Network Intrusion Detection using Machine Learning and Voting techniques

Tich Phuoc Tran[1], Pohsiang Tsai[1], Tony Jan[1] and Xiaoying Kong[2]
[1] *Centre for Innovation in IT Services and Applications (iNEXT)*
University of Technology, Sydney
[2] *Centre for Real-time Information Networks (CRIN)*
University of Technology, Sydney

Keywords: Network Intrusion Detection, Neural Network, Adaptive Boosting, Multi-Expert Classification.

Abstract:
As the result of recent advent and rapid growth of the Internet, there have been an increasing number of corporations relying on computers and networks for communications and critical business transactions. Because of the network complexity and advanced hacking techniques, such reliance on computer networks often presents unanticipated risks and vulnerabilities. A huge volume of attacks on major sites and networks have been recently reported including those of private companies, government agencies and even military classified networks. Therefore, it is important to deploy protection measures for networks and their services from unauthorized modification, destruction, or disclosure of sensitive information. Intrusion detection systems (IDS) have emerged as an important part of today's network security infrastructure which can monitor the network traffic and detect possible attacks. Currently existing IDS suffer from low detection accuracy and system robustness for new and rare security breaches. To improve detection capability of IDS, this chapter proposes an innovative Machine Learning (ML) framework in which different types of intrusions will be detected with different classifiers, including different attribute selections and learning algorithms. Outputs of these classifiers are then combined by appropriate voting techniques. Experiments on the KDD-99 dataset show that our approach obtains superior performance in comparison with other state-of-the-art detection methods, achieving low learning bias and improved generalization at an affordable computational cost.

1. Introduction

As a result of the revolutionary advances in computing science and the wide spread deployment of the Internet, people are encouraged to communicate and exchange information over the computer-mediated environment. This provides convenience and benefits such as shortening the effective geographical distances and sharing information efficiently. On the other hand, information exchange in such environments pose a problem, which is that, intruders or malicious users may compromise the communications. The safeguarding of security is becoming even more difficult, because the possible technologies of attacks are very sophisticated; at the same time, less technical ability is required for the novice attackers due to easy access to proven past methods through the Web. A traditional approach to defend computer networks was based on static defense mechanisms in which software, such as the operating system, was kept up-to-date to prevent the exploitation of security holes; and the firewalls deployed at critical network segments to improve the security at the entry level. However, firewalls aim to regulate and control the flow of information into and out of a network rather than detecting whether or not the network is under attack. To complement simple firewalls, Intrusion Detection Systems (IDS) are normally used to gather and analyze network data to identify possible security breaches, which include both intrusions (attacks from outside the organization) and misuse (attacks from within the organization). Although IDS have become an important part of most network security architectures which provides essential second line of defense, the majority of them face a number of challenges such as low detection rates which can miss serious intrusion attacks and high false alarm rates, which falsely classifies a normal connection as an attack and therefore obstructs legitimate user access to the network resources (Sommer, 2008). These problems are due to the sophistication of the attacks and their intended similarities to normal behavior.

Most IDS utilize some Machine Learning (ML) techniques to obtain high detection capability for novel attacks and automation to save human labor from manually constructing signatures of attacks or specifying the normal behavior of a sensor node. Theoretically, it is possible for a ML algorithm to achieve the best performance, i.e. it can minimize the false alarm rate and maximize the detection accuracy; however, this normally requires infinite training sample sizes (Kononenko & Kukar, 2007). In practice, such condition is impossible due to limited computing resources and real-time response requirement of IDS. Intrusion detection, therefore, remains very challenging. In this chapter, a learning framework is proposed to enhance the performance of intrusion detection for rare and complicated attacks; that is, the framework can increase the detection accuracy and decrease false alarm with acceptable computational expenses. In particular, characteristics of different anomaly categories are captured using different strategies, also referred to as local experts, with different feature extraction schemes and advanced learning methods. The outputs of these experts are then fused by appropriate voting techniques. In addition to this framework, we also introduce a highly performing ML algorithm that combines a light-weight Radial Basis Function Neural Network and an Ensemble Learning technique. This algorithm is compared against other learning methods for the purpose of local expert creation. This work falls well under the category of bias-variance-computations tradeoff problem. In general, we wish to reduce bias (for higher accuracy), variance (for fewer false alarms) and computations (for fast real time response). An extensive empirical analysis conducted on the Knowledge Discovery and Data Mining (KDD-99) intrusion detection data suggests that the proposed framework obtains noticeable performance improvement compared with other state-of-the-art techniques, in terms of detection accuracy, system robustness and total cost.

This chapter starts with an overview of network intrusion detection technology and the related works of ML approaches for Network Security domain, followed by a study of the

Radial Basis Function Neural Network (RBKNN) family which has been reported for great successes in many applications. Emphasis is put on the Generalized Regression Neural Network (GRNN) and Vector-Quantized GRNN (VQ-GRNN) for their typical learning and system robustness properties. We also provide an overview of Ensemble Learning methods in which multiple classifiers are trained to solve the same problem and their decisions are then aggregated in some manner. Such methods can be used to boost predictive performance of some learning algorithms. Next, the Multiple-Expert Classification Framework (MECF) with implementation of advanced voting techniques is presented. The usefulness of this model is then illustrated through its application to the network intrusion detection problem, focusing on detection capability on rare attack categories. Finally, this chapter is concluded with future research direction discussed.

2. Intrusion Detection and Machine Learning techniques

2.1. Intrusion Detection Systems

As more and more corporations rely on computers and networks for communications and critical business transactions, securing digital information has become one of the largest concerns of the business community. A powerful security system is not only a requirement but essential to the livelihood of enterprises (Kemmerer & Vigna, 2002). In recent years, there has been a great deal of research conducted in this area to develop intelligent and automated security tools which can fight the latest cyber attacks. The security achieved must be reasonable yet sufficient, balancing needs for accountability with equally important needs for privacy and accessibility. Alongside the existing techniques for preventing intrusions such as encryption and firewalls, Intrusion Detection technology has established itself as an emerging research field that is concerned with detecting unauthorized access and abuse of computer systems from both internal users and external offenders. An Intrusion Detection System (IDS) is defined as a protection system that monitors computers or networks for unauthorized activities based on network traffic or system usage behaviors, thereby detecting if a system is targeted by a network attack such as a denial of service attack (McHugh, Christie, & Allen, 2000). In response to those identified adversarial transactions, IDS can inform relevant authorities to take corrective actions.
There are a large number of IDS available on the market to complement firewalls and other defense techniques. These systems are categorized into two types of IDS, namely (1) misuse-based detection in which events are compared against pre-defined patterns of known attacks and (2) anomaly-based detection which relies on detecting the activities deviating from system "normal" operations.

2.2. Application of Machine Learning to Intrusion Detection

Artificial Intelligence (AI) is the key technology in many of today's novel applications, ranging from banking systems that detect attempted credit card fraud or a robot that can sense and respond to human emotions, to software systems that can work as a human expert to offer appropriate advice when needed. These technologies would not exist without the knowledge gained from AI research. As a major part of AI, Machine Learning (ML) refers to algorithmic mechanisms that allow computers to learn from experience, examples and analogy (Mitchell, 1997). The output of this learning process is actionable knowledge that can be used to solve a specific problem. In the case of Intrusion Detection, learning involves discovering patterns of normal behavior or intrusive behavior by analyzing the sample data of such activities. This sample data is also called a training set. It should be sufficient to represent the whole population of patterns to be discovered. The learned models can then be used to make classification on a new data instance based on its similarity to normal behavior (anomaly detection) or known attack signatures (misuse detection). Many security systems have implemented ML to achieve a generalization capability from limited training data. That means, given known intrusion signatures, a security system should be able to detect similar or new attacks.

2.2.1. Related works

One of the rule-based methods which is commonly used by early IDS is the Expert System (ES) (Bauer & Koblentz, 1988; Ilgun, 1995). In such systems, the knowledge of human experts is encoded into a set of rules. This allows more effective knowledge management than that of a human expert in terms of reproducibility, consistency and completeness in identifying activities that match the defined characteristics of misuse and attacks. However, ES suffers from low flexibility and robustness. Unlike ES, Data Mining approach derives association rules and frequent episodes from available sample data, not from human experts. It utilizes statistical techniques to discover subtle relationships between data items, and from that, constructs predictive models. Using the derived rules, Lee et. al. developed a data mining framework for the purpose of intrusion detection (W. Lee, Stolfo, & Mok, 1999a, 1999b). In particular, system usage behaviors are recorded and analyzed to generate rules which can recognize misuse attacks. The drawback of such frameworks is that they tend to produce a large number of rules and thereby, increase the complexity of the system.

Decision Trees are one of the most commonly used supervised learning algorithms in IDS (Amor, Benferhat, & Elouedi, 2004; J.-H. Lee, Lee, Sohn, Ryu, & Chung, 2008; Levin, 2000a; V. Miheev, Vopilov, & Shabalin, 2000; Pfahringer, 2000a) due to its simplicity, high detection accuracy and fast adaptation. Another highly performing method is Artificial Neural Networks (ANN) which can model both linear and non-linear patterns. The resulting model can generate a probability estimate of whether given data matches the characteristics that it has been trained to recognize. Cannady (1998) developed a network-based detection system in which 9-packet-level network data was retrieved from the RealSecure database and then classified by a feed-forward neural network (Cannady, 1998). Though this prototype is not a complete IDS, the results clearly demonstrate the potential of an ANN in detecting network attacks. Latter ANN-based IDS (Mukkamala, 2002; Zhang, Li, Manikopoulos, Jorgenson, & Ucles, 2001) have reportedly achieved great successes in detecting difficult attacks. For unsupervised intrusion detection, data clustering methods can be applied (Shah, Undercoffer, & Joshi, 2003). These methods involve computing a distance between numeric features and therefore they cannot easily deal with symbolic attributes, resulting in inaccuracy. Another well-known ML technique used in IDS is Naïve Bayes classifiers (Amor et al., 2004). Because Naïve Bayes assumes the conditional independence of data features, which is often not the case for intrusion detection, correlated features may degrade its performance. In (Kruegel, Mutz, Robertson, & Valeur, 2003), the authors apply a Bayesian network for IDS. The network appears to be attack specific and its size grows rapidly as the number of features and attack types increase. Beside popular decision trees and ANN, Support Vector Machines (SVMs) are also a good candidate for intrusion detection systems (Ambwani, 2003) which can provide real-time detection capability, deal with large dimensionality of data. SVMs plot the training vectors in high dimensional feature space through nonlinear mapping and labeling each vector by its class. The data is then classified by determining a set of support vectors, which are members of the set of training inputs that outline a hyperplane in the feature space. SVMs are scalable as they are relatively insensitive to the number of data points (Ambwani, 2003).

2.2.2. The Knowledge Discovery and Data Mining Benchmark

Current security systems are facing two fundamental challenges. First, the unbalanced nature of security dataset indicates dramatic changes in the distribution of classes compared with the normal trends i.e., some classes dominate others with their overwhelming occurrences (Kemmerer & Vigna, 2002). This will bias the resultant predictive models to favor the dominant classes. Second, increased dimensionality, especially when noise is involved, can degrade learning significantly (Kemmerer & Vigna, 2002). Together, these two characteristics make the detecting intrusive activities very challenging.

In order to facilitate the comparison of advanced research in the area of network security, the Lincoln Laboratory at the Massachusetts Institute of Technology (MIT) conducted the 1998 and 1999 evaluations of intrusion detection (McHugh et al., 2000). Funded by The Defense Advanced Research Projects Agency (DARPA), the purpose of the evaluation program is to provide a basis for making comparisons of existing IDS under a common set of circumstances and assumptions. Data obtained from these programs were then used as the benchmark training and test data sets for "Classifier Learning Contest" organized in conjunction with the 5th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining in 1999 (KDD-99).

The KDD-99 dataset has simulated the two challenging problems discussed earlier, namely, the unbalanced nature and high dimensionality of security data. It contains 7 weeks of training traffic data and 2 weeks of testing data (McHugh et al., 2000). Preprocessing was applied to abstract and summarize the raw tcpdump data to form network connections.

a) Attack types and categories

Each connection record in the KDD-99 dataset is labeled as either normal or one type of attack. There are totally 39 types of attacks which are grouped into 4 major categories (McHugh et al., 2000): Probe, Denial of Service (DoS), User to Root (U2R) and Remote to Local (R2L). In particular, Probe attacks refer to the incidents in which some malicious programs can automatically scan a network of computers to gather sensitive information or search for security vulnerabilities while a DoS attack prevents normal use of network resources for legitimate purposes by consuming the bandwidth or overloading the computational resources of the victim system. The R2L attacks occur when an intruder who has no valid account on a machine can exploit some system vulnerabilities to gain local access as a legitimate user by sending packets over a network. In contrast, U2R attacks assume that the attacker has already access to a system as a normal user and he can exploit some security holes to gain user root privileges.

b) Features

41 features were used to summarize the connection information. These features are grouped as basic features and additional features respectively (McHugh et al., 2000).

Basic Features
Bro is used as the network analyzer to derive the 9 basic features from packet headers without inspecting the packet contents (Kemmerer & Vigna, 2002). Some examples of basic features include duration of connection, protocol types and service types.
Additional Features

- *Content features*: The payload of TCP packets is assessed by applying the domain knowledge. Examples of content-based features include the number of unsuccessful logins and whether the root access was gained or not.
- *Time based features*: It is important to inspect the packets within some time interval to cope with the temporal nature of network attacks. These features are designed to capture properties within a 2 second temporal window. Number of connections to the same host is an example of time-based features.
- *Host based features*: Utilize a historical window estimated over the number of connections (100 connections in KDD-99) instead of time. Host based features are therefore used to assess attacks which span over intervals longer than 2 seconds.

c) Learning methods that use KDD-99 dataset

Among intrusion detection models tested on KDD-99 dataset, most of them are reported to provide unacceptably low detection capability for U2R and R2L attacks. Some typical examples of such models include a rule-based predictive model (PNrule) (Agarwal, 2000) which is studied to effectively detect DoS and Probe attacks; the winning entry of KDD99 contest (Pfahringer, 2000b) which is composed from 50×10 C5 decision trees fused by cost-sensitive bagged boosting. Similar techniques are also developed such as a decision tree forest constructed by Kernel Miner (KM) tool (Levin, 2000b) and two layers of voting decision trees augmented with human security expertise (V. Miheev, Vopilov, A. Shabalin, I., 2000). Due to poor performance of these approaches on some sophisticated attacks, we are motivated to develop a new learning method to improve the overall detection performance on KDD 99 benchmark.

3. Artificial Neural Network and Ensemble Learning

Unlike other pattern recognition tasks which may sacrifice accuracy for system robustness and stability, Intrusion Detection requires very high accuracy which implies both high detection rate and low false alarm rate (Sommer, 2008). An undetected intrusion can cause serious damage to computer networks. In this regards, high detection accuracy is of great importance for new security systems. In addition to accuracy, security systems must be also fast enough not to cause bottlenecks in communication networks. That is, network administrators should be alerted that their systems have been penetrated or have been used as springboards for attacks on other systems right after the incidences have occurred. In general, security system with high accuracy requires heavy computations. In our approach, we develop a system that achieves high accuracy for real time IDS but requires relatively small computational complexity. This ensures that the systems can both perform accurately and respond to incidences in a timely fashion.

3.1. Bias-Variance Dilemma

Though several ML techniques have been adopted in the Network Security domain with certain success, there remain performance limitations including low detection accuracy and

high false alarm rates, especially for rare and complicated attacks. For instance, the winning entries of KDD-99 competition do not provide satisfactory performance on U2R and R2L attack categories due to their low frequency and complicated nature. Several learning methods have been developed to increase the detection capability including ANN models. As a flexible “model-free" learning method, ANN can fit training data very well and thus provide a low learning bias. However, they are also susceptible to the overfitting problem, which can cause instability in generalization (Mitchell, 1997). This degraded performance is the consequence of the overfitting or overtraining problem, in which data sensitivity causes the resulting classifier to have small bias but large variance.
The learning bias is defined as the measure of how accurately the model fits the available sample data while the generalization variance measures how stably the model performs for prediction or classification tasks (Mitchell, 1997). To avoid overfitting, some methods which are less dependent on available data are introduced, but they may misrepresent the true functional relationships and have a large bias. The bias and variance hence are said to be inversely related (Mitchell, 1997), i.e. with a fixed data set, reducing one will inevitably cause the other to increase.
Some approaches are proposed to improve the generalization stability by reducing generalization variance at the cost of higher learning bias, i.e. allowing underfitting. This would deteriorate the overall performance to a certain level. In critical modeling applications, underfitting is not acceptable because a miss in detection may be very costly, i.e. causing the whole computer network compromised. Therefore, a sensible detection system which can achieve both stable generalization and accurate data learning is very much desirable. Theoretically, both bias and variance may be reduced at the same time given infinite sized models. Nevertheless, this condition is generally infeasible since the model complexity must be limited in real life. In this research, we seek a compromise solution which can retain the desirable data-fitting capacity of ANN while reducing generalization variance at a minimal computational cost. A learning algorithm is proposed by combining a radial basis function neural network with an adaptive boosting method. An overview of these relevant technologies is provided in the next 2 sections.

3.2. Overview on VQ-GRNN

A family of ANN models, RBFNN, has recently drawn great research attention due to its good generalization ability and a simple network structure that avoids unnecessary and lengthy calculations as compared to the Multilayer Feedforward Networks (MFN) (Zaknich, 2003). Considering the node characteristics and the training algorithms, RBFNN are very different from MFN. The node characteristics for MFN are usually chosen as sigmoidal functions while for RBFNN, as indicated in the name, radial basis functions are employed. A popular algorithm in RFBNN family is the Generalized Regression Neural Network (GRNN) proposed by Specht (Spetch, 1991) which contains a hidden layer of radial units. Each radial unit models a Gaussian response surface which can be determined by its center point and a radius. Because these functions are nonlinear, it is enough for a single hidden layer to describe any shape of function. The output of these Gaussians is then linearly weighted to produce the desirable response. The following is the general form of GRNN:

$$\hat{y}(\underline{x}) = \frac{\sum_{n=1}^{NV} y_n f_n(\underline{x} - \underline{x}_n, \delta)}{\sum_{n=1}^{NV} f_n(\underline{x} - \underline{x}_n, \delta)} \tag{1}$$

Where

$\underline{x}$: Input vector (under line refers to vector)

$\underline{x}_n$: All other training vectors in the input space

δ : Single smoothing parameter chosen during network training

y_n : Scalar output related to $\underline{x}_n$

NV : Total number of training vectors

In many applications, GRNN provides high accuracy. However, it is computationally expensive as well as sensitive to the selection of variances for smoothing functions. In fact, GRNN incorporates each and every training example $\{\underline{x}_i \rightarrow y_i\}$ into its architecture, i.e. all of the training vectors needs to be processed and Gaussian function's parameters such as centers and variance will need to be computed with respect to all other surrounding vectors. In order to overcome this problem, an approximation of GRNN, namely, the Vector Quantized – Generalized Regression Neural Network (VQ-GRNN) is proposed by Zaknich (Zaknich, 1998) for application to general signal processing and pattern recognition problems. VQ-GRNN is a generalization of Probabilistic Neural Network (PNN) and is related to Generalized Regression Neural Network (GRNN) classifiers (Spetch, 1991). In particular, this method approximates GRNN by quantizing the data space into clusters and associate a specific weight for each of these clusters.

If there exists a corresponding scalar output y_i for each local region (cluster) which is represented by a center vector $\underline{c}_i$, then a GRNN can be approximated by a VQ-GRNN formulated as follow:

$$\hat{y}(\underline{x}) = \frac{\sum_{i=0}^{M} Z_i y_i f_i(\underline{x} - \underline{c}_i, \delta)}{\sum_{i=0}^{M} Z_i f_i(\underline{x} - \underline{c}_i, \delta)} \tag{2}$$

Where

$\underline{c}_i$ = center vector for cluster *i* in the input space

y_i = scalar output related to $\underline{c}_i$

Z_i = number of input vectors x_j within cluster $\underline{c}_i$

δ = single smoothing parameter chosen during network training

M = number of unique centers $\underline{c}_i$

Comparing Equation 2 and Equation 1 of GRNN, we can find the only difference is that VQ-GRNN applies its computation on a smaller number of clusters of input vectors represented by centers vectors $\underline{c}_i$ rather than working on individual input vectors $\underline{x}_n$. Though VQ-GRNN has minimal computational overheads and hence, more stable than GRNN, it may be less accurate in predicting attacks with low frequency of occurrence. The next section discusses a possible approach to enhance the predictive power of VQ-GRNN.

3.3. Overview on Ensemble Learning

The goal of learning algorithms is to discover the underlying functional relationship of input variables. Ordinary ML methods work by searching through a space of possible functions, called hypotheses, to find the best approximation to the unknown function. The best hypothesis can be identified based on how well it fits the training data and how consistent it is with any available prior knowledge about the problem. Ensemble learning algorithms take a different approach. Rather than finding one best learner to explain the data, they construct a set of learners, called a committee or ensemble, and then have those learners vote in some manner to predict the label of new data points. Even though the component learners within the ensemble are all attempting to solve the same problem, it is likely that each of them would have different strengths and weaknesses in different situations. Realizing and managing the situations in which the learner do not perform as well as expected is the key challenge for ensemble research (Costa, Filippi, & Pasero, 1995). A number of research (Windeatt & Roli, 2003) has supported a widespread view that for an ensemble to achieve best performance on a task, the component predictors should exhibit "diverse errors", meaning that they should have different error rates. However, in achieving this, the individual accuracy may be affected. Therefore, training an ensemble is actually a balancing act between error diversity and individual accuracy.

Due to the significant performance improvements over single classifiers, ensemble construction has become one of the most active fields of AI and has received immense research attention. In particular, ensemble algorithms iteratively run a base learning algorithm (called base learner) and then form a vote out of the resulting hypotheses (Schapire, 1999). There are two main approaches to producing these component hypotheses.

The first approach, namely bagging, is to construct each hypothesis independently in such a way that the resulting set of hypotheses is accurate and diverse, that is, each individual hypothesis has reasonably low error rate for making new predictions and yet the hypotheses disagree with each other in many of their predictions. It is empirically shown that an ensemble of those hypotheses is more accurate than any of its component classifiers, because their disagreements will "cancel out" when the ensemble comes to the joint classification stage (Optiz & Maclin, 1999).

Unlike bagging, which relies on resampling the training dataset randomly with a uniform probability distribution, boosting (Schapire, 1999) guides changes of the training data to direct further classifiers toward more "difficult cases". This method is a stepwise technique that combines learners in such a way that the composite – boosted learner – outperforms the single learner. Amongst popular boosting variants, Adaptive Boosting or AdaBoost is the most widely adopted method which allows the designer to continue adding weak learners until some desired low training error has been achieved (weak learners have accuracy only slightly better than chance whereas weak hypotheses are generated based on the performance of previous ones). AdaBoost is "adaptive" in the sense that it does not require prior knowledge of the accuracy of these hypotheses (Schapire, 1999). Instead, it measures the accuracy of a base hypothesis at each iteration and sets its parameters accordingly.

Without loss of generality, let us consider the standard two-class supervised ML problem: given a set of N independent and identically distributed (i.i.d) training examples (x_n, y_n), n = 1,…,N, with $x_n \in X$ and $y_n \in Y \coloneqq \{-1, +1\}$, we would like to learn a function $f: X \rightarrow Y$ that is able to generalize well on unseen data generated from the same distribution as the training data. To obtain such a function, the boosting algorithm iteratively trains a weak hypothesis on a weighted data sample. As boosting progresses, training examples that are hard to predict correctly, get incrementally higher weights than the other examples. The intended effect is to force the weak learner to concentrate on examples and labels that will be most beneficial to the overall goal of finding a highly accurate classification rule. This update process is repeated, until a certain stopping condition is met (e.g a given number of weak classifiers are trained or the learning error reaches a desirable level). The final joint classification is the linear weighted combination of the base hypotheses (Huang, Ertekin, Song, Zha, & Giles, 2007):

$f_\alpha(x) = sign[\sum_{t=1}^{M} \alpha_i \, h_i \, (x)]$

Motivated by the need of an accurate detection system for network security applications, we seek a learning algorithm which provides a good tradeoff for learning bias, generalization variance and computational requirement. In theory, the GRNN can achieve the optimal Bayesian estimate (with infinity network size) but with a cost of extremely demanding computation resource. The VQ-GRNN reduces the computationally extensive nonparametric GRNN to a semiparametric neural network by applying vector quantization techniques on the input space. This reduction significantly improves the robustness of the algorithm (low variance), but also affects its learning accuracy to some extent. To overcome this limitation, AdaBoost is used to boost its performance. The boosted version of VQ-GRNN which is referred to as Boosted VQ-GRNN will be implemented in the Multi-Expert Classification Framework.

4. Multi-Expert Classification Framework

Different learning algorithms behave variably on different classes. They may obtain superior performance on some classes but present unacceptable low accuracy for others. The imbalance of predictive performance motivates this research to construct an intelligent multi-expert learning framework which can aggregate expert knowledge from class-specific models, i.e. classifiers specialized in detecting a specific class. There is a good deal of research that shows the potentials of models that combines classification results from individual sub-models. Basically, there are two forms of classifier combination, the multi-stage (or hierarchical) (Vuurpijl, 2000) methods and the ensemble (or late fusion) (Kuncheva, 2002.) methods. In the first approach, the classifiers are placed in a multi-layered architecture where the output of one layer affects the model selection in the next layer. On the other hand, the second approach explores ensembles of classifiers, trained on different distributions of the original dataset and using different or similar features and learning algorithms. The outputs of these classifiers are then fused into one compound classification using voting techniques.

For a multi-class classification problem such as network intrusion detection, instead of trying to design a learning algorithm that is accurate over the entire space, we can focus on creating a model that can predict well for a specific portion within the space. We then combine such models to obtain a joint classifier which performs accurately on many classes. Under this light, a Multi-Expert Classification Framework (MECF) combining different classifiers for different types of attacks is proposed. Its sub-models are trained in an attack-specific manner and then integrated to accumulate their specializations. Boosted VQ-GRNN will be compared with several algorithms and then used for creating component classifiers.

4.1. Framework description

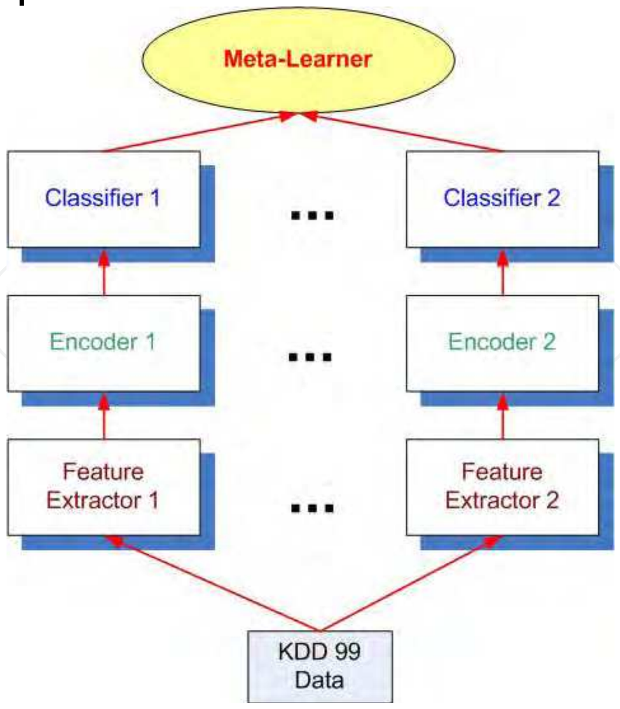

Fig.1. Multi-expert classifier for Intrusion Detection

Fig.1 describes a generic predictive model which combines different classifiers; each with special expertise in detecting a specific attack type. Each of these classifiers will be trained on different subsets of an underlying universal dataset. These subsets differ from each other in terms of attribute selections (Feature selectors) and attack-specific encoding schemes (Encoder). We aim to construct the class-specific classifiers, called experts, which have high Detection Rate on specific classes. To do so, several combinations of different attributes will be tested to gain the best performance for a particular attack category. We then train those classifiers on the dataset whose labels indicate whether a data instance belongs to a particular attack category or not. For example, if a classifier is created to recognize Probe attacks (Probe expert), then the data labels will be encoded as Probe for instances belonging to probe category or Non-probe otherwise. This has an effect of reducing a multi-class learning problem into a multiple binary classifications. The learning speed will be faster and the resulting classifier will be more "specialized" in detecting particular categories of attacks and less prone to overfitting problem.

Another useful aspect of this approach originates from the fact that even when different classifiers are trained on the same dataset and have comparable performance on the test set, they still have different "inductive biases" (Mitchell, 1997). This prevents these models from generalizing in identical ways. Under the proposed arrangement, component classifiers are very different from each other in terms of their biases. From experiments, it is shown that if a classifier is trained with a dataset which emphasizes a particular attack, it will have good detection rate for that particular attack but does not detect other attacks well. One of the widely used approaches is the cross-validation which perceives the different "inductive biases" as an indication to select "super" classifiers which perform best on all classes. As a result, some models will be discarded because of their low performance. This leads to a potential loss of useful information and effort. In contrast, an ensemble can effectively make use of such complementary information to reduce model variance and bias (Tumer & Ghosh, 1995). The meta-learner in this framework could be as simple as a lookup table to a more advanced voting techniques. A brief review of related voting methods is given in the next section.

4.2. Voting techniques for pattern recognition

In human society, voting is a common concept in which voters indicate their preference choices from multiple options (candidates) by means of a vote (Parhami, 1994). These votes are then integrated into one final decision (the winner). This process is called an election. In the context of classifier combination, the voters are the individual classifiers that can generate a single class or a ranked list of all classes as a vote; the possible classes are the candidates and an election is the classification of one sample. The winner is the candidate that is chosen as result of the classification procedure of the sample by the combination of classifiers. There are a number of families of voting techniques.

Firstly, the un-weighted voting methods consider each vote equally and the only differentiation between the candidates is the number of votes they have received. As a consequence, voters cannot express the degree of preference of one candidate over another (Parhami, 1994). Apart from this limitation, un-weighted voting such as majority voting is still commonly used, due to its simplicity and relatively good performance. Particularly, every voter has one vote that can be cast for any one candidate and the candidate that obtained the majority of the votes will win the election.

The second family of voting methods is confidence voting in which voters can express the degree of their preference for a candidate by assigning a confidence value to candidates. The higher the total confidence value a candidate received, the more it is preferred by the voter. In our experiments, confidence value is equivalent to probabilities of class membership that are generated by local experts. There are 3 common ways of computing the total confidence votes: (1) summing up all confidence values (Sum rule); (2) multiplying all confidence values (Product rule); (3) repeatedly applying a majority vote based on the highest ranked candidate of each voter's preference ranking and transferring votes between candidates (Single transferable vote-STV) (Doron, 1977). The basic principle of STV is that voters rank the candidates in order of preference. In order to be elected, a candidate must achieve a computed quota. The votes can be transferred in two cases:

- Excess votes over the quota are appropriately down-weighted and allocated to the next preference of voters (this is not applicable in our case because we terminate voting when a winner is selected).
- If no candidate reaches the quota, the candidate with the least number of votes is eliminated and their votes transferred to next preferences.

In the context of classifier combination, voting techniques like STV are necessary because it can better integrate the preference choices of the local experts. For example, if no expert has enough confidence to classify an input vector, instead of marking it as an "unknown" instance which implies overheads for further investigation, the least voted candidate class is eliminated and its votes will be transferred to other classes. By this means, we not only utilize the votes that are otherwise wasted but also reduce the need for further processing of the unknown instances. In our experiments on the KDD-99 dataset, we attempt to use different voting methods and examine their behaviors in a multi-expert framework.

5. Experimental analysis

5.1. Cost-based Analysis

The KDD-99 dataset takes the cost sensitivity into consideration in evaluating learning methods. An error on a particular class may not be equally serious as errors on other classes. To make comparison between intrusion detection methods sensitive to cost, a cost matrix (CostM) is given for different attack categories.

Predicted / Actual	Normal (0)	Probe (1)	DoS (2)	U2R (3)	R2L (4)
Normal (0)	0	1	2	2	2
Probe (1)	1	0	2	2	2
DoS (2)	2	1	0	2	2
U2R (3)	3	2	2	0	2
R2L (4)	4	2	2	2	0

Table 1. Cost matrix for the KDD-99 dataset (Levin, 2000a)

In this table, rows correspond to actual categories, while columns correspond to classified values. The Normal category is symbolized as class 0, Probe as 1 and so forth. According to this cost matrix, if a R2L attack is falsely classified as Normal connection, the incurred penalty cost is 4 while misclassification of a Probe attack as normal has a cost of 1. This suggests that R2L attacks are more serious than Probes.
During the testing phase, the outputs of a classifier will be generated in form of a Confusion Matrix (ConfM) which summarizes the classification results. The difference between CostM and ConfM is that an entry at row i and column j in the cost matrix, CostM(i,j), represents the cost associated with a connection which actually belongs to class i and is classified as class j while the same position in the confusion matrix, ConfM(i.j), displays the number of connections of type i and is classified (correctly or incorrectly) as class j. Given a test set, the average cost of a classifier is calculated as below (McHugh et al., 2000):

$$Cost = \frac{1}{N}\sum_{i=1}^{5}\sum_{j=1}^{5} ConfM(i,j) * CostM(i,j)$$

Where
N: total number of connections in the dataset
ConfM(i,j): the entry at row i, column j in the confusion matrix.
CostM(i,j): the entry at row i, column j in the cost matrix.

5.2. Experiment design

We are motivated to explore how different learning algorithms perform for different attack categories, i.e. to check weather a certain algorithm may achieve superior performance for a specific attack category. In the light of this possibility, we compare several detection models using different pattern recognition methods and select the best performing algorithms as well as the most discriminant features for each attack category. A multi-classifier system then evolves which improves the overall detection performance on the KDD-99 benchmark.

A generic ensemble model is developed as in Fig.1, containing 5 classifiers (experts) which specialize in detecting certain classes (Normal, Probe, DoS, U2R and R2L). There are 3 major phases in this model:

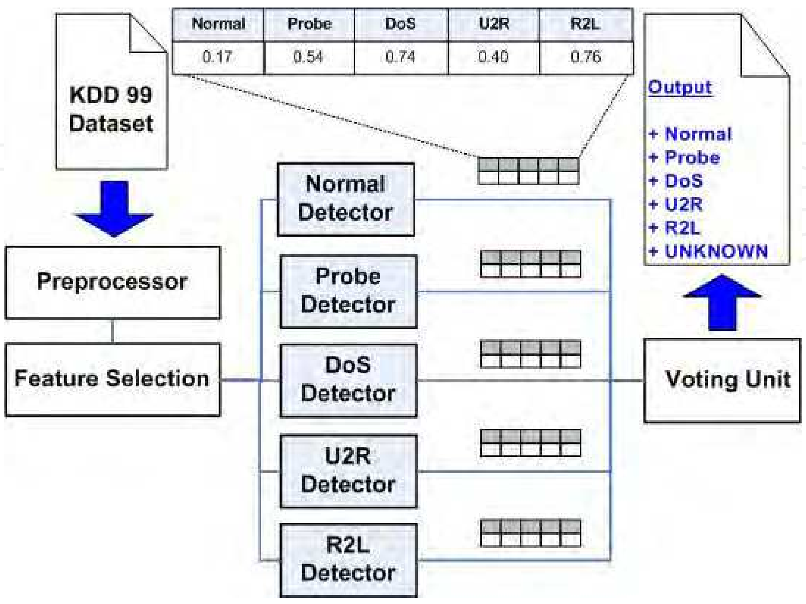

Fig. 2. Multi-expert classification framework (MECF)

5.2.1. Data preprocessing

The KDD-99 dataset contains attributes of different forms such as continuous, discrete and symbolic with varying resolutions and ranges. In order to build predictive models, this data is first converted into a compatible format. Several preprocessing techniques include data reduction (removing duplicated data and sampling data into smaller sets), data encoding and normalization (mapping k-value nominal and ordinal attributes into k integers, re-scaling numeric attributes), dimensionality reduction (extract most informative attributes by adapting Protocol-based Logistic Regression (PLR) (Yu, Wu, & Wong, 2008)).

5.2.2. Local expert creation

Local experts (detectors) are constructed by selecting the best performing learning algorithms and best disriminant features for specific attack types. For each class (normal and attack categories), a specialized classifier is created by three steps:

a) Apply class-specific encoding schemes on data

For example, to produce a DoS detector, each data record is encoded to indicate whether it is actually a DoS attack or not, i.e. its label is assigned to *1* if it belongs to this class, or *0* otherwise.

Given a dataset *S* in which the input features are represented by vector *x* and the output (or target) class is denoted by label *c* where *c*=1,…,*K* and *K* is the number of possible labeling (in KDD-99, *K*=5). To construct a local classifier which is specialized in detecting a specific class *k*, the label *c* should be recoded to y_k such that: $\begin{cases} y_k = 1 \ if \ c = k \\ y_k = 0 \ if \ c \neq k \end{cases}$

b) Select important features from the input data

We train some classifiers with different combinations of attributes in the encoded data. The combination which gives the best performance will be selected for that particular class.
c) Choose the best performing classifier
The learning algorithm with the best trade-off between high detection capability and low false alarm rate will be selected.

5.2.3. Expert Combination

Given an input vector, each local expert computes an array of probabilities (ranging from 0 to 1) of class membership for each available class. These probabilities are merged by voting methods to decide the final classification of the input vector. Several voting approaches will be implemented and their performance will be compared in the next section.

5.3. Experiment results

5.3.1. Constructing MECF

To construct local experts, different predictive models are trained with different combinations of data feature groups including basic group (B), content-based group (C) and traffic-based group (T) for basic, content-based and traffic based features respectively. In particular, the semi-parametric algorithms such as Boosted VQ-GRNN will be compared against the parametric models such as decision trees (linear discriminant), boosted trees and non-parametric methods (MLP, GRNN).The boosted tree algorithm is the combination of J48 and AdaBoostM1 methods which are available from the Weka package. For the GRNN and Boosted VQ-GRNN models, a similar model size was chosen with one hidden layer containing 15 hidden nodes. The MLP has a structure of three layers with the number of input neurons equal to the number of input features, five hidden neurons and five output neurons (1 for each class). The above numbers of hidden nodes are selected from multiple experiments (number of hidden nodes varying from 5 to 55 with steps of 10) by choosing the setting with lowest bias and variance.

Model	B	T	C	B+T	B+C	**T+C**
	Detection rate for Normal (%)					
J48 Tree	81.1	11.4	11.2	79.9	80.2	**41.2**
Boosted J48	94.6	12.5	13.1	92.0	92.2	**43.2**
MLP	85.6	8.6	8.5	85.5	86.7	**30.4**
SVM	90.4	10.9	10.4	91.2	91.5	**29.8**
GRNN	95.1	13.5	12.5	93.8	94.6	**40.2**
Boosted VQ-GRNN	95.6	14.5	13.5	95.1	95.2	**33.8**
	Detection rate for Probe (%)					
J48 Tree	46.2	20.7	22.4	86.1	56.4	**39.1**
Boosted J48	45.1	25.1	23.1	90.7	56.8	**33.2**
MLP	39.1	27.7	11.3	82.3	40.1	**32.1**
SVM	42.7	30.8	16.9	87.5	50.6	**68.3**
GRNN	44.6	25.1	21.7	84.3	47.8	**44.1**
Boosted VQ-GRNN	45.1	29.1	20.3	86.2	50.2	**43.5**

	Detection rate for DoS (%)					
J48 Tree	78.2	17.2	34.7	88.6	76.1	**51.3**
Boosted J48	79.0	24.1	50.1	90.3	77.3	**53.7**
MLP	66.7	15.7	40.1	87.6	60.1	**49.1**
SVM	78.2	20.2	33.5	85.6	58.3	**50.3**
GRNN	78.1	22.2	44.8	92.0	75.7	**51.4**
Boosted VQ-GRNN	78.2	23.1	43.2	92.3	78.2	**50.5**
	Detection rate for U2R (%)					
J48 Tree	0.4	0.0	5.0	0.3	22.3	**7.6**
Boosted J48	0.1	0.1	4.3	1.1	27.2	**7.8**
MLP	0.3	0.8	6.7	0.1	18.1	**7.1**
SVM	0.0	0.5	4.6	0.2	11.6	**6.6**
GRNN	0.1	3.2	7.2	0.1	28.6	**6.0**
Boosted VQ-GRNN	0.0	3.2	7.2	0.4	27.2	**6.1**
	Detection rate for R2L (%)					
J48 Tree	0.9	0.0	2.9	0.0	34.0	**2.8**
Boosted J48	0.5	0.0	2.8	0.0	35.2	**2.3**
MLP	0.0	0.1	2.1	0.1	22.1	**3.0**
SVM	0.0	0.0	1.5	0.0	33.8	**2.5**
GRNN	0.9	0.1	2.3	0.3	36.2	**3.3**
Boosted VQ-GRNN	**0.1**	**0.1**	**2.1**	**0.3**	**41.2**	**3.1**

Table 2. Detection rate for different classes and features

From the results in Table 2, for each type of attacks, the best performing combination of features and learning algorithms will be chosen and highlighted. For example, the combination of basic and traffic features with the Boosted J48 Tree achieve the highest detection rate for Probe attacks (90.7%). Note that, these models are trained on the data which is encoded for specific categories. Therefore, their detection rates are only valid on the encoded data. The Table 3 shows the best performing strategies (feature combination, attack-specific encoding scheme and learning algorithm) selected.

Model	**Features used**	**Encoding scheme used**	**Algorithm used**
Normal Expert	Basic	1 (Normal) ; 0(non- Normal)	*Boosted VQ-GRNN*
Probe Expert	Basic + Traffic	1 (Probe) ; 0(non-probe)	*Boosted J48 Tree*
DoS Expert	Basic + Traffic	1 (DoS) ; 0(non- DoS)	*Boosted VQ-GRNN*
U2R Expert	Basic + Content	1 (U2R) ; 0(non-U2R)	*GRNN*
R2L Expert	Basic + Content	1 (R2L) ; 0(non- R2L)	*Boosted VQ-GRNN*

Table 3. Local experts' configuration

5.3.2. Performance evaluation

The classification results of the constructed local experts are combined in the Multi-Expert Classification Framework (MECF) using different voting strategies including majority vote (MECF–MV), sum rule (MECF–SR), product rule (MECF–PR) and Single Transferable Vote

(MECF–STV) voting methods. These models are then compared against other existing methods, including the KDD-99 winner (Pfahringer, 2000a), the rule-based PNrule approach (Agarwal, 2000) and the Columbia Model (W. Lee & Stolfo, 2000). Results from some of these techniques may not be complete (e.g. FAR is not available or results of Normal class are not provided). The comparison between learning algorithms is presented Table 4 where for each class, the highest DR and lowest FAR are in bold and the best performing method is highlighted.

	Normal	**Probe**	**DoS**	**U2R**	**R2L**	DR/FAR (%)
KDD 99 winner (Pfahringer, 2000a)	99.5	83.3	97.1	13.2	8.4	DR
	27.0	35.2	0.1	28.6	1.2	FAR
PNrule(Agarwal, 2000)	99.5	73.2	96.9	6.6	10.7	DR
	27.0	7.5	**0.05**	89.5	12.0	FAR
Columbia Model (W. Lee & Stolfo, 2000)		96.7	24.3	81.8	5.9	DR
MECF–MV	99.4	88.0	97.2	29.3	11.9	DR
	30.2	40.1	2.8	14.5	20.0	FAR
MECF–SR	99.5	92.0	96.7	21.8	17.1	DR
	3.3	6.7	0.09	7.1	8.7	FAR
MECF–PR	82.1	85.3	98.0	11.4	6.8	DR
	5.2	21.4	0.56	4.3	15.7	FAR
MECF–STV	**99.8**	**99.3**	**98.1**	**89.7**	**48.2**	DR
	3.6	1.1	0.06	**0.03**	**0.19**	FAR

Table 4. Detection Rate (DR %) and False Alarm Rate (FAR %) comparison

Across the classes, in comparison with existing techniques, MECF using simple majority vote (MECF-MV) does not provide noticeable improvement in DR while its FAR is quite high in most cases (Probe, U2R, R2L and Normal). The product rule voting technique (in MECF-PR) is found unstable because it suddenly increases DR for the DoS attack (98.0%) while its results for the remaining classes are largely degraded. The MECF-SR, on the other hand, has a stable performance with fairly high DR and low FAR for most of the classes (it has lowest FAR for the Normal class).

Among the methods considered here, our classification framework that uses STV technique (MECF-STV) and the LCRF (Gupta, Nath, & Kotagiri, 2008) are the most recent and they seem to be the most accurate models (high DR and low FAR). Most methods do not perform well for the U2R attacks (DR is lower than 30%), except for a dramatic increase in DR is noted for Decision Tree (J.-H. Lee et al., 2008) (58.8%), Columbia Model (W. Lee & Stolfo, 2000) (81.8%), LCRF (Gupta et al., 2008) (86.30%) and MECF-STV (89.7%). For the R2L category, only MECF-STV provides a significantly high DR (48.2%) and lowest FAR (0.19%).

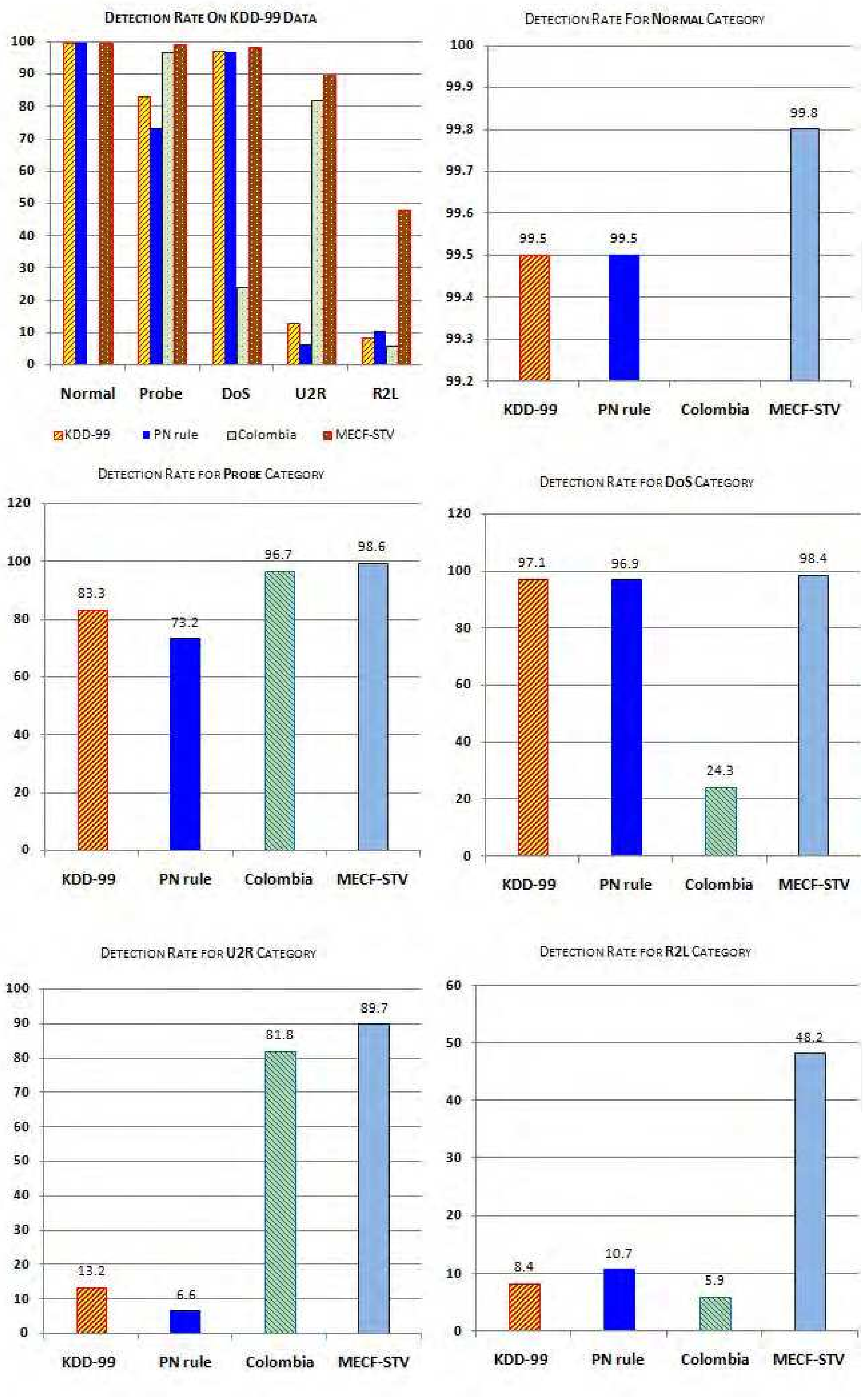

Fig. 3. Detection Rate comparision

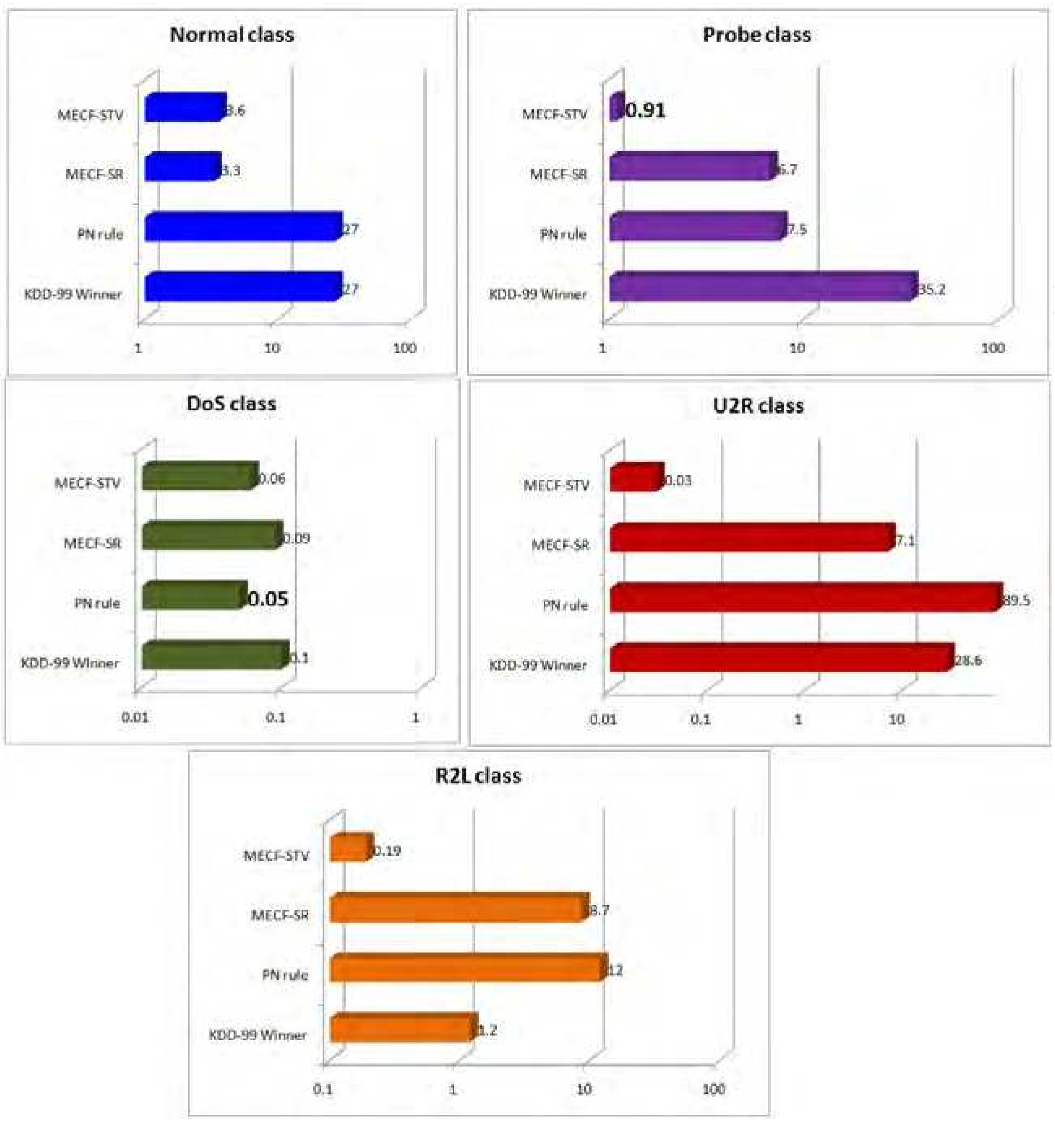

Fig. 4. False Alarm comparision

Though no model can provide both highest DR and lowest FAR for all the classes, our MECF-STV is the most promising model which makes the best combination of detection capability (DR) and system robustness (FAR). That is, MECF-STV can achieve highest DR for all the classes and its FAR for the two rare U2R and R2L categories are the lowest. In the case that other models obtain lower FAR, the performance difference between that model and MECF-STV is very small. Moreover, the average cost of MECF-STV is 0.1089 per test sample, which is much lower than the KDD winner (0.2332). It is also important to note that the test data used in our experiments follows a different distribution than in the training data and contains an additional 14 attack types not included in the training data. Therefore, achieving high DR on this test dataset suggests that our model is robust to data distribution changes and is able to detect unseen attacks. Fig. 3 and Fig. 4 visualize DR and FAR of the classifiers on the KDD-99 dataset.

In summary, our proposed MECF-STV (multi-expert classification framework using single transferable voting) can significantly reduce the total misclassification cost compared with KDD-99 winner. Its detection rates are the highest for Normal, U2R and R2L categories and very close to that of the best performing classifiers for Probe and DoS categories. Finally, False alarm rates obtained by MECF-STV are often the lowest compared to other methods.

6. Conclusions and Future work

6.1. Conclusion remarks

Motivated by the low detection rates on rare and complicated attacks in the KDD-99 benchmark, we develop the Multi-Expert Classification Framework (MECF) in which Vector Quantized Generalized Regression Neural Network (VQ-GRNN) and Adaptive Boosting (AdaBoost) are deployed. It is shown that some learning algorithms that use certain sets of features and class-specific encoding schemes can achieve superior detection capability for a given attack category. Consequently, MECF aims to capture the characteristics of different intrusive classes and normal instances by constructing a set of five local classifiers (experts) to classify data into five different classes including Normal, Probe, DoS, U2R and R2L. For each of these classes, a tailored learning strategy (an expert) is employed. The outcomes of these experts will then be combined using high performance voting methods. Experimental results indicate that the weighted voting strategies outperform simple majority voting. Using the transferable voting approach, our Multi-Expert Classification Framework using Single Transferable Voting (MECF-STV) model can significantly improve the detection rates of not only minority and distributed U2R and R2L attacks but also majority classes compared with other techniques. Moreover, this model achieves a low detection cost.
In conclusion, the empirical analysis from this research suggests that our proposed framework performs very well in the intrusion detection problem in terms of accuracy and system robustness while offering “affordable” computation compared with existing state-of-the-art techniques. However, no system is absolutely secure given the best possible detection algorithms. That is true as long as the system is connected to other networks. The absolute security can only be achieved by disconnecting the system from the outside world which is against the principal benefits of internetworking – accessibility of information. This means that protecting our resources from network attacks is an ongoing task and computer security is always an active and challenging research area.

6.2. Future works

Increasingly-intense distributed denial-of-service (DDoS) attacks on Internet Service Provider (ISP) backbones are surpassing providers' capacity and knocking customers offline (Kemmerer & Vigna, 2002). Such attacks are more dangerous than traditional DoS due to its complex and distributed nature. To fight these attacks, the generic IDS examined here can be extended to a multi-level agent detection system for distributed networks. Literature in this area has highlighted several generic limitations associated with Distributed Intrusion Detection System (DIDS) such as inability to cope with huge amount of data in different formats and ineffective coordination between distributed sensors and agents (Kemmerer & Vigna, 2002). Some of these problems were outlined and different approaches have been implemented to solve those problems. It is interesting to explore the applicability of our framework in such distributed context, i.e. using the MECF-STV to develop a multi-level agent framework to construct a robust, distributed, error tolerant and self protecting DIDS. To design such DIDS, an appropriate design of centralization and decentralization of data processing and detection capabilities needs to be considered.

7. References

Agarwal, R., Joshi, M. V. . (2000). PNrule: A New Framework for Learning Classifier Models in Data Mining. Paper presented at the Technical Report

Ambwani, T. (2003). Multi class support vector machine implementation to intrusion detection. Paper presented at the Proceedings of the International Joint Conference of Neural Networks.

Amor, N. B., Benferhat, S., & Elouedi, Z. (2004). Naive Bayes vs. Decision Trees in Intrusion Detection Systems. Proc. ACM Symp. Applied Computing, 420-424.

Bauer, D. S., & Koblentz, M. E. (1988). NIDX – an expert system for real-time network intrusion detection. Paper presented at the Proceeding of the Computer Networking Symposium.

Cannady, J. (1998). Artificial neural networks for misuse detection. Paper presented at the In Proceedings of the National Information Systems Security Conference.

Costa, M., Filippi, E., & Pasero, E. (Eds.). (1995). Artificial neural network ensembles: a bayesian standpoint. : World Scientic.

Doron, G., Kronick, R. (1977). Single transferable vote: an example of a perverse social choice function. American Journal of Political Science, 21(2), 303–311.

Gupta, K. K., Nath, B., & Kotagiri, R. (2008). Layered Approach using Conditional Random Fields for Intrusion Detection. IEEE Transactions on Dependable and Secure Computing, 5(4).

Huang, J., Ertekin, S., Song, Y., Zha, H., & Giles, C. L. (2007). Efficient Multiclass Boosting Classification with Active Learning. SIAM International Conference on Data Mining.

Ilgun, K., Kemmerer, R. & Porras, P. (1995). State transition analysis: a rule-based intrusion detection approach. IEEE Transactions on Software Engineering, 181-199.

Kemmerer, R. A., & Vigna, G. (2002). Intrusion detection: A brief history and overview. IEEE Security and Privacy.

Kononenko, I., & Kukar, M. (2007). Machine Learning and Data Mining: Introduction to Principles and Algorithms Horwood Publishing Limited.

Kruegel, C., Mutz, D., Robertson, W., & Valeur, F. (2003). Bayesian Event Classification for Intrusion Detection. Proc. 19th Annual Computer Security Applications Conference, 14-23.

Kuncheva, L. I. (2002.). A theoretical study on six classifier fusion strategies. Paper presented at the IEEE Transactions on Pattern Analysis and Machine Intelligence.

Lee, J.-H., Lee, J.-H., Sohn, S.-G., Ryu, J.-H., & Chung, T.-M. (2008). Effective Value of Decision Tree with KDD 99 Intrusion Detection Datasets for Intrusion Detection System. Paper presented at the 10th International Conference on Advanced Communication Technology.

Lee, W., & Stolfo, S. (2000). A Framework for Constructing Features and Models for Intrusion Detection Systems. Information and System Security, 4, 227-261.

Lee, W., Stolfo, S., & Mok, K. (1999a). A Data Mining Framework for Building Intrusion Detection Model. Proc. IEEE Symp. Security and Privacy, 120-132.

Lee, W., Stolfo, S., & Mok, K. (1999b). Mining Audit Data to Build Intrusion Detection Models. Proc. Fourth International Conference Knowledge Discovery and Data Mining 66-72.

Levin, I. (2000a). KDD-99 Classifier Learning Contest: LLSoft's Results Overview. SIGKDD Explorations, 1, 67–75.

Levin, I. (2000b). KDD-99 Classifier Learning Contest: LLSoft's Results Overview Paper presented at the SIGKDD Explorations.

McHugh, J., Christie, A., & Allen, J. (2000). Defending Yourself: The Role of Intrusion Detection Systems. Software, IEEE, 17(5), 42-51.

Miheev, V., Vopilov, A., & Shabalin, I. (2000). The MP13 Approach to the KDD'99 Classifier Learning Contest. SIGKDD Explorations, 1, 76–77.

Miheev, V., Vopilov, A. Shabalin, I. (2000). The MP13 Approach to the KDD'99 Classifier Learning Contest. Paper presented at the SIGKDD Explorations.

Mitchell, T. (1997). Machine Learning. New York: McGraw-Hill.

Mukkamala, S., Janoski, G., &Sung , A. (2002). Intrusion detection using neural networks and support vector machines. Paper presented at the International Joint Conference on Neural Networks (IJCNN).

Optiz, D., & Maclin, R. (1999). Popular ensemble methods: An empirical study. Journal of Artificial Research, 11, 169–198.

Parhami, B. (1994). Voting Algorithms. Paper presented at the IEEE Transactions of Reliability

Pfahringer, B. (2000a). Winning the KDD99 Classification Cup: Bagged Boosting. SIGKDD Explorations, 1, 65–66.

Pfahringer, B. (2000b). Winning the KDD99 Classification Cup: Bagged Boosting. Paper presented at the SIGKDD Explorations.

Schapire, R. E. (1999). A brief introduction to boosting. Paper presented at the Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, San Francisco, CA.

Shah, H., Undercoffer, J., & Joshi, A. (2003). Fuzzy Clustering for Intrusion Detection. Proc. 12th IEEE International Conference Fuzzy Systems (FUZZ-IEEE '03), 2, 1274-1278.

Sommer, R. (2008). Viable Network Intrusion Detection: Trade-Offs in High-Performance Environments: VDM Verlag.

Spetch, D. F. (1991). A general regression neural network. IEEE Transactions on Neural Networks, 2(6), 568-576.

Tumer, K., & Ghosh, J. (1995). Order statistics combiners for neural classifiers. World Congress on Neural Networks, 1, 31-34.

Vuurpijl, L., Schomaker, L. (2000). Two-stage character classification: a combined approach of clustering and support vector classifiers. Paper presented at the In International Workshop on Frontiers in Handwriting Recognition (IWFHR).

Windeatt, T., & Roli, F. (Eds.). (2003). Multiple Classier Systems, 4th International Workshop, MCS 2003. Guilford, UK: Springer.

Yu, K.-M., Wu, M.-F., & Wong, W.-T. (2008). Protocol-based classification for intrusion detection. Paper presented at the Proceedings of the 7th WSEAS International Conference on Applied Computer and Applied Computational Science.

Zaknich, A. (1998). Introduction to the modified probabilistic neural network for general signal processing applications. IEEE Transactions on Signal Processing, 46, 1980-1990.

Zaknich, A. (2003). Neural Networks for Intelligent Signal Processing. Sydney: World Scientific Publishing.

Zhang, Z., Li, J., Manikopoulos, C. N., Jorgenson, J., & Ucles, J. (2001). HIDE: A Hierarchical Network Intrusion Detection System Using Statistical Preprocessing and Neural Network Classification. Proc. IEEE Workshop Information Assurance and Security, 85-90.