}$$

Training subset TR_t is drawn according to D_t, and the BaseClassifier is then trained on TR_t. A hypothesis h_t is generated by the t_{th} classifier, whose error ϵ_t is computed on the entire validation data V as the proportion of the misclassified instances

$$\epsilon_t = \sum_{i=1}^{l} [|h_t(x_i) \neq y_i|] \tag{2}$$

where $[|\cdot|]$ evaluates to 1 if the predicate holds true, and 0, otherwise. This measure is different from that used by Learn++ where the error is computed on the training data S instead of the separate validation set V. This may be necessary in scenarios where sufficient data are not available; however, this is certainly not the case for our application. Moreover, Learn++ uses the sum of the weights assigned by D_t to the misclassified instances as the error measure. This is not desirable because, as we will see later, the instances with a higher weight are those that have not been learned yet (not picked in the training subset TR_t), or those that are difficult to learn. So the error measure, which will, in turn, be used to quantify the strength of the classifier, should not be higher for misclassifying these instances (if anything, it should be lower). Since we use a validation set V to compute a more representative error measure, we do not have access to the values of D_t for these data points. Hence, we simply use the proportion of misclassified instances in V as the error measure, which we found gives better results than the measure employed by Learn++. As mentioned before, we insist that this error be less than 0.5. If this is the case, the hypothesis h_t is accepted and the error is normalized to obtain

$$\beta_t = \frac{\epsilon_t}{1 - \epsilon_t}, 0 < \beta_t < 1. \tag{3}$$

If $\epsilon_t > 0.5$, then the current hypothesis is discarded, and a new training subset is selected to train a new hypothesis. All hypotheses generated thus far are then combined using the weighted sum rule [25] to obtain the composite hypothesis H_t. Let the outputs of the individual classifiers generated this far be ρ_t and η_t, where ρ_t is the positive score that can be interpreted as the tth classifier's confidence that the instance is positive, and η_t is the respective negative score. ρ_t and η_t are normalized to sum to 1. The weighted sum rule gives us

$$P_t = \frac{\sum_{i=1}^{t} \rho_i \log\left(\frac{1}{\beta_i}\right)}{\sum_{i=1}^{t} \rho_i \log\left(\frac{1}{\beta_i}\right) + \sum_{i=1}^{t} \eta_i \log\left(\frac{1}{\beta_i}\right)} \tag{4}$$

$$N_t = \frac{\sum_{i=1}^{t} \eta_i \log\left(\frac{1}{\beta_i}\right)}{\sum_{i=1}^{t} \rho_i \log\left(\frac{1}{\beta_i}\right) + \sum_{i=1}^{t} \eta_i \log\left(\frac{1}{\beta_i}\right)}. \quad (5)$$

The composite hypothesis thus becomes

$$H_t = [|P_t > N_t|] \quad (6)$$

where H_t now represents the ensemble decision, and classifies an instance as positive if $P_t > N_t$. The weighted sum rule used is an undemocratic but useful combination procedure: each hypothesis is assigned a weight as the logarithm of the reciprocal of its normalized error. Therefore, those hypotheses with smaller validation error are awarded a higher voting weight and, thus, have more say in the final classification decision. The logarithm function is used to map a wide range of $(1)/(\beta)$ values to a smaller and more meaningful interval. This combination rule is different than that used by Learn++, which employs a weighted majority voting scheme. A classifier combination rule that incorporates the confidence of the classifiers for the different classes instead of just their vote for a class clearly incorporates more information. Kittler *et al.* [25] present a theoretical analysis to demonstrate that sum rules are more effective when the classifiers to be combined have noisy outputs, while the product rules are more effective when combining strong classifiers. Hence, at this stage in the algorithm, we use the weighted sum rule to combine the weak classifiers. At later stages in the algorithm, when combining stronger classification systems, we use a weighted product rule. Moreover, we found empirically that these weighted sum and product rules perform better than the weighted majority voting scheme, more so since each classifier is a binary classifier.

The error of the composite hypothesis H_t is then computed as the sum of the distribution weights of the training data instances in S that are misclassified by the ensemble decision H_t

$$E_t = \sum_{i=1}^{m} D_t(i)[|H_t(x_i) \neq y_i|]. \quad (7)$$

If $E_t > 0.5$, then the current hypothesis is discarded, and a new training subset is selected to train a new hypothesis. If this is not the case, the composite error is normalized to obtain

$$B_t = \frac{E_t}{1 - E_t}, 0 < B_t < 1 \quad (8)$$

and is used for updating the distribution weights assigned to individual instances

$$w_{t+1}(i) = w_t(i) \times B_t^{1-[|H_t(x_i) \neq y_i|]}. \quad (9)$$

Equation (9) indicates that the distribution weights of the instances correctly classified by the composite hypothesis H_t are reduced by a factor of B_t $(0 < B_t < 1)$. Effectively, this increases the weights of the misclassified instances making them more likely to be selected in the training subset of the next iteration.

It should be noted that (7) is computed on the training data S and not the validation data V as in (2) because the value of E_t is used to determine which instances in the training data should be more likely to be picked in the next iteration. So the higher the weight of the instances misclassified (i.e., the harder the instances or the more neglected the instances so far), the higher their likelihood is to be picked in the next iteration. Due to this, E_t must be computed based on the training data and not a validation dataset. And since E_t is not used to quantify the strength of a classifier in any way, this is appropriate.

Once all of the T_k classifiers are generated for this kth feature set, the error of this ensemble of T_k classifiers on its validation data V_k is computed

$$\gamma_k = \sum_{i=1}^{l} [|H_{T_k}(x_i) \neq y_i|]. \quad (10)$$

Since the errors of the individual classifiers on the validation data are less than 0.5, the composite error is also less than 0.5. The normalized error is calculated as

$$\alpha_k = \frac{\gamma_k}{1 - \gamma_k}, 0 < \alpha_k < 1 \quad (11)$$

which will be used to determine the weight of this kth feature set. Again, as with earlier error measures, these are different from those employed by Learn++.

The aforementioned procedure is repeated for all K feature sets and, in turn, for all C classes.

B. Testing

When a test instance is presented, the scores for this instance corresponding to each K feature set for a particular class c are computed using (4) and (5) using all T_k classifiers trained for each kth feature set. These are combined using the weighted product rule to obtain an aggregate score for this instance belonging to the class c as follows:

$$\pi_c = \frac{\prod_{k=1}^{K} P_{T_k} \log\left(\frac{1}{\alpha_k}\right)}{\prod_{k=1}^{K} P_{T_k} \log\left(\frac{1}{\alpha_k}\right) + \prod_{k=1}^{K} N_{T_k} \log\left(\frac{1}{\alpha_k}\right)} \quad (12)$$

$$\mu_c = \frac{\prod_{k=1}^{K} N_{T_k} \log\left(\frac{1}{\alpha_k}\right)}{\prod_{k=1}^{K} P_{T_k} \log\left(\frac{1}{\alpha_k}\right) + \prod_{k=1}^{K} N_{T_k} \log\left(\frac{1}{\alpha_k}\right)}. \quad (13)$$

It can be seen that the contribution of each feature set is weighted inversely as its normalized error on the validation data and, hence, the stronger sources of information have a bigger say in the final decision.

Learn++ uses a weighted majority voting scheme at this stage as well, while we use the weighted product rule, which we found to be more effective. As stated before, we use the weighted product rule at this stage instead of the weighted sum rule as in (4) and (5) because at this stage, the classifiers being combined are ensemble of classifiers and, hence, are strong while they were weak individual classifiers in (4) and (5). And as demonstrated in [25], the weighted sum rule is more robust for weak classifiers; however, the weighted product rule is more appropriate for combining strong classifiers.

It should be noted that if T_{kc} is the number of classifiers corresponding to the kth feature set and cth class, then the number of classifiers through which the test instance is run is $\sum_{k=1}^{K} \sum_{c=1}^{C} T_{kc}$. The complexity of the testing depends on the complexity of testing the individual base classifiers.

If the costs of both errors (false positive and false negative) for all classes were equal, the instance would be assigned to class c if $\pi_c > \mu_c$ or $\pi_c > 0.5$ since π_c and μ_c sum to 1. However, since there are different costs associated with different errors, this classification rule is not the optimum strategy to make classification decisions and this threshold needs to be altered in order to minimize the cost. For instance, if the cost of a false positive for class c is twice that of a false negative, we would want to assign an instance of class c only if we are very confident that it belongs to class c. Otherwise, we would rather err on the side of a false negative since it has a lower cost. So we would want to assign an instance to class c only if $\pi_c > 2\mu_c$ for instance. A principled understanding of this is provided by the Bayes classification rule for costs [38]. The goal is to incorporate such intelligence in making this final classification decision, and gearing it toward minimizing the cost of errors instead of maximizing the classification rate, since cost is the true performance metric.

IV. Cost Minimization: dCMS

In order to minimize the cost, we need to determine C optimum thresholds for each C class to which π_c can be compared. If we can determine these, the class corresponding to the value of π_c that exceeds its respective threshold can be picked as the classification decision. If none or multiple outputs exceed their thresholds, voting resolution techniques, such as weighted majority voting, etc. [25] can be employed. We simply pick the class whose score exceeds its corresponding optimum threshold the most.

However, the cost matrix provided to us is a $C \times C$ cost matrix, and jointly determining the C optimum thresholds we are interested in is not straightforward. We first need to break the $C \times C$ cost matrix down into C 2×2 cost matrices. The procedure for this is described next.

Since the classification system has been trained, a confusion matrix can be computed on validation data using initial values of thresholds to be 0.5. Let the computed $C \times C$ confusion matrix be

$$\begin{bmatrix} n_{11} & n_{12} & \dots & n_{1C} \\ n_{21} & n_{22} & \dots & n_{2C} \\ \dots & & & \\ n_{C1} & n_{C2} & \dots & n_{CC} \end{bmatrix}$$

where n_{ij} is the number of instances that belong to Class i but were classified as Class j. $i, j \in \{1, 2, \dots, C\}$.

Using this confusion matrix, the probability of a misclassified instance classified as Class c actually belonging to Class i (one type of error contributing to the false positive rate for Class c), p_{ic} can be computed as

$$p_{ic} = \frac{n_{ic}}{\sum_{j=1}^{C} n_{jc}, j \neq c}. \tag{14}$$

Let the provided $C \times C$ cost matrix be

$$\begin{bmatrix} 0 & s_{12} & \dots & s_{1C} \\ s_{21} & 0 & \dots & s_{2C} \\ \dots & & & \\ s_{C1} & s_{C2} & \dots & 0 \end{bmatrix}$$

where s_{ij} is the cost of classifying an instance that belongs to Class i as Class j. Here, without loss of generality, the cost of correct classifications s_{ii} is assumed to be zero for all classes i, $j \in \{1, 2, \dots, C\}$.

The expected cost of a false positive for Class c, $E[\phi_c]$ can be computed using the aforementioned cost matrix and (15)

$$E[\phi_c] = \sum_{i=1}^{C} p_{ic} \cdot s_{ic}, i \neq c. \tag{15}$$

Similarly, the expected cost of a false negative for Class c, $E[\nu_c]$ can be computed.

Hence, we now have a 2×2 cost matrix for Class c of the form

$$\begin{bmatrix} 0 & E[\nu_c] \\ E[\phi_c] & 0 \end{bmatrix}.$$

Doing this for all C classes, we can break the $C \times C$ cost matrix into C 2×2 cost matrices.

Having determined these C 2×2 cost matrices, the problem breaks down to individually finding C optimum thresholds for each C score. The following strategy is used to determine each C threshold.

Consider class c. The 2×2 cost matrix for class c provides us with the ratio of the expected cost of false positives over the expected cost of false negatives for this class. Let this ratio be δ_c.

Suppose the histogram of the π_c scores for the instances that belong to class c are $N_c^+(t)$ and the histogram of π_c for the instances that do not belong to class c are $N_c^-(t)$. For a given value of threshold $t = \tau$, the number of false positives and the number of false negatives is given, respectively, by $\sum_{t>\tau} N_c^-(t)$ and $\sum_{t<\tau} N_c^+(t)$.

Hence, the cost incurred for this value of the threshold $t = \tau$ is

$$\kappa_c(\tau) = \delta_c \sum_{t>\tau} N_c^-(t) + \sum_{t<\tau} N_c^+(t). \tag{16}$$

The optimum threshold (that minimizes the cost) is then

$$\tau_c^* = \arg\min_{\tau} \kappa_c(\tau). \tag{17}$$

Having determined the τ_c^* for all of the C classes, the classification decision for a test instance is then based on these new optimum thresholds and not 0.5. This produces a new confusion matrix, and the aforementioned algorithm is repeated iteratively until a convergence condition is met. In practice, we found that the very first iteration provides the maximum boost in performance, and subsequent iterations are not necessary.

It may be useful to compare our approach to Neyman–Pearson hypothesis testing. Our proposed approach relies on the Bayesian criterion for making minimum risk decisions, and is hence very related to Neyman–Pearson hypothesis testing. The Neyman–Pearson approach is more appropriate in scenarios (often binary classification problems) where the maximum allowable false positive rate α_{FP} is known, and we wish to maximize the detection rate given this constraint. For the multiclass intrusion detection problem that has a cost matrix associated with it, the explicit criterion that

should be minimized is the cost per instance, which lends itself naturally into the Bayesian criterion for making minimum risk decisions. Determining the α_{FP} parameter for each class is not intuitive in this setting.

V. Experimental Results

The experiments were conducted on the MIT-DARPA intrusion detection database [14], also used at the KDD 1999 context [13]. To our knowledge, it is the only publicly available intrusion detection dataset. It is a 41-feature database with more than five million data points. The 41 features can be grouped into three groups: intrinsic (9), traffic (13), and content (19) features. There are five classes: four for different types of networks attacks—DenialOfService (DOS), Probe, UserToRoot (U2R), RootToLocal (R2L), and one normal traffic.

Three sets of experiments were conducted:

1) to evaluate the effectiveness of dLEARNIN for data fusion for intrusion detection;
2) to evaluate the effectiveness dCMS for cost minimization for intrusion detection;
3) to compare our performance to performances of other pattern recognition approaches to intrusion detection reported in literature on this dataset.

The dLEARNIN algorithm was used to obtain five trained classification systems—each of which is designed to recognize the five classes. Each system was capable of combining information from the three sources of information available. For each source, 15 MLP neural networks were trained. The five classification systems were combined using dCMS. The cost matrix used was as provided with the dataset

	DOS	Probe	U2R	R2L	Normal
DOS	0	1	2	2	2
Probe	2	0	2	2	1
U2R	2	2	0	2	3
R2L	2	2	2	0	4
Normal	2	1	2	2	0

.

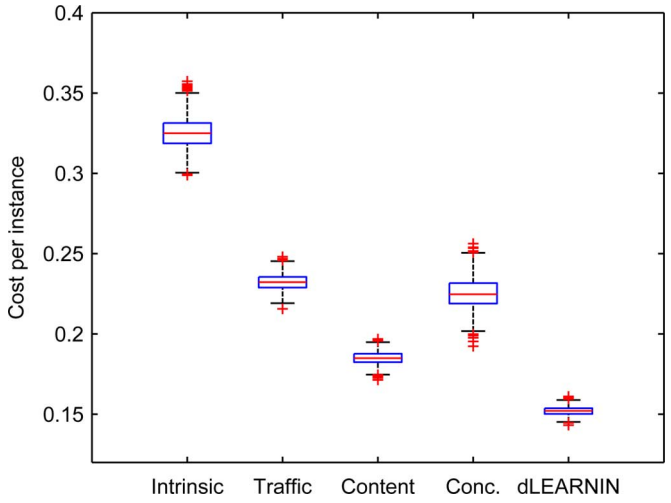

Fig. 2. Results obtained using our information fusion algorithm. It can be seen the cost per instance incurred using dLEARNIN is lower than that obtained by any individual feature set alone, or a naive concatenation of features.

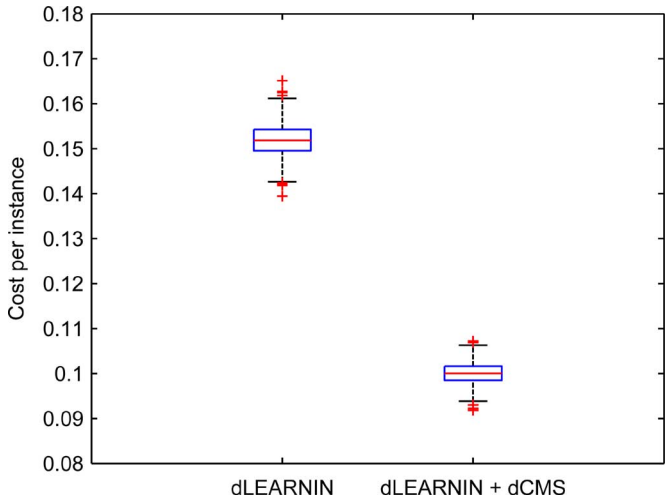

Fig. 3. Results obtained using our proposed cost minimization strategy. The cost incurred drops significantly by applying dCMS.

A. Data Fusion

To demonstrate the effectiveness of data fusion with dLEARNIN, we compare the cost per instance obtained using dLEARNIN to the individual feature sets alone. This was accomplished by using dLEARNIN for each individual feature set, as if there was only one source of information $K = 1$. We also compared the performance of data fusion by dLEARNIN to naive concatenation of features. Again, dLEARNIN was used on the feature sets concatenated, and treated as a single source of information. The cost minimization strategy dCMS was not used in order to demonstrate the data-fusion effects only.

The dataset was randomly split into 0.3 million data points for training and validation, and the rest for testing. The experiment was repeated 50 times with random splits of training, validation, and testing data, and the results obtained were summarized in Fig. 2. The average performance as well as the 95% confidence intervals are shown. It can be seen that data fusion with dLEARNIN provides better results than any of the individual feature sets and the naive concatenation, with statistical significance.

B. Cost Minimization

In order to demonstrate the effectiveness of the cost minimization strategy dCMS, we compare the results obtained using dLEARNIN alone, to that obtained by using dLEARNIN followed by dCMS. Again, the dataset was randomly split into 0.3 million data points for training and validation, and the rest for testing, and the experiment was repeated 50 times. The results obtained are summarized in Fig. 3. It can be seen that using dCMS can significantly reduce the costs incurred.

C. Comparison to State of the Art

The MIT-DARPA dataset [14] was used at the KDD'99 [13] context for a given split of training and testing data, and since then, several machine-learning algorithms have been applied to

TABLE I
COMPARISON OF OUR RESULTS TO THOSE OF OTHER ALGORITHMS REPORTED IN LITERATURE

Algorithm	Cost per instance
dLEARNIN + dCMS	0.224
dLEARNIN	0.228
Multi-classifier (class)	0.229
KDD 1999 winner	0.233
Learn++ + dCMS	0.233
Learn++	0.239
Multilayer perceptron	0.239
K-means clustering	0.239
Decision tree	0.240
Nearest cluster algorithm	0.247
Belief fuctions	0.248
Hyperspehere algorithm	0.249
Fuzzy ARTMAP	0.250
Leader algorithm	0.253
Bayesian average	0.296
Majority voting	0.344
Gaussian classifier	0.362

this split. For comparison, we run our proposed algorithm on the same split, and Table I [15] provides a comparison between the cost per instance value achieved using our algorithm and some of the best values reported in the literature. Several other algorithms and approaches have also been applied, however, only on subsets of the available data and, hence, do not form fair grounds for comparison here. It can be seen that the proposed algorithm dLEARNIN provides the best results reported so far, and with dCMS added on, the results further improve. It can also be seen that dLEARNIN provides better results that Learn++ using comparable paramters. It should be noted that although we achieve state-of-the-art results on the KDD dataset, the cost per instance values are much higher than those shown in Figs. 2 and 3, where the data were split randomly into training and testing. This indicates that the split of data into training and testing as provided by the KDD contest has significantly different distributions for training and testing data, which is hard for most statistical pattern recognition algorithms to learn. Our own initial analysis indicates this, and this is further confirmed by a more thorough analysis provided by [39].

VI. FUTURE WORK

There are several avenues for future work that we are pursuing. The different classifiers combined at different stages of dLEARNIN through sum and product rules are trained on different subsets of data and often on different features. Hence, they are likely to have different statistical properties, and their outputs are not directly comparable. So a transform that accounts for these differences in statistical properties and gives us better estimates of posterior probabilities to be used for more effective classifier combination rules is desirable. Also, other base classifiers can be used instead of MLPs. A thorough and formal analysis of the KDD dataset split would provide some insights into the difference in the distributions of the training and test data as provided. This would give us a better insight into what statistical pattern recognition paradigms are appropriate for this dataset, and which ones are not. Finally, the capabilities of Learn++ and dLEARNIN to learn incrementally and adaptively are very relevant to intrusion detection where the properties of the network traffic change dynamically. This should be further investigated.

VII. CONCLUSION

Our proposed algorithm addresses two characteristics of an intrusion detection problem. First, intrusion detection has much to gain from combining information from multiple sources of information in a meaningful way; and second, the true objective function to be minimized in an intrusion detection scenario is the cost of classification errors, and not the error rate itself. An ensemble of classifiers approach dLEARNIN was successfully implemented for intrusion detection to achieve efficient data fusion. The results obtained surpassed the best results reported thus far using pattern recognition algorithms for intrusion detection. A cost minimization strategy dCMS was applied to the outputs of the data-fusion algorithm, which reduced the cost per instance incurred with a statistically significant margin.

REFERENCES

[1] J. McHugh, A. Christie, and J. Allen, "Defending yourself: The role of intrusion detection systems," *IEEE Softw.*, vol. 17, no. 5, pp. 42–51, Sep./Oct. 2000.
[2] G. Giacinto and F. Roli, "Intrusion detection in computer networks by multiple classifier systems," in *Proc. Int Conf. Pattern Recognition*, 2002, pp. 390–393.
[3] L. Didaci, G. Giacinto, and F. Roli, "Ensemble learning for intrusion detection in computer networks," presented at the Workshop Machine Learning Methods Applications, Siena, Italy, Sep. 10–13, 2002.
[4] J. Allen, A. Christie, W. Fithen, J. McHugh, J. Pickel, and E. Storner, State of the Practice of Intrusion Detection Technologies Tech. Rep. CMU/SEI-99-TR-028, 2000.
[5] R. Duda, P. Hart, and D. G. Stork, *Pattern Classification*. New York: Wiley, 2001.
[6] W. Lee and S. J. Stolfo, "A framework for constructing features and models for intrusion detection systems," *ACM Trans. Inf. Syst. Security*, vol. 3, no. 4, 2000.
[7] G. Giacinto, F. Roli, and L. Didaci, "Fusion of multiple classifiers for intrusion detection in computer networks," in *Pattern Recognit. Lett.*, 2003, vol. 24, no. 12, pp. 1795–1803.
[8] S. C. Lee and D. V. Heinbuch, "Training a neural-network based intrusion detector to recognize novel attacks," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 31, no. 4, pp. 294–299, Jul. 2001.
[9] A. K. Ghosh and A. Schwartzbard, "A study in using neural networks for anomaly and misuse detection," in *Proc. USENIX Security Symp.*, 1999, p. 12.
[10] J. Cannady, "An adaptive neural network approach to intrusion detection and response," Ph.D. dissertation, School Comput. Inform. Sci., Nova Southeastern University, Fort Lauderdale, FL, 2000.
[11] J. M. Bonifkio, A. M. Cansian, A. C. P. L. F. de Carvalho, and E. S. Moreira, "Neural networks applied in intrusion detection systems," *Proc. IEEE World Congr. Computer Intelligence*, pp. 205–210, 1998.
[12] C. Elkan, "Results of the KDD'99 classifier learning," in *ACM Special Interest Group on Knowledge Discovery and Data Mining Explorations*, 1, 2000.
[13] *The UCI KDD Archive, Information and Computer Science*, [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html., Univ. California, Irvine.
[14] "MIT Lincoln Laboratory—DARPA Intrusion Detection Evaluation." [Online]. Available: http://www.ll.mit.edu/IST/ideval/data/data_index.html.

[15] M. Sabhnani and G. Serpen, "Application of machine learning algorithms to KDD intrusion detection dataset within misuse detection context," in *Proc. Int. Conf. Machine Learning, Models, Technologies and Applications*, 2003, pp. 209–215.
[16] G. Giacinto, F. Roli, and L. Didaci, "A modular multiple classifier system for the detection of intrusions in computer networks," in *Proc. IEEE Int. Conf. Multiple Classifier Systems*, 2003, pp. 346–355.
[17] G. Giacinto, R. Perdisci, and F. Roli, "Network intrusion detection by combining one class classifiers," presented at the Int. Conf. Image Analysis and Processing, Cagliari, Italy, 2005.
[18] R. Polikar, L. Udpa, S. Udpa, and V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks," *IEEE Trans. System, Man Cybern, C, Appl. Rev.*, vol. 31, no. 4, pp. 497–508, Nov. 2001, Special Issue on Knowledge Management.
[19] Y. Freund and R. Schapire, "A decision theoretic generalization of online learning and an application to boosting," *Comput. Syst. Sci.*, 1997.
[20] D. Parikh and R. Polikar, "An ensemble based incremental learning approach to data fusion," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 37, no. 2, pp. 437–450, Apr. 2007.
[21] D. Hall and J. Llinas, "An introduction to multisensor data fusion," *Proc. IEEE*, vol. 85, no. 1, pp. 6–23, Jan. 1997.
[22] , D. Hall and J. Llinas, Eds., *Handbook of Multisensor Data Fusion*. Boca Raton, FL: CRC, 2001.
[23] L. A. Klein, *Sensor and Data Fusion Concepts and Applications*. Belingham, WA: SPIE, 1999, vol. TT35.
[24] J. Grim, J. Kittler, P. Pudil, and P. Somol, "Information analysis of multiple classifier fusion," presented at the Int. Workshop on Multiple Classifier Systems, Cambridge, U.K., 2001.
[25] J. Kittler, M. Hatef, R. Duin, and J. Matas, "On combining classifiers," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 3, pp. 226–239, Mar. 1998.
[26] L. O. Jimenez, A. M. Morales, and A. Creus, "Classification of hyperdimensional data based on feature and decision fusion approaches using projection pursuit, majority voting and neural networks," *IEEE Trans. Geosci. Remote Sensors*, vol. 37, no. 3, pp. 1360–1366, May 1999.
[27] G. J. Briem, J. A. Benediktsson, and J. R. Sveinsson, "Use of multiple classifiers in classification of data from multiple data sources," in *Proc. IEEE Geoscience Remote Sensor Symp.*, 2001, vol. 2, pp. 882–884.
[28] F. M. Alkoot and J. Kittler, "Multiple expert system design by combined feature selection and probability level fusion," in *Proc. Int. Conf. FUSION*, 2000, vol. 2, no. 10, pp. THC5/9–THC516.
[29] D. Wolpert, "Stacked generalization," *Neural Netw.*, 1992.
[30] L. I. Kuncheva, "Switching between selection and fusion in combining classifiers: An experiment," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 32, no. 2, pp. 146–156, Apr. 2002.
[31] L. I. Kuncheva, "A theoretical study on six classifier fusion strategies," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 2, pp. 281–286, Feb. 2002.
[32] L. Kuncheva and C. Whitaker, "Feature subsets for classifier combination: An enumerative experiment," in *Proc. Int. Workshop on Multiple Classifier Systems*, 2001.
[33] N. Oza and K. Tumer, "Input decimation ensembles: Decorrelation hrough dimensionality reduction," presented at the Int. Workshop on Multiple Classifier Systems, Cambridge, U.K., 2001.
[34] A. Jain, A. Ross, and S. Prabhakar, "An introduction to biometric recognition," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 1, pp. 4–20, Jan. 2004.
[35] N. Lachiche and P. Flach, "Improving accuracy and cost of two-class and multi-class probabilistic classifiers using ROC curves," presented at the Int. Conf. Machine Learning, Washington, DC, 2003.
[36] Y. Ma and X. Ding, "Robust real-time face detection based on cost-sensitive adaboost method," in *Proc. IEEE Int. Conf. Multimedia Expo.*, 2003, pp. 465–468.
[37] D. Parikh and T. Chen, "Classification-error cost minimization strategy: dCMS," in *Proc. IEEE Statistical Signal Processing Workshop*, 2007, pp. 620–624.
[38] D. Duda, P. Hart, and D. Stork, *Pattern Classif.*, ser. 2/e. New York: Wiley, 2001, ch. 2, pp. 20–29.
[39] M. Sabhnani and G. Serpen, "Why machine learning algorithms fail in misuse detection on KDD intrusion detection data set," *Intell. Data Anal.*, 2004.

Devi Parikh (S'05) received the B.Sc. degree in electrical and computer engineering from Rowan University, Glassboro, NJ, in 2005 and the M.Sc. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, in 2007, where she is currently pursuing the Ph.D. degree.

Her research interests include computer vision, pattern recognition, and machine learning. Apart from using ensemble of classifiers for data fusion and cost minimization strategies for intrusion detection, she has worked on several pattern recognition and computer vision projects ranging from estimates of posterior probabilities from classifier outputs, to feature-based retrieval for 3-D reassembly, to unsupervised learning of relationships among objects in a scene, and studying the role of contextual information for enhanced image understanding.

Ms. Parikh is a member of the Society of Women Engineers (SWE), the Golden Key National Honor Society, the Order of Engineers, and the New Jersey Epsilon Honor Society. She has been awarded the Hazel P. Valiant Student award in 2002 and the Broome Alumni Undergraduate award in 2003 and 2004. She has also received the Best Paper Award at the Beyond Patches Workshop at the IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2007. She is the recipient of the Dean's Fellowship at Carnegie Mellon and a National Science Foundation Graduate Research Fellowship.

Tsuhan Chen (F'07) received the B.S. degree in electrical engineering from National Taiwan University, Taipei City, Taiwan, R.O.C., in 1987 and the M.S. and Ph.D. degrees in electrical engineering from the California Institute of Technology, Pasadena, in 1990 and 1993, respectively.

Currently, he is a Professor and Associate Department Head with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, where he has been since 1997. From 1993 to 1997, he was with AT&T Bell Laboratories, Holmdel, NJ. He was the Editor-in-Chief for the IEEE TRANSACTIONS ON MULTIMEDIA in 2002–2004. He was also on the Editorial Board of the *IEEE Signal Processing Magazine* and Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, the IEEE TRANSACTIONS ON IMAGE PROCESSING, the IEEE TRANSACTIONS ON SIGNAL PROCESSING, and the IEEE TRANSACTIONS ON MULTIMEDIA. He co-edited a book titled *Multimedia Systems, Standards, and Networks*.

Dr. Chen received the Charles Wilts Prize from the California Institute of Technology, Pasadena, in 1993 and was a recipient of the National Science Foundation CAREER Award from 2000 to 2003. He received the Benjamin Richard Teare Teaching Award at Carnegie Mellon University in 2006. He was elected to the Board of Governors, IEEE Signal Processing Society, from 2007 to 2009. He is a member of the Phi Tau Phi Scholastic Honor Society. He is a Distinguished Lecturer of the Signal Processing Society.