Feature Extraction Methods for Intrusion Detection Systems

Hai Thanh Nguyen
Katrin Franke
Slobodan Petrović

NISlab, Department of Computer Science and Media Technology,
Gjøvik University College, P.O. Box 191, N-2802 Gjøvik, Norway,
Email:{hai.nguyen, katrin.franke, slobodan.petrovic}@hig.no

ABSTRACT

Intrusion Detection Systems (IDSs) have become an important security tool for managing risk and an indispensable part of overall security architecture. An IDS is considered as a pattern recognition system, in which feature extraction is an important pre-processing step. The feature extraction process consists of feature construction and feature selection . The quality of the feature construction and feature selection algorithms is one of the most important factors that affects the effectiveness of an IDS. Achieving reduction of the number of relevant traffic features without negative effect on classification accuracy is a goal that largely improves the overall effectiveness of the IDS. Most of the feature construction as well as feature selection works in intrusion detection practice is still carried through manually by utilizing domain knowledge. For automatic feature construction and feature selection, the filter, wrapper and embedded methods from machine learning are frequently applied. This chapter provides an overview of various existing feature construction and feature selection methods for intrusion detection systems. A comparison between those feature selection methods is performed in the experimental part.

INTRODUCTION

Intrusion Detection Systems (IDSs) have become an important security tool for managing risk and an indispensable part of overall security architecture (Northcutt, 1999). An Intrusion detection system gathers and analyzes information from various sources within computers and networks to identify suspicious activities that attempt to illegally access, manipulate, and disable computer systems. The two main intrusion detection approaches are misuse detection and anomaly detection (Denning, 1986). Misuse detection systems, for instance, Snort (Roesch, 1999), detect intrusions by looking at specific signatures of known attacks. This approach is similar to the way of detecting viruses in many antivirus applications. A set of patterns of known attacks is necessary be built in advance for further detections. It is easy to implement misuse detection systems. However, these systems are not effective against novel attacks that have no matched patterns yet. Anomaly detection systems, such as IDES (Lunt et al., 1992), can overcome the shortcoming of the misuse detection systems. An anomaly detector assumes that normal behaviors are different from abnormal behaviors. Therefore, abnormal activities can be detected by looking at normal activities only. In fact, in these system, a profile of normal behavior is set up and is utilized to flag any observed activities that deviate significantly from the established profile as anomalies or possible intrusions. Although anomaly detection systems have potentials of detecting novel attacks, it is not easy to define normal behaviors and these systems tend to generate more false positive alerts than the misuse detection systems.

An approach for building anomaly intrusion detection systems is to utilize machine learning and statistical techniques. By means of this approach, an intrusion detection system is considered as a statistical pattern recognition system. Figure 1 shows the model of a statistical pattern recognition system that consists of two phases: training and classification. The test and training patterns as raw data are normalized, noise as well as unwanted data is removed by the preprocessing modules. In the training phase, the feature extraction/selection module looks for a representative feature set from the input patterns. Those features are then utilized for training a classifier. In the classification phase, the trained classifier is applied to assign the test patters to one of the pattern classes under consideration of the selected features from the training phase. In the following, we will focus on this approach for intrusion detection.

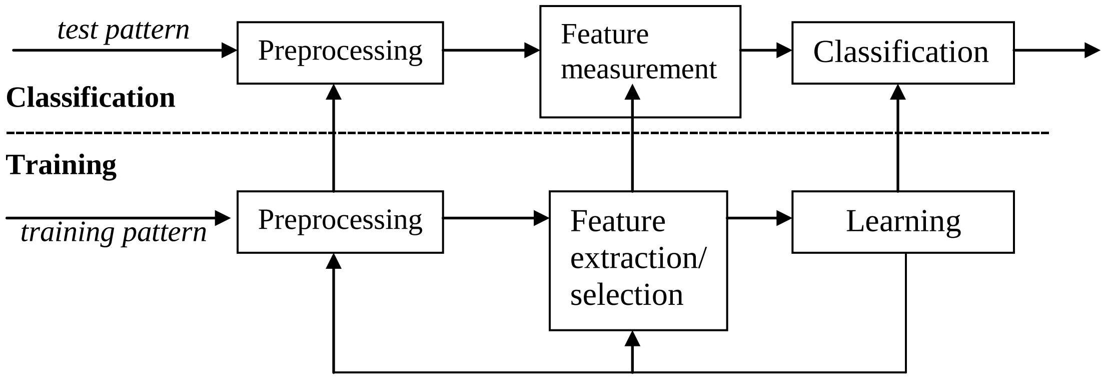

Figure 1. Model of a statistical pattern recognition (Jain et al., 2000)

From Figure 1, it can be observed that feature extraction is an important part of a pattern recognition system. The feature extraction process consists of feature construction and feature selection. The quality of the feature construction and feature selection algorithms is one of the most important factors that influence the effectiveness of an IDS. Achieving reduction of the number of relevant traffic features without negative impact on classification accuracy is a goal that largely improves the overall effectiveness of the IDS. Most of the feature construction as well as feature selection works in intrusion detection practice is still carried out through manually utilizing domain knowledge. For automatic feature construction and feature selection, the filter, wrapper and embedded methods from machine learning are frequently applied. This chapter provides an overview of various existing feature construction and feature selection methods for intrusion detection systems. A comparison between those feature selection methods is provided in the experimental part.

As this chapter aims to serve a wide audience from researchers to practitioners, we first introduce the basic concepts and describe the main feature extraction methods for intrusion detection; then present the practical applications of these methods to extract features from public benchmarking data sets for intrusion detection systems.

THEORETICAL BACKGROUND

Feature, which is a synonym for input variable or attribute, is any representative information that is extracted from the raw data set. A special pattern, which is directly selected from the data set, can be considered as a feature, for example, a representative substring of SQL injection code in a HTTP request. The distributions of characters or groups of characters are features. In some cases, the structures or semantics of data sets are considered as features. The relations between patterns or between features of the data are normally hidden, but are important for representing the data. Therefore, they are necessary be extracted from the data. The process of determining the most compact and informative features of a given data set is called Feature Extraction. By means of this process not only the efficiency of data storage is improved, but also the processing performance of intrusion detection systems is increased.

A feature extraction algorithm composes of two steps: feature construction and feature selection. Feature construction is one of the key steps in the data representation process for many tasks, such as classification or regression problems, largely conditioning the success of any subsequent statistic or modeling of a given raw data. This process refers to determining representative features from the original data. One can manually construct the features by looking at direct patterns in the data, for example, as we carry through when building signatures or rules for misuse intrusion detection systems. For automatic feature construction, several approaches, such as n-grams, association rule learning and frequency episode extraction, are usually applied. These methods will be introduced in more detail in the next sections.

At the feature construction stage, one should beware that any information could not be lost from the original data set. A common idea is to take into account all possible informative features. However, adding more features seems to come at a price: it increases the dimensionality of the data that is considered, thus increases the complexity of intrusion detection systems. Moreover, the irrelevant and redundant features are possibly contained in the set of features. How do we know when a feature is relevant or important? That is what feature selection is about and it is the main focus of this chapter. In general, feature selection can provide the following benefits (Guyon et al., 2006):

- General data reduction, i.e., to limit storage requirements and increase algorithm speed;
- Feature set reduction, i.e., to save resources in the next round of data collection or during utilization;
- Performance improvement, i.e., to gain predictive accuracy;
- Data understanding, i.e., to gain knowledge about the process that generated the data or simply visualize the data.

There are two ways of selecting features for intrusion detection systems: manual and automatic. Later on the automatic feature selection methods, which include filter, wrapper and embedded models from machine learning, will be emphasized.

Feature Construction

In this section, we present feature construction methods for intrusion detection systems that include association rule learning, frequency episode extraction and n-grams extraction.

Association Rule Learning

Association rule learning (Agrawal et al., 1993) is one of the most popular methods in data mining for discovering interesting relations between variables or features in large data sets. Such interesting relations are normally hidden in the raw data, and when they are extracted, they can be utilized for describing the data efficiently. For example, whenever one buys bread, she or he is likely to also buy butter and milk. Such information can be utilized as the basic for describing customers' behavior for making marketing activities. Another example in intrusion detection is that certain programs only get access to certain system files in specific directories, certain users (root, normal users or super users) have certain behavior or activities, such as normal users use mostly the utilities of the systems, whereas super users manage the systems, for instance, creating new users, new profiles or logging activities of other users, processes and so on. In this example, the interesting relations, which are associations between users and the used programs, are necessary be extracted and should be included in normal and suspicious usage profiles for further intrusion detection. For instance, if we have a profile of all users in a system, in which only super users have the right to modify a directory, then a normal user attempts to carry through the modification, his activity should be detected as suspicious, since in the profile of the system, there is no description of this activity for normal users.

The formal definition of association rule learning is given as follows: Let $I = \{i_1, i_2, ..., i_n\}$ be a set of *n* features called items of a system audit data. Let $D = \{r_1, r_2, ..., r_m\}$ be a set of records in this data set. Each record r_i contains a subset of features in *I*. A rule is defined as an implication of the form:

$$X \Rightarrow Y \text{, where } X, Y \subseteq I \text{, and } X \cap Y = 0.$$

When interpreting the above example in intrusion detection according to this definition, the followings are item sets: $X = i_1 = programmer(users)$, and $Y = i_2 = C\_compiler(used\_programs)$. The implication $X \Rightarrow Y$ is an association rule.

Before describing the main idea of association rules learning algorithms, two important concepts are introduced: the support *SUPP(X)* of an item set *X*, and the confidence $CONF(X \Rightarrow Y)$ of a rule $X \Rightarrow Y$ as follows:

- The support of an item set *X* (*SUPP(X))* is defined as the proportion of records in the data set that contain the item set *X*.
- The confidence $CONF(X \Rightarrow Y)$ of a rule ($X \Rightarrow Y$) is defined as follows:

$$CONF(X \Rightarrow Y) = \frac{SUPP(X \cup Y)}{SUPP(X)}$$

An association rule learning algorithm consists of two separate steps: Firstly, choosing a minimum threshold of the support values and looking for all possible frequent item sets in the data that have support values exceeding the chosen threshold. Secondly, these obtained frequent item sets are utilized to construct rules, which have confidence values exceeding the minimum threshold. There are several efficient algorithms for association rule learning, such as Apriori (Agrawal and Srikant, 1994) and Eclat (Zaki, 2000) algorithms.

Frequency Episode Extraction

Frequency episodes (Mannila et al., 1995 and Mannila & Toivonen, 1996) are normally utilized for representing a sequential audit data. In fact, frequent episodes are collections of events occurring frequently together. For example, in the sequence of Figure 2, the episode "E is followed by F" occurs several times:

```
Events:      E       D       F               B       C       E       F       C       D       A
       -------|---------|---------|-------------------|---------|---------|---------|----------|---------|---------|-----
Time:        30                                                                                   65
```

Figure 2. A sequence of events.

Episodes, in general, are partially ordered sets of events. In intrusion detection, when discovering episodes in a system audit data, the goal is to look for relations between sequential patterns. Such relations will then be analyzed to understand the temporal as well as statistical nature of many attacks and normal users' behavior. From that, additional features will be extracted for detecting incoming traffic.

The formal definition of frequent episodes is given as follows: Let $I = \{i_1, i_2, ..., i_n\}$ be a set of *n* features called items of a system audit data. Let $D = P\{r_1, r_2, ..., r_m\}$ be a set of records in this data set. Each record r_i contains a subset of features in *I*. A frequent episode is defined as an expression of the form:

$X, Y \Rightarrow (Z, w)$, where $X, Y, Z \subseteq I$ and *w* is the width of considered time interval.

N-grams Extraction

Many attacks that exploit vulnerabilities of protocols and services can be detected by analyzing header information from network packets or by monitoring the network traffic connection attempts and session behavior. For detecting attacks that tend to send bad payloads to vulnerable services or applications, such as viruses or malicious codes, consideration of the header information is not sufficient. The payload information of packets is necessary be analyzed. Some patterns of attacks can be selected from the payload by using domain knowledge in order to build a set of signatures. For automatic feature extraction, *n-grams* extraction method is usually applied (Wang et al., 2006 and Wang & Stolfo, 2004). An *n-gram* is a subsequence of *n* items from a given sequence. In the case of intrusion detection, if a payload is considered as a string, then an *n-gram* is a substring of *n* characters. With an assumption that payloads of normal traffic are different from payloads of attack traffic, the following is an automatic feature construction method based on *n-grams* extraction for intrusion detection: We consider *n-grams* ($n \geq 1$), thus the space *S* of all possible *n-grams* has the size of 2^{8n}, as considering 8 bits representation for each character:

$$S = \{n - grams_i \mid i = 1...2^{8n}\}$$

Given a payload *p*, a feature vector of *p* can be constructed as follows:

$x_p = (x_1, x_2, ..., x_{2^{8n}})$, where x_i is the number of appearances of $n - gram_i$ in *p*.

Feature Selection

In this section, we present automatic feature selection methods for intrusion detection systems that include the filter model, the wrapper model and the embedded model from machine learning (Guyon et al., 2006 and Liu & Motoda, 2008).

The wrapper model assesses selected features by a learning algorithm's performance. In other words, in a wrapper model, one employs a learning algorithm and utilizes its performance to determine the quality of selected features. Therefore, the wrapper method requires a lot of time and computational resources to obtain the best feature subsets. However, these wrapper approaches are aimed at improving results of the specific classifiers they work with. In the following, one of the most popular machine learning algorithms, which are usually applied in wrapper models-Support Vector Machine (Vapnik, 1995), will be introduced.

The filter model considers statistical characteristics of a data set directly without involving any learning algorithm. Due to the computational efficiency, the filter method is usually utilized to select features from high-dimensional data sets, such as intrusion detection systems. The filter model encompasses two groups of methods: the feature ranking methods and the feature subset evaluating methods. The feature ranking methods assign weights to features individually based on their relevance to the target concept. The feature subset evaluating methods estimate feature subsets not only by their relevance, but also by the relations between features that make certain features redundant. It is well known that the redundant features can reduce the performance of a pattern recognition system (Guyon et al., 2006). Therefore, the feature subset evaluating methods are more suitable for selecting features for intrusion detection. A major challenge in the IDS feature selection process is to choose appropriate measures that can precisely determine the relevance and the relation between features of a given data set. Since the relevance and the relation are usually characterized in terms of correlation or mutual information (Guyon et al, 2006), in the following, two measures are considered: the correlation feature selection (CFS) measure (Hall, 1999) and the minimal-redundancy- maximal-relevance (mRMR) measure (Peng et al., 2005). It will be shown that these two measures can be fused and generalized into a generic feature selection (GeFS) measure for intrusion detection and also it will be presented how to obtain the best feature subsets by means of the GeFS measure.

In contrast to the filter and wrapper models, the embedded model of feature selection does not separate the learning from feature selection part. The embedded model integrates the selection of features in the model building. An example of such model is the decision tree induction algorithm (Duda, 2001), in which at each branching node, a feature has to be selected. Another example of the embedded model is SVM-based feature selection methods (Weston et al., 2001 and Guyon et al., 2002), in which the task of feature selection can be understood as looking for the feature subsets that lead to the largest possible generalization or equivalently to minimal risk. This SVM example will be shown in more detail in the next sections.

Wrapper Models for Feature Selection

Support Vector Machines

There are many machine learning algorithms that can be applied in the wrapper model for feature selection, such as Support Vector Machines, Neural Networks, Bayesian Networks and so on (Duda et al., 2001). In the following, one of the most popular algorithms, Support Vector Machine (SVM) (Vapnik, 1995), which is applied successfully in many application, is considered. SVM is a supervised learning technique, which can be utilized to both classification and regression problems. The main idea of the SVM is to construct a hyper-plane that tends to separate data points in the space. Below is the formal formulation of the Support Vector Machine algorithm.

A training data set *D* is given with *m* instances: $D = \{(a_i, c_i) \mid a_i \in R^n, c_i\{-1,1\}\}_{i=1}^m$, where a_i is the instance that has *n* features and class label c_i; a_i can be represent as a data vector as follows: $a_i = (a_{i1}, a_{i2}, ..., a_{in})$, where a_{ij} is the value of the j^{th} feature in instance a_i.

For the two-class classification problem, SVM learns the separating hyper-plane that maximizes the margin distance.The primal form of SVM is given below (Vapnik, 1995), where *w* is the weight vector and *b* is the bias:

$$\min_{w,b} \frac{1}{2}\|w\|_2^2 \qquad (1)$$

subject to the following constraints:

$$\{c_i(wa_i - b) \geq 1, i = 1...m\}$$

In 1995, Cortes and Vapnik (Cortes and Vapnik, 1995) proposed a modified version of SVM that allows for mislabeled instances. They called this version of SVM Soft Margin, which has the following form:

$$\min_{w,b,\xi} \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{m}\xi_i \qquad (2)$$

subject to the following constraints:

$$\begin{cases} c_i(wa_i - b) \geq 1 - \xi_i, \\ \xi_i \geq 0, i = 1...m, \\ \xi = (\xi_1, \xi_2, ..., \xi_m), \end{cases}$$

where ξ_i is a slack variable, which measures the degree of misclassification of instance a_i, $C > 0$ is the error penalty parameter.

The feature selection process is based on the evaluation of performances of the SVM algorithm. For a given feature subset, if the accuracy of the SVM is good, then the feature subset will be selected. In order to obtain the best feature subset, all 2^n possible feature subsets are necessary be tested. For more computational efficiency, heuristic search methods, such as backward search and forward search (Guyon et al., 2006), are usually applied.

Embedded Models for Feature Selection

SVM-based Feature Selection

The SVM-based feature selection methods were studied in depth in previous works (Guyon et al., 2002 and Weston et al., 2001). As an example of the embedded models this section focuses on the feature selection method for linear SVMs (Vapnik, 1995) for two-class classification problems, such as for intrusion detection task. In particular, the utilization of L1-norm SVM for feature selection, which was first proposed by Bradley and Mangasarian in 1998 (Bradley and Mangasarian, 1998), is considered. Feature selection is an indirect consequence of the training process of SVMs. In fact, in the context of linear SVMs for binary classification tasks, the number of selected important features is the number of non-zero elements of the weight vectors after the training phase. Following the notations from the definition of the SVM method described above, the formal formulation of L1-norm SVM feature selection (Bradley and Mangasarian, 1998) is given below:

$$\min_{w,b,\xi} \|w\|_1 + C\sum_{i=1}^{m}\xi_i \qquad (3)$$

subject to the following constraints:

$$\begin{cases} c_i(wa_i - b) \geq 1 - \xi_i, \\ \xi_i \geq 0, i = 1...m, \\ \xi = (\xi_1, \xi_2, ..., \xi_m) \end{cases}$$

Define $w = p - q$ with $p, q \geq 0$. The problem (3) is then equivalent to the following linear programming problem (Bradley and Mangasarian, 1998):

$$\min_{p,q,b,\xi} e_n^T (p+q) + C\sum_{i=1}^{m} \xi_i$$

subject to the following constraints:

$$\begin{cases} c_i(pa_i - qa_i - b) \geq 1 - \xi_i, \\ \xi_i \geq 0, i = 1...m, \\ \xi = (\xi_1, \xi_2, ..., \xi_m), \\ p, q \geq 0, e_n^T = (1,1,...,1) \in R^n \end{cases}$$

Bradley and Mangasarian (Bradley & Mangasarian, 1998 and Mangasarian, 2007) have shown in many applications that the utilization of L1-norm SVM leads to a feature selection method, whereas the utilization of the standard SVM (Vapnik, 1995) does not. However, it is realized that the Bradley and Mangasarian's method considers only one case of all *n* full-set features in the training phase. Since there probably exists irrelevant and redundant features (Guyon et al., 2006 and Liu & Motoda, 2008), it is necessary to test all 2^n possible combinations of features for training the SVM. In the following, a new general formulation of L1-norm SVM (GL1-SVM), which takes into account all 2^n possible feature subsets, is presented. Therefore, the traditional L1-norm SVM proposed by Bradley and Mangasarian is just only one case of the GL1-SVM introduced in this study. This is the reason why the method introduced in this study is called a general L1-norm SVM. The main idea of the GL1-SVM is that the weight vector and the data matrix are encoded by utilizing binary variables $x_j (j = 1...n)$ and additional non-negative variables $z = (z_1, ..., z_n, ..., z_{2n}) \geq 0$ as follows:

$$\begin{cases} p = (x_1 z_1, x_2 z_2, ..., x_n z_n), \\ q = (x_1 z_{n+1}, x_2 z_{n+2}, ..., x_n z_{2n}), \\ a_i = (a_{i1} x_1, a_{i2} x_2, ..., a_{in} x_n) \end{cases}$$

Following this encoding scheme, the GL1-SVM can be represented as a polynomial mixed 0-1 programming problem (PM01P), which can be solved to obtain the globally optimal solutions by applying the proposed methods described above in the section of filter models for feature selection.

$$\min_{x \in \{0,1\}^n} [\min_{z,b,\xi} (\sum_{j=1}^{n} x_j z_j + \sum_{j=1}^{n} x_j z_{n+j} + C\sum_{i=1}^{m} \xi_i)] \qquad (4)$$

subject to the following constraints:

$$\begin{cases}\sum_{j=1}^{n} c_i a_{ij} x_j^2 z_j - \sum_{j=1}^{n} c_i a_{ij} x_j^2 z_{n+j} - c_i b \geq 1 - \xi_i, \\ x = (x_1, x_2, ..., x_n) \in \{0,1\}^n \\ \xi = (\xi_1, \xi_2, ..., \xi_m), \xi_i \geq 0, i = 1...m, \\ z = (z_1, ..., z_n, ..., z_{2n}), z_k \geq 0, k = 1...2n.\end{cases}$$

Proposition 7: Suppose that *S1, S2* are minimal values of the objective functions from (3), (4), respectively. The following inequality is true:

$$S2 \leq S1$$

Proof.
It is obvious, since the problem (3) is a case of the problem (4) when $x = (x_1, x_2, ..., x_n) = (1,1,...,1)$. □

Remark: As consequence of the Proposition 7, solving the problem (4) provides a smaller error penalty and enlarges the margin between two support vector hyper-planes, thus possibly provides better generalization capability of SVM than solving the traditional L1-norm SVM proposed by Bradley and Mangasarian (Bradley and Mangasarian, 1998).

Filter Models for Feature Selection

Correlation Feature Selection Measure

The Correlation Feature Selection (CFS) measure evaluates subsets of features on the basis of the following hypothesis: *"Good feature subsets contain features highly correlated with the classification, yet uncorrelated to each other"* (Hall, 1999). This hypothesis gives rise to two concepts. One is the feature-classification (r_{cf_i}) correlation and another is the feature-feature ($r_{f_i f_j}$) correlation. There exist broadly two measures of the correlation between two random variables: the classical linear correlation (Rodgers and Nicewander, 1988) and the correlation which is based on information theory (Shannon, 1948). The feature-classification correlation r_{cf_i} indicates how much a feature f_i is correlated to a target variable *C*, while the feature-feature correlation $r_{f_i f_j}$ is, as the very name says, the correlation between two features f_i and f_j. The following equation from (Ghiselli, 1964) applied in (Hall, 1999) provides the merit of a feature subset *S* consisting of *k* features:

$$Merit_S(k) = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}} \qquad (5)$$

Here, $\overline{r_{cf}}$ is the average feature-classification correlation, and $\overline{r_{ff}}$ is the average feature-feature correlation, as given below:

$$\overline{r_{cf}} = \frac{\sum_{i=1}^{k} r_{cf_1}}{k}, \qquad \overline{r_{ff}} = 2\frac{\sum_{i,j=1, i\neq j}^{k} r_{f_i f_j}}{k(k-1)}$$

Therefore, equation (1) can be rewritten as follows:

$$Merit_S(k) = \frac{\sum_{i=1}^{k} r_{cf_i}}{\sqrt{k + 2(\sum_{i,j=1,i\neq j}^{k} r_{f_i f_j})}} \qquad (6)$$

In fact, the equation (5) is Pearson's correlation coefficient, where all variables have been standardized. It shows that the correlation between the feature subset *S* and the target variable *C* is a function of the number *k* of features in the subset *S* and the magnitude of the inter-correlations among them, together with the magnitude of the correlations between the features and the target variable *C*. From the equation (5), the following conclusions can be drawn: The higher the correlations between the features and the target variable *C*, the higher the correlation between the feature subset S and the target variable *C*; The lower the correlations between the features of the subset *S*, the more significant the correlation between the feature subset *S* and the target variable *C*.

The task of feature subset selection by means of the CFS measure: Suppose that there are *n* full set features. The subset *S* of *k* features, which has the maximum value of $Merit_S(k)$ over all 2^n possible feature subsets, is necessary be found:

$$\max_S \{Merit_S(k), 1 \le k \le n\}. \qquad (7)$$

Minimal-Redundancy-Maximal-Relevance Measure

In 2005, Peng et al. proposed a feature selection method, which is based on mutual information from information theory (Shannon, 1948). In this method, relevant and redundant features are considered simultaneously. In terms of mutual information, the relevance of a feature set *S* for the class *C* is defined by the mean value of all mutual information values between the individual feature f_i and the class *C* as follows:

$$D(S,C) = \frac{1}{|S|} \sum_{f_i \in S} I(f_i; C)$$

The redundancy of all features in the set *S* is the mean value R(*S*) of all mutual information values between the feature f_i and the feature f_j and is given below:

$$R(S) = \frac{1}{|S|^2} \sum_{f_i, f_j \in S} I(f_i; f_j)$$

The mRMR criterion is a combination of two measures given above and is defined as follows:

$$\max_S \phi_S(D,R) = \frac{1}{|S|} \sum_{f_i \in S} I(f_i; C) - \frac{1}{|S|^2} \sum_{f_i, f_j \in S} I(f_i; f_j) \qquad (8)$$

The task of feature subset selection by means of the mRMR measure: Suppose that there are *n* full set features. The task of feature selection by means of this mRMR measure is to look for the subset *S*, which has the maximum value $\max_S \phi_S(D,R)$ of over all 2^n possible feature subsets:

For the tasks of feature subset selection by means of the CFS and the mRMR measures, the exhaustive search method can be applied to scan all possible feature subsets when the number *n* is small. But when this number becomes large, the heuristic and random search strategies, such as the best first search or genetic algorithm, are commonly chosen due to their computational efficiency. Consequently, the obtained results will always be approximate. It is desirable to obtain optimal subsets of features. In the sequel, the generalization of these two feature selection measures into a generic feature selection (GeFS) measure and a new search method that ensures globally optimal feature subsets by means of GeFS will be introduced.

Generic Feature Selection Measure for Intrusion Detection

Definition 1: A generic feature selection measure applied in the filter model for intrusion detection is a function *GeFS(x)* , which has the following form (Nguyen et al., 2010b):

$$GeFS(x) = \frac{a_0 + \sum_{i=1}^{n} A_i(x)x_i}{b_0 + \sum_{i=1}^{n} B_i(x)x_i}, x \in \{0,1\}^n \qquad (9)$$

In this definition, binary values of the variable x_i indicate the appearance $x_i = 1$ or the absence $x_i = 0$ of the feature f_i; a_0, b_0 are constants; $A_i(x)$, $B_i(x)$ are linear functions of variables x_1, x_2, ..., x_n.

Definition 2: The feature selection problem is to look for $x \in \{0,1\}^n$ that maximizes the function *GeFS(x)* (Nguyen et al., 2010b)

$$\max_{x \in \{0,1\}^n} GeFS(x) = \frac{a_0 + \sum_{i=1}^{n} A_i(x)x_i}{b_0 + \sum_{i=1}^{n} B_i(x)x_i} \qquad (10)$$

Proposition 1: The CFS measure and the mRMR measure are instances of the GeFS measure (Nguyen et al., 2010b).

Proof. The binary values of the variable x_i are utilized in order to indicate the appearance $x_i = 1$ or the absence $x_i = 0$ of the feature f_i in the globally optimal feature subset. The problems (7) and (8) can be described as optimization problems (11) and (12), respectively as follows:

$$\max_{x \in \{0,1\}^n} \left[\frac{(\sum_{i=1}^{n} r_{Cf_i} x_i)^2}{\sum_{i=1}^{n} x_i + \sum_{i \neq j} 2r_{f_i f_j} x_i x_j} \right] \qquad (11)$$

and

$$\max_{x \in \{0,1\}^n} \left[\frac{\sum_{i=1}^{n} I(f_i;C)x_i}{\sum_{i=1}^{n} x_i} - \frac{\sum_{i,j=1}^{n} I(f_i;f_j)x_i x_j}{(\sum_{i=1}^{n} x_i)^2} \right] \qquad (12)$$

It is obvious that the CFS measure and the mRMR measure are instances of the GeFS measure. □

In the following, the optimization problems (11) and (12) are considered as polynomial mixed 0−1 fractional programming (P01FP) problems and it is shown how to solve these problems.

Polynomial Mixed 0-1 Fractional Programming

A general polynomial mixed *0−1* fractional programming (P01FP) problem (Chang, 2001) is represented as follows:

$$\min \sum_{i=1}^{m} \left(\frac{a_i + \sum_{j=1}^{n} a_{ij} \prod_{k\in J} x_k}{b_i + \sum_{j=1}^{n} b_{ij} \prod_{k\in J} x_k} \right) \qquad (13)$$

subject to the following constraints:

$$\begin{cases} b_i + \sum_{j=1}^{n} b_{ij} \prod_{k\in J} x_k > 0, i = 1,...,m, \\ c_p + \sum_{j=1}^{n} c_{pj} \prod_{k\in J} x_k \le 0,\ p = 1,...,m, \\ x_k \in \{0,1\}, k \in J, \\ a_i, b_i, c_p, a_{ij}, b_{ij}, c_{pj} \in R. \end{cases}$$

By replacing the denominators in (13) by positive variables $y_i (i = 1...m)$, the P01FP then leads to the following equivalent polynomial mixed *0−1* programming problem:

$$\min \sum_{i=1}^{m} (a_i y_i + \sum_{j=1}^{n} a_{ij} \prod_{k\in J} x_k y_i) \qquad (14)$$

subject to the following constraints:

$$\begin{cases} b_i y_i + \sum_{j=1}^{n} b_{ij} \prod_{k\in J} x_k y_i = 1, i = 1,...,m, \\ c_p + \sum_{j=1}^{n} c_{pj} \prod_{k\in J} x_k \le 0,\ p = 1,...,m, \\ x_k \in \{0,1\}, k \in J, y_i > 0, \\ a_i, b_i, c_p, a_{ij}, b_{ij}, c_{pj} \in R. \end{cases} \qquad (15)$$

In order to solve this problem, Chang (Chang, 2001) proposed a linearization technique to transfer the terms $\prod_{k\in J} x_k y_i$ into a set of mixed *0−1* linear inequalities. Basing on this technique, the P01FP then becomes a mixed *0−1* linear programming (M01LP) which can be solved by means of the branch-and-bound method to obtain the globally optimal solution.

Proposition 2: A polynomial mixed *0−1* term $\prod_{k\in J} x_k y_i$ from (14) can be represented by the following program (Chang, 2000):

$$\min z_i$$

subject to the following constraints:

$$\begin{cases} z_i \geq M(\sum_{k\in J} x_k - |J|) + y_i, \\ z_i \geq 0, \end{cases} \qquad (16)$$

where M is a large positive value.

Proposition 3: A polynomial mixed *0−1* term $\prod_{k\in J} x_k y_i$ from (15) can be represented by a continuous variable v_i, subject to the following linear inequalities (Chang, 2000):

$$\begin{cases} v_i \geq M(\sum_{k\in J} x_k - |J|) + y_i, \\ v_i \leq M(|J| - \sum_{k\in J} x_k) + y_i, \\ 0 \leq v_i \leq Mx_i, \end{cases} \qquad (17)$$

where M is a large positive value.

In the following, the optimization problem of the GeFS measure (10) is formulated as a polynomial mixed *0−1* fractional programming (P01FP) problem.

Proposition 4: The optimization problem of the GeFS measure (10) can be considered as a polynomial mixed *0−1* fractional programming (P01FP) problem (Nguyen et al., 2010b).

Proof. The sign of *GeFS(x)* in (10) is changed to make a minimum problem. Therefore, (10) can be written as (13). □

Remark: By applying the Chang's method, this P01FP problem is transformed to the M01LP problem. The number of variables and constraints will depend on the square of *n*, where *n* is the number of features, because the number of terms $\prod_{k\in J} x_k y$, which are replaced by the new variables in forms $(\sum_{i\neq j} 2a_i a_j x_i x_j y)$ or $(\sum_{i\neq j} 2b_{ij} x_i x_j y)$, is *n(n − 1)/2*. The branch-and-bound algorithm then can be utilized to solve this M01LP problem. However, the efficiency of the method depends strongly on the number of variables and constraints. The larger the number of variables and constraints an M01LP has, the more computational complexity the branch-and-bound algorithm has.

In the next section, an improvement of the Chang's method is presented to obtain an M01LP with a linear number of variables and constraints in the number of full set variables. A new search strategy to obtain the relevant subsets of features by means the GeFS measure is also provided.

Optimization of the GeFS Measure

By introducing an additional positive variable, denoted by *y*, it is now considered that the following problem equivalent to (10):

$$\min_{x\in\{0,1\}^n}\{-GeFS(x)\} = -a_0 y - \sum_{i=1}^{n} A_i(x) x_i y \qquad (18)$$

subject to the following constraints:

$$\begin{cases} y > 0, \\ x = (x_1, x_2, ..., x_n) \in \{0,1\}^n, \\ b_0 y + \sum_{i=1}^{n} B_i(x) x_i y = 1 \end{cases} \qquad (19)$$

This problem is transformed into a mixed 0-1 linear programming problem as follows:

Proposition 5: A term $A_i(x)x_i y$ from (18) can be represented by the following program (Nguyen et al., 2010b):

$$\min z_i$$

subject to the following constraints:

$$\begin{cases} z_i \geq M(x_i - 1) + A_i(x)y, \\ z_i \geq 0, \end{cases} \qquad (20)$$

where M is a large positive value.

Proof.

a) If $x_i = 0$, then $z_i \geq M(0-1) + A_i(x)y \leq 0$ will cause $\min z_i$ to be zero, because $z_i \geq 0$ and M is a large positive value.

b) If $x_i = 1$, then $z_i \geq M(1-1) + A_i(x)y \geq 0$ will cause $\min z_i$ to be $A_i(x)y$, because $z_i \geq 0$.

Therefore, the above program on z_i reduces to:

$$\min z_i = \begin{cases} 0, \ if\ x_i = 0 \\ A_i(x)y, \ if\ x_i = 1 \end{cases}$$, which is the same as $A_i(x)x_i y = \min z_i$. □

Proposition 6: A term $B_i(x)x_i y$ from (19) can be represented by a continuous variable v_i, subject to the following linear inequality constraints (Nguyen et al., 2010b):

$$\begin{cases} v_i \geq M(x_i - 1) + B_i(x)y, \\ v_i \leq M(1 - x_i) + B_i(x)y \\ 0 \leq v_i \leq Mx_i, \end{cases} \qquad (21)$$

where M is a large positive value.

Proof.

a) If $x_i = 0$, then (15) becomes

$$\begin{cases} v_i \geq M(0-1) + B_i(x)y, \\ v_i \leq M(1-0) + B_i(x)y, \\ 0 \leq v_i \leq 0, \end{cases}$$

v_i is caused to be zero, as M is a large positive value.

b) If $x_i = 1$, then (15) becomes
$$\begin{cases} v_i \geq M(1-1) + B_i(x)y, \\ v_i \leq M(1-1) + B_i(x)y, \\ 0 \leq v_i \leq M, \end{cases}$$

v_i is caused to be $B_i(x)y$, as *M* is a large positive value.

Therefore, the constraints on v_i reduce to:

$$v_i = \begin{cases} 0, \ if \ x_i = 0 \\ B_i(x)y, \ if \ x_i = 1 \end{cases}$$

which is the same as $B_i(x)x_i y = v_i$.

Each term $x_i y$ in (20), (21) is substituted by new variables t_i satisfying constraints from Proposition 2. Then the total number of variables for the M01LP problem will be *4n+1*, as they are x_i , y, t_i , z_i and $v_i (i = 1...n)$. Therefore, the number of constraints on these variables will also be a linear function of *n*. As mentioned above, in the Chang's method (Chang, 2001) the number of variables and constraints depends on the square of *n*, thus the new method introduced here actually improves Chang's method by reducing the complexity of branch and bound algorithm.

In the following, a new search strategy for obtaining the best subset of relevant features by means of the GeFS measure is presented:

The new search method for relevant feature subsets by means of the GeFS measure:

- **Step 1:** Analyzing the statistical properties of the given data set in order to choose an appropriate feature selection instance (CFS or mRMR) from the generic feature selection measure GeFS. The CFS is selected in the case if the data set has many features that linearly correlate to the class label and to each others. Otherwise, the mRMR measure is chosen.
- **Step 2:** Calculating all feature-feature and feature-classification correlations or mutual information corresponding to the statistical properties of the training data set.
- **Step 3:** Constructing the optimization problem (7) or (8) from the correlations or mutual information calculated above. In this step, domain knowledge can be utilized by assigning the value *1* to the variable x_i if the feature f_i should be in the final selected feature subset and the value *0* otherwise.
- **Step 4:** Transforming the optimization problem of GeFS to a mixed *0−1* linear programming (M01LP) problem, which is to be solved by the branch-and-bound algorithm. A non-zero integer value of x_i from the globally optimal solution indicates the relevance of the feature f_i regarding to the GeFS measure.

METHOD APPICATIONS

In this section, first, the 1998 DARPA data set (Lippmann et al., 1998 and Lippmann et al., 2000), which was generated by MIT Lincoln Laboratory for off-line intrusion detection evaluation, is described. Then it is shown how to construct the KDD CUP 1999 benchmarking data set (Lee, 1999) from the 1998 DARPA data set by utilizing feature construction methods that have been introduced in the previous sections. Various feature selection methods, which were applied to select features from the KDD CUP 1999 data set, will be described. Finally, a comparison of those feature selection algorithms for intrusion detection will be shown. The KDD CUP 1999 data set is considered, since it was a data set for evaluating IDSs and most the feature selection methods for intrusion detection were practiced on it.

The 1998 DARPA and the KDD CUP 1999 Data Sets

In 1998, MIT's Lincoln Laboratory launched a research project to evaluate the different intrusion detection systems, sponsored by the Air Force Research Laboratory (Lippmann et al., 1998). The goal of the project was mainly to create a benchmarking data set for intrusion detection by simulating background traffic and attack traffic. It took seven weeks to gather the training data and two weeks for the testing data were took place. The generated traffic was similar to that on a government sites containing hundreds of users. Custom software automata simulated hundreds of programmers, secretaries, managers and other types of users running common UNIX application programs. Many types of traffic were created by using a variety of network services. User automata sent and received emails, browsed websites, sent and received files using FTP or used Telnet to log into remote computers and performed works and so on. A more detail of the simulated network is described as follows. The inside of the Air Force base network contains three machines which were the most frequent victims of attacks (Linux2.0.27, SunOS 4.1.4 and Sun Solaris 2.5.1), and a gateway to hundreds of other inside emulated PCs and workstations. The outside of this network simulated the Internet. It contained a sniffer to capture traffic, a gateway to hundreds of emulated workstations on many other subnets and a second gateway to thousands of emulated web servers. Data collected for evaluating IDSs included network sniffing data from the outside sniffer, Sun Basic Security Module (BSM) audit data captured from the Solaris hosts and full disk dumps from the three UNIX victim machines.

For the normal traffic, a large amount of web, telnet and email traffic was generated between the inside PS's with workstations and the outside workstations with the websites. In addition, there are many user automata of various types (e.g. secretaries, managers and programmers) on outside workstations, who performed work using telnet and other services on the three inside victim machines, and the other inside workstations. The contents of network traffic, such as SMTP, HTTP and FTP file transfers, as they mentioned are either statistically similar to live traffic, or sampled from public-domain sources. For example, some email message contents were created using statistical bigrams frequencies to preserve word and two-word sequence statistics from a sampling of roughly 10,000 actual email messages to and from computer professionals filtered using a 40,000 word dictionary to remove names and other private information. Similar approaches were applied to produce content for FTP file transfer. The contents of the web servers were initially captured using a custom web automaton that was run on the real Internet. This automaton was programmed to visit thousands of websites popular with university and government personnel with a frequency that depends on the site's popularity and to visit a random number of links at each site before traversing to another sites.

For the attack traffic, there were more than 3,000 instances of 38 different simulated attacks against victim UNIX hosts. All the attacks can be categorized into 4 main groups: Denial of Service (DoS) attacks, Probe attacks, User to Root (U2R) attacks and Remote to Local (R2L) attacks. In more details, DoS attacks (e.g. smurt) load a legitimate network service, others (e.g. teardrop, Ping of Death) create malformed packets, which are incorrectly handled by the victim machine, and others (e.g. apache2, back, syslogd) take advantage of software bugs in network daemon programs. The Probe attacks of Scan attacks are programs that can automatically scan a network of computers to gather information and search for known vulnerabilities. The U2R attacks attempt to obtain privileges normally reserved for the root or super users. In the case of R2L attacks, an attacker, who does not have an account on a victim machine. sends packets to that machine and gain local access. Some R2L attacks exploit buffer overflow in network server software (e.g. imap, named, sendmail), others exploit weak or misconfigured security policies (e.g. dictionary, ftp-write, guest) and one (xsnoop) is a trojan password capture program.

Feature Construction from the 1998 DARPA to the KDD CUP 1999 Data Sets

In 1999, Lee W. and Stolfo S. (Lee, 1999 and Lee & Stolfo, 2000) proposed a novel framework, MADAM ID, that applies data mining algorithms to extract frequent patterns from system audit data and construct predictive features from the patterns. Machine learning algorithms then were applied as intrusion detection systems to distinguish attacks from normal traffic in the audit records that are represented by feature vectors. They conducted the experiments on the 1998 DARPA data set as follows: the raw tcpdump data sets provided by the Lincoln Laboratory (Lippmann et al., 1998) were fist summarized into network connection records, in which a set of intrinsic features from domain knowledge was utilized. An example of network connection records and the list of intrinsic features are given in Table 1 and Table 2 as below:

Table 1. An example of network connection records (Lee and Stolfo, 2000)

Timestamp	duration	Service	src_host	dst_host	src_bytes	dst_bytes	flag
1.1	0	http	spoofed_1	Victim	0	0	S0
1.1	0	http	spoofed_2	Victim	0	0	S0
10.2	2	ftp	A	B	200	300	SF
12.3	1	smtp	B	D	250	300	SF
13.4	60	telnet	A	D	200	12100	SF
...	...	...	...	...	...	...	...

Table 2. Intrinsic features of individual TCP connections (Lee and Stolfo, 2000)

ID	Feature Name	Description	Type
1	duration	length (number of seconds) of the connection	continuous
2	protocol_type	type of the protocol, e.g. tcp, udp, etc.	discrete
3	dervice	network service on the destination, e.g., http, telnet, etc.	discrete
4	src_bytes	number of data bytes from source to destination	continuous
5	dst_bytes	number of data bytes from destination to source	continuous
6	flag	normal or error status of the connection	discrete
7	land	1 if connection is from/to the same host/port; 0 otherwise	discrete
8	wrong_fragment	number of "wrong" fragments	continuous
9	urgent	number of urgent packets	continuous

The association rule learning algorithms were applied to search frequent associations or relations between intrinsic features. From those associations, Lee W. and Stolfo S. generated frequent episodes of sequential patterns of both the gathered normal and the attack traffic. They then compared the obtained patterns to look for the intrusion-only patterns that appear only in the intrusion data sets. Those intrusion patterns were utilized as guidelines for constructing additional features to build better classification models. In the Table 3, there is an example of frequent SYN flood patterns that they found in the audit data.

Table 3. Example Intrusion Pattern (Lee and Stolfo, 2000)

Frequent Episode	Meaning
(flag=S0, service=http,dst_host=victim), (flag=S0, service=http,dst_host=victim) $\Rightarrow$ (flag=S0, service=http,dst_host=victim) [0.93, 0.03, 2]	93% of the time, after two *http* connections with *S*0 flag are made to host *victim*, within 2 seconds from the first of these two, the third similar connection is made, and this pattern occurs in 3% of the data.

This SYN flood pattern guides to construct the following additional features: a count of connections to the same *dst_host* in the past 2 seconds, and among these connections, the percentage of those that have the same *service*, and the percentage of those that have the “S0” *flag*.

Here the additional features automatically constructed by Lee's proposed method (Lee and Stolfo, 2000) are summarized as follows:

- the “same host” features that examine only the connections in the past 2 seconds that have the same destination host as the current connection: the count of such connections, the percentage of connections that have the same service as the current one, the percentage of different services, the percentage of SYN errors, and the percentage of REJ (i.e., rejected connection) errors.

- the “same service” features that examine only the connections in the past 2 seconds that have the same service as the current connection: the count of such connections, the percentage of different destination hosts, the percentage of SYN errors, and the percentage of REJ errors.

They called these features "time-based" features for connection records. For several "slow" Probe attacks that scan the destination hosts or ports using more time than 2 seconds, there were not any intrusion-only patterns of these attacks within 2 seconds of connection. Therefore, Lee W. and Stolfo S. proposed to sort the connection records by the destination hosts and to consider 100 connections instead of the time window of 2 seconds. The automatic feature construction algorithms were applied again to obtain a mirror set of "host-based traffic" features as the "time-based traffic" features. All names of "time-based" and "host-based traffic" features are listed in the Table 4 and Table 5, respectively.

As many attacks, such as R2L and U2R attacks, are embedded in the payloads of the packets and involve only a single connection, the automatic feature construction algorithm, which is based on frequent sequential patterns of connection records, would fail to create any features of these attacks. Therefore, Lee W. and Stolfo S. proposed to look at the payloads of packets and combine it with domain knowledge to define suitable features for R2L and U2R attacks. These features are (Lee and Stolfo, 2000): number of failed logins, successfully logged in or not, whether logged in as root, whether a root shell is obtained, whether a *su* command is attempted and succeeded, number of access to access control files (e.g., “/etc/passwd”, “.rhosts”, etc.), number of compromised states on the destination host (e.g., file/path “not found” errors, and “Jump to” instructions, etc.), number of hot indicators, (e.g., access to system directories, creation and execution of programs, etc.), and number of outbound connections during a *ftp* session. These features are summarized in Table 6.

Table 4. Traffic features computed using a two-second time window (Lee and Stolfo, 2000)

ID	Feature Name	Description	Type
10	count	number of connections to the same host as the current connection in the past two seconds	Continuous
Note: The following features refer to these same-host connections.			
11	serror_rate	% of connections that have "SYN" errors	Continuous
12	rerror_rate	% of connections that have "REJ" errors	Continuous
13	same_srv_rate	% of connections to the same service	Continuous
14	diff_srv_rate	% of connections to different services	Continuous
15	srv_count	number of connections to the same service as the current connection in the past two seconds	Continuous
Note: The following features refer to these same-service connections.			
16	srv_serror_rate	% of connections that have "SYN" errors	Continuous
17	srv_rerror_rate	% of connections that have "REJ" errors	Continuous
18	srv_diff_host_rate	% of connections to different hosts	Continuous

Table 5. Traffic features computed using a window of 100 connections (Lee and Stolfo, 2000)

ID	Feature Name	Description	Type
19	dst_host_count	number of connections to the same host as the current connection in the past two seconds	Continuous
Note: The following features refer to these same-host connections.			
20	dst_host_serror_rate	% of connections that have "SYN" errors	Continuous
21	dst_host_rerror_rate	% of connections that have "REJ" errors	Continuous
22	dst_host_same_srv_rate	% of connections to the same service	Continuous
23	dst_host_diff_srv_rate	% of connections to different services	Continuous
24	dst_host_srv_count	number of connections to the same service as the current connection in the past two seconds	Continuous
Note: The following features refer to these same-service connections.			
25	dst_host_srv_serror_rate	% of connections that have "SYN" errors	Continuous
26	dst_host_srv_rerror_rate	% of connections that have "REJ" errors	Continuous
27	dst_host_srv_diff_host_rate	% of connections to different hosts	Continuous
28	dst_host_same_src_port_rate	% of connections to the same source port	Continuous

Table 6. Content features within a connection suggested by domain knowledge (Lee and Stolfo, 2000)

ID	Feature Name	Description	Type
29	hot	number of "hot" indicators	Continuous
30	num_failed_logins	number of failed login attempts	Continuous
31	logged_in	1 if successfully logged in; 0 otherwise	Discrete
32	num_compromised	number of "compromised" conditions	Continuous
33	root_shell	1 if root shell is obtained; 0 otherwise	Discrete
34	su_attempted	1 if "su root" command attempted; 0 otherwise	Discrete
35	num_root	number of "root" accesses	Continuous
36	num_file_creations	number of file creation operations	Continuous
37	num_shells	number of shell prompts	Continuous
38	num_access_files	number of operations on access control files	Continuous
39	num_outbound_cmds	number of outbound commands in an ftp session	Continuous
40	is_hot_login	1 if the login belongs to the "hot" list; 0 otherwise	Discrete
41	is_guest_login	1 if the login is a "guest" login; 0 otherwise	Discrete

Feature Selection from the KDD CUP 1999 Data Set

In this section, we first describe the application of several existing feature selection methods, such as Markov Blanket, Support Vector Machine Wrapper and Classification and Regression Trees, on the KDD CUP 1999 data set. We then show the performance of the recently proposed generic feature selection (Ge_FS) measure for intrusion detection on this data set. Finally, a comparison between these feature selection methods is given.

Markov Blanket

Markov blanket *MB(C)* of the output variable *C* is defined as the set of input variables such that all other variables are probabilistically independent of *C*. Knowledge of *MB(C)* is sufficient for perfectly estimating the distribution of the *C*. Therefore, Markov blanket can be applied for feature selection (Koller and Sahami, 1996). In 2005, Chebrolu et al. (Chebrolu et al., 2005) have proposed to apply Markov blanket for selecting important features for intrusion detection. In order to conduct the experiment, they constructed a Bayesian Network (BN) (Duda et al., 2001) from the original data set. A Bayesian network $B = (N,A,Q)$ is a Directed Acyclic Graph (DAG) (N,A) where each node $n \in N$ represents a domain variable (e.g. a data set attribute or variable), and each arc $a \in A$ between nodes represents a probabilistic dependency among the variables. A BN can be utilized to compute the conditional probability of one node, given values assigned to the other nodes. From the constructed BN, the Markov blanket of the class label *C* is the union of *C* 's parents, *C* 's children and eventually other parents of *C* 's children. For conducting the experiment, Chebrolu et al. (Chebrolu et al., 2005) randomly chose 11,982 instances from the overall (5 millions of instances) KDD CUP 1999 data set (Lee, 1999). 17 features were selected and the Bayesian Network (Duda et al, 2001) was applied for classifying the obtained data set after removing irrelevant features. The results are given in the Table 7.

Table 7. Performance of Bayesian Network using selected features (Chebrolu et al., 2005)

Classes	Number-of-selected-features	Accuracy
KDD99-normal	17	99.64%
KDD99-DoS	17	98.16%
KDD99-Probe	17	98.57%
KDD99-U2R	17	60.00%
KDD99-R2L	17	98.93%

Support Vector Machine Wrapper

Sung and Mukkamala (Sung and Mukkamala, 2003) have utilized the ranking methodology to select important features for intrusion detection: one input feature is deleted from the data at a time and the resultant data set is then utilized for training and testing the classifier Support Vector Machine (SVM) (Vapnik, 1995). Then the SVMs performance was compared to that of the original SVM (based on all features) in terms of relevant performance criteria, such as overall accuracy of classification, training time and testing time. The deleted feature was ranked as ”important”, ”secondary” or ”insignificant” according to the following rules (Sung and Mukkamala, 2003):

- If accuracy decreases **and** training time increases **and** testing time decreases, **then** the feature is important.
- If accuracy decreases **and** training time increases **and** testing time increases, **then** the feature is important.
- If accuracy decreases **and** training time decreases **and** testing time increases, **then** the feature is important.
- If accuracy is not changed **and** training time increases **and** testing time increases, **then** the feature is important.
- If accuracy is not changed **and** training time decreases **and** testing time increases, **then** the feature is secondary.
- If accuracy is not changed **and** training time increases **and** testing time decreases, **then** the feature is secondary.
- If accuracy is not changed **and** training time decreases **and** testing time decreases, **then** the feature is insignificant.
- If accuracy increases **and** training time increases **and** testing time decreases, **then** the feature is secondary.
- If accuracy increases **and** training time decreases **and** testing time increases, **then** the feature is secondary.
- If accuracy increases **and** training time decreases **and** testing time decreases, **then** the feature is insignificant.

The experiment in (Sung and Mukkamala, 2003) was conducted on a part of KDD CUP’99 data set (Lee, 1999). This data set contains normal traffic and four main attack classes: Denial-of-Service (DoS) attacks, Probe attacks, User-to-Root (U2R) attacks and Remote-to-Local (R2L) attacks. Some important features were selected and the obtained data set after removing irrelevant features was classified by SVM. The results are given in Table 8.

Table 8. Performance of SVM using selected feature (Sung and Mukkamala, 2003)

Classes	Number-of-selected-features	Accuracy
KDD99-normal	25	99.59%
KDD99-DoS	19	99.22%
KDD99-Probe	7	99.38%
KDD99-U2R	8	99.87%
KDD99-R2L	6	99.78%

Classification and Regression Trees

The Classification and Regression Trees (CART) (Breiman, 1984) is an approach of the decision tree learning techniques. This method is based on binary recursive partitioning. The process is binary because parent nodes are always split into exactly two child nodes and recursive because it is repeated by treating each child node as a parent. The key elements of CART methodology are a set of splitting rules in a tree; deciding when the tree is complete and assigning a class to each terminal node. Feature selection for intrusion detection is based on the contribution of the input variables to the construction of the decision tree from the original data set. The importance of features is determined by the role of each input variable either as a main splitter or as a surrogate. Surrogate splitters are considered as back-up rules that closely mimic the action of primary splitting rules. For example, in the given model, the algorithm splits data according to the variable *protocol type* and if a value for *protocol type* is not available then the algorithm might utilize the *service* feature as a good surrogate. Feature importance, for a particular feature is the sum across all nodes in the tree of the improvement scores that the predictor has when it acts as a primary or surrogate splitter. For example, for the node *i*, if the feature appears as the primary splitter then its importance could be given as $i_{IMPORTANCE}$. But if the feature appears as the n^{th} surrogate instead of the primary variable, then the importance becomes $i_{IMPORTANCE} = (p^n) \times i_{IMPORTANCE}$, in which p is the *surrogate improvement weight* which is a user controlled parameter set between 0 and 1.

Chebrolu et al. (Chebrolu et al., 2005) have conducted the experiment on the data set, which contains randomly chosen 11,982 instances from the overall (5 millions of instances) KDD CUP 1999 data set (Lee, 1999). 12 features were selected and the CART (Breiman, 1984) was used for classifying the obtained data set after removing irrelevant features. The results are given in Table 9.

Table 9. Performance of CART using selected features (Chebrolu et al., 2005)

Classes	Number-of-selected-features	Accuracy
KDD99-normal	12	100%
KDD99-DoS	12	85.34%
KDD99-Probe	12	97.71%
KDD99-U2R	12	64.00%
KDD99-R2L	12	95.56%

Generic Feature Selection Measure for Intrusion Detection

As the generic feature selection (GeFS) measure for intrusion detection has two instances: the correlation feature selection (CFS) measure and the minimal-redundancy-maximal-relevance (mRMR) measure, and following the new method proposed above for obtaining relevant feature subsets by means of the GeFS measure, it is necessary to choose an appropriate measure depending statistical properties of the KDD CUP 1999 data set. In order to make the choice, the whole data set is visualized on two-dimension space to obtain a plot matrix, in which each element is the distribution of data points depending on the values of a feature and the class label or the values of two features. For example, the figure below shows the distribution of more than 32,000 data points of the KDD CUP 1999 data set according to values of "count" and "duration" features.

It can be observed that many features has no correlation with the class label, meanwhile they linearly correlate to each others. In fact, more than 65% of all feature-class correlation coefficients are very low (less than 0.01), meanwhile more than 35% of all feature-feature correlation coefficients are very high (more than 0.5). Therefore, the CFS-based approach for selecting features from the KDD CUP 199 data set was opted. In the following, an experiment was conducted to validate the new CFS-based feature selection method as well as to compare it with the new mRMR-based approach and other heuristic algorithms.

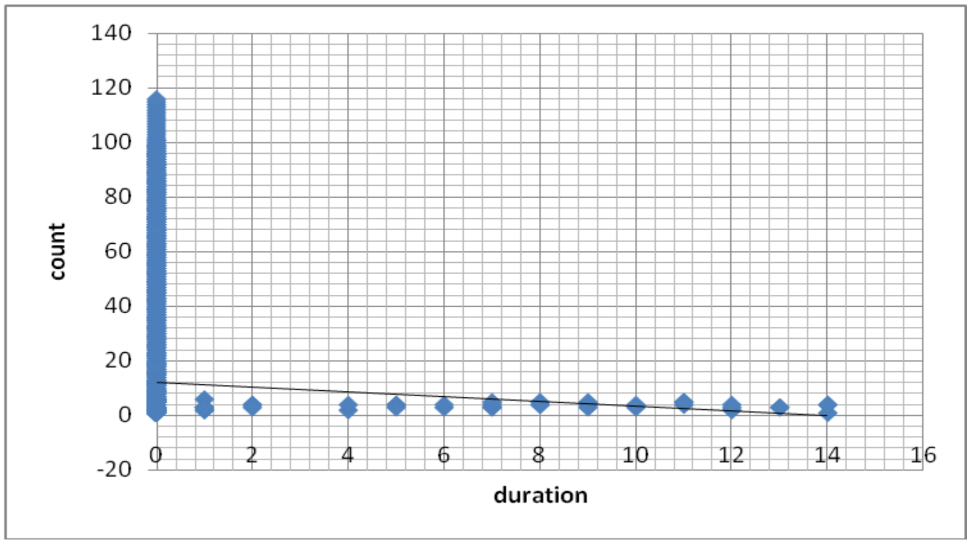

Figure 3. Example distribution of data points from the KDD CUP 1999

Experimental Setting

For evaluating the performance of the new CFS-based feature selection approach for intrusion detection introduced in this study, the new mRMR-based approach and two available feature selection methods based on the CFS measure (Chen, 2006) were implemented. One is the best-first-CFS method, which employed the best first search strategy to obtain the locally optimal subset. The other employed the genetic algorithm for searching. Note that the best first search and genetic algorithm may not guarantee to obtain the globally optimal solution. However, this issue could be overcome with the new method. The exhaustive search method was not selected since it was not feasible for feature selection from data sets with a large number of features. Even for this experiment, no access to required computing resource was provided. Machine learning algorithms were applied for evaluating the classification accuracy on selected features, since there was no standard IDS.

The experiment was performed with 10% of the overall (5 millions) KDD CUP’99 IDS benchmarking labeled data. This data set contained normal traffic and four main attack classes: (i) Denial of Service (DoS) attacks, (ii) Probe attacks, (iii) User to Root (U2R) attacks, and (iv) Remote to Local (R2L) attacks. The number of instances for the four attack classes and normal class was relatively different, e.g. the relation of the number of U2R to the number of DoS was $1.3*10^{-4}$. Details of the number of class instances are given in Table 10.

Table 10. The partition of KDD CUP'99 dataset utilized in experiment (Nguyen et al., 2010a)

Classes	**Number-of-instances**	**Percentage**
KDD99-normal	97,278	18.30%
KDD99-DoS	391,458	73.74%
KDD99-Probe	41,113	7.74%
KDD99-U2R	52	0.01%
KDD99-R2L	1,126	0.21%
Total	**531,027**	**100%**

In more details the performance of the newly proposed CFS-based feature selection method was testified as follows:

1) Feature selection is performed on the basis of the whole data set: (1a) Each attack class and the normal class are processed individually, so that a five-class problem can be formulated for feature extraction and classification with one single classifier. (1b) All attack classes are fused so that a two-class problem can be formulated, meaning the feature selection and classification for normal and abnormal traffic is performed. It might be well possible that the attack-recognition results are not satisfactory for all of the classes, since the number of class instances are unevenly distributed, in particular classes U2R and R2L are under-represented. The feature selection algorithm and the classifier, which is utilized for evaluation of the detection accuracy on selected features, might concentrate only on the most frequent class data and neglect the others. As a consequence, relevant characteristics of the less representative classes might be missed.

2) As the attack classes distributed differently, it is preferred to process these attack classes separately. With the specific application of IDS the four different two-class problems could also be formulated. Four classifiers were derived utilizing specific features for each classifier in order to detect (identify) a particular attack. The reason for this approach was that the most accurate classification was predicted if each of the four intrusion detectors (classifiers) was fine-tuned according to the significant features. This approach might also be very effective, since the four light-weight classifiers could be operated in parallel.

Table 11. Unclassified instances (UI) by the C4.5 classifier (Nguyen et al., 2010a)

Classes	Number of UI	Percentage
Normal	65	0.07%
DoS	21	0.01%
Probe	39	0.10%
U2R	18	34.6%
R2L	39	3.46%

To understand the effect, as mentioned in 1), a small experiment was conducted. The aim was to show that the classifier highly neglected U2R attack instances. In order to carry through the experiment, all attack classes were mixed to obtain only one data set and considered the five-class (normal, DoS, Probe, U2R, and R2L) problem. The C4.5 machine learning algorithm was employed as a classifier. Five-folds cross-validation for evaluating the detection accuracy of the C4.5 were applied. The result of the experiment is given in Table 11. It can be seen from Table 11 that the C4.5 highly misclassified U2R attack instances with 34.6% error.

In order to perform the experiment 2), normal traffic was added into each attack class to obtain four data sets: KDD99-normal&DoS, KDD99-normal&Probe, KDD99-normal&U2R and KDD99-normal&R2L. With each data set, three feature-selection algorithms were run: the new CFS-based method proposed in this study, the best-first CFS-based, and the genetic algorithm CFS-based methods. The numbers of selected features and their identifications are given in Tables 12 and 13, respectively. Then the C4.5 and the BayesNet machine learning algorithm were applied on each original full set as well as each newly obtained data set that included only the selected features from feature selection algorithms. By applying 5-folds cross-validation evaluation on each data set, the classification accuracies are reported in Tables 14 and 15.

The new CFS-based method proposed in this study was compared with the new mRMR-based approach, the best first-CFS, and the genetic-algorithm-CFS methods regarding the number of selected features and regarding the classification accuracies of 5-folds cross-validation of the BayesNet and C4.5 learning algorithms. Weka tool (Witten et al., 1999) was employed to obtain the results. In order to solve the M01LP problem, the TOMLAB tool (Kenneth et al., 2003) was utilized. All the obtained results are listed in Tables 12, 14, and 15.

Table 12. Number of selected features (GA: genetic algorithm) (Nguyen et al., 2010a)

Data Set	Full-set	GeFS_CFS	GeFS_mRMR	Best-first	GA
KDD99-normal&Dos	41	3	22	6	11
KDD99-normal&Probe	41	6	14	7	17
KDD99-normal&U2R	41	1	5	4	8
KDD99-normal&R2L	41	2	6	5	8

Table 13. Identification of selected features by means of GeFS_CFS (Nguyen et al., 2010a)

Data Set	**Identifications**
KDD99-normal&Dos	4, 5, 31
KDD99-normal&Probe	4, 5, 13, 26, 27, 31
KDD99-normal&U2R	33
KDD99-normal&R2L	29, 41

Table 14. Classification accuracies of C4.5 performed on KDD CUP'99 (Nguyen et al., 2010a)

Data Set	**Full-set**	**GeFS_CFS**	**GeFS_mRMR**	**Best-first**	**GA**
KDD99-normal&Dos	97.80	98.89	99.98	96.65	96.09
KDD99-normal&Probe	99.98	99.70	99.35	99.71	99.89
KDD99-normal&U2R	99.97	99.96	99.94	99.97	99.95
KDD99-normal&R2L	98.70	99.11	99.14	99.01	98.86
Average	**99.11**	**99.41**	**99.60**	**98.84**	**98.69**

Table 15. Classification accuracies of BayesNet performed on KDD CUP'99 (Nguyen et al., 2010a)

Data Set	**Full-set**	**GeFS_CFS**	**GeFS_mRMR**	**Best-first**	**GA**
KDD99-normal&Dos	99.99	98.87	99.36	99.09	99.72
KDD99-normal&Probe	98.96	97.63	98.65	97.65	99.19
KDD99-normal&U2R	99.85	99.95	99.94	99.97	99.93
KDD99-normal&R2L	99.33	98.81	99.17	98.95	99.28
Average	**99.53**	**98.82**	**99.28**	**98.91**	**99.52**

Experimental Results

Table 12 shows the number of features selected by CFS-based approach and those selected by utilizing the mRMR-based method, the best-first and GA search strategies. The identification of selected features is given in Table 13 (for feature names, see Tables 2, 4, 5 and 6). Table 14 and Table 15 summarize the classification accuracies of the BayesNet and the C4.5, respectively, performed on four data sets (see above).

It can be observed from Table 12 that the CFS-based approach introduced in this study selects the smallest number of relevant features in comparison with the full and the feature sets selected by the mRMR-based method, the best-first and GA search strategies. In particular, in some cases, the new method introduced in this study extremely compresses the full set of features. For example, only one feature was selected out of 41 features of the KDD99-normal&U2R data set.

In the Table 14 and Table 15 , it can be observed that with this study approach the average classification accuracies are slightly different from the ones obtained by utilizing the best-first search or the genetic algorithm. The absolute difference between them does not exceed 0.69%. In the case of the C4.5 classifier, the better performance was an evident. Even though the gain of classification accuracy is not very high compared to other methods, the overall gain of the feature selection classification procedure lies in significantly improved efficiency and in the obtained classification results due to the reduced number of relevant features.

Therefore, based on all these experiments it can be concluded that in general the new CFS-based method introduced in this study outperforms the mRMR-based approach, the best-first-CFS and genetic-algorithm-CFS methods by removing much more redundant features and still keeping the classification accuracies or even obtaining better performances.

A Comparison of Feature Selection Methods

In this section, we summarize the feature selection results described above to make a comparison between different methods. We use the general performance of intrusion detection systems as a measurement for this comparison. Since different intrusion detection systems used different feature selection methods and different classifiers with the aim of achieving the best classification results, we compared general performance of intrusion detection systems in terms of numbers of selected features and the classification accuracies of the machine learning algorithms giving the best classification results. The feature selection algorithms involved in this comparison are the SVM-wrapper, Markov Blanket, CART algorithms and the recently proposed generic feature selection (GeFS) method with 2 instances applicable in intrusion detection: the correlation-feature-selection (*GeFS_CFS*) and the minimal-redundancy-maximal-relevance (*GeFS_mRMR*) measures.

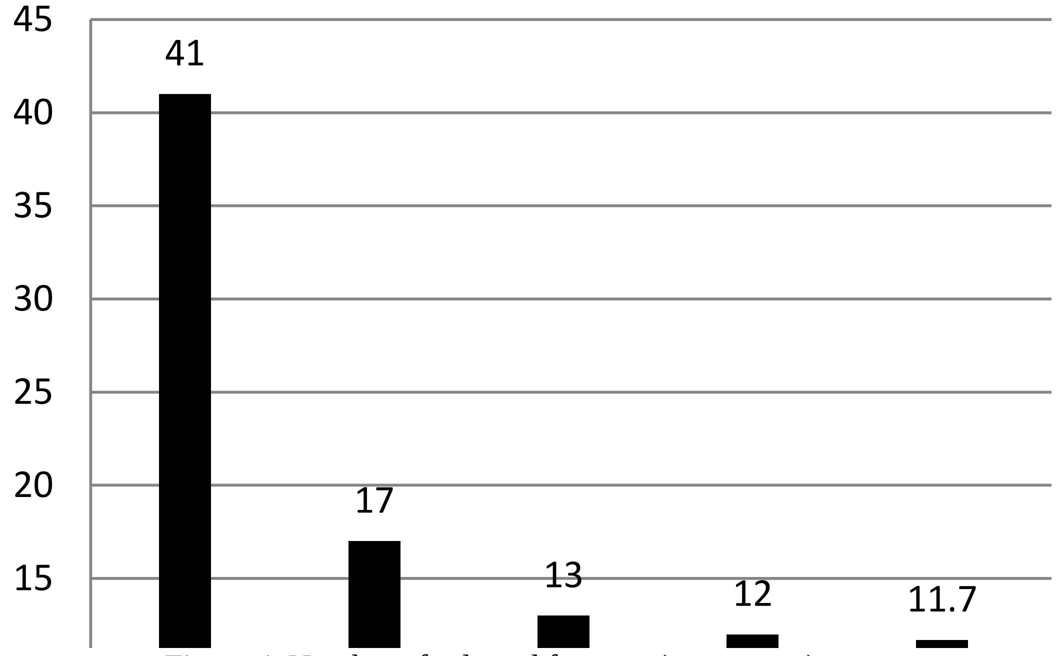

Figure 4. Number of selected features (on average)

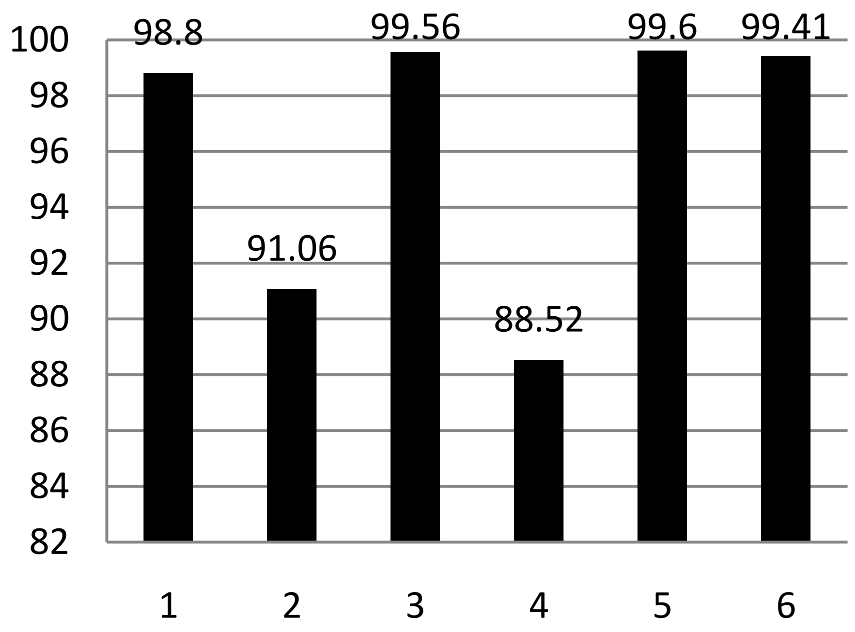

Figure 5. Classification accuracies (on average)

Figure 4 shows the average number of selected features by various feature selection methods. Figure 5 summarizes the average classification accuracies of chosen machine learning algorithms as classifiers for intrusion detection process. It can be observed from Figure 4 that the proposed generic feature selection (GeFS) method selects the smallest number of relevant features. Figure 5 shows that with also the GeFS approach, the average classification accuracies are approximately the same or even better than those achieved by applying other methods.

DISCUSSIONS AND FUTURE DIRECTIONS

McHugh (McHugh, 2000) has criticized the 1998 DARPA dataset, arguing that several procedures utilized in the simulation for generating background traffic are questionable and described superficially. There was no statistical evidence proving the similarity between the synthesized data and the real data. McHugh suspected that the 1998 DARPA dataset has statistically different characteristics from real traffic. Malhoney and Chan (Malhoney and Chan, 2003) has confirmed McHugh’s criticisms by conducting experiments, in which the 1998 DARPA data set was compared with real traffic gathered from a network environment. It is shown that many features of the network traffic appeared to be relevant for detecting attacks in the 1998 DARPA, but less important in the real traffic. For example, there are only 9 of the possible 256 TTL values, which were found in the 1998 DARPA data set. In the real traffic, Malhoney and Chan (Malhoney and Chan, 2003) observed 177 different values.

Although the 1998 DARPA data set was highly innovative for its time and it was widely utilized by IDS community. Nowadays, the data set is out of date and does not contain many recent attacks. It would be necessary to generate a new benchmarking data set for IDSs and to test again the performance of the feature construction and the feature selection algorithms on the new data set.

Regarding future research directions on feature extraction for intrusion detection, the following list shows important questions that needs to be answered:

- *Stability*: How to build feature extraction methods that are stable with slight changes in the data sets? This issue is important in analyzing polymorphic attacks, such as virus or worms.
- *Imbalance*: The distributions of normal and attack traffic in data sets in general are different. How to build feature selection algorithms which considers most frequent traffic, without neglecting less frequent traffic?
- *Adaptation*: The feature extraction process has to be adapted to changes in the data set. How to measure the adaptation and how to build algorithms which have the adaptation property?
- *Impact of feature extraction and feature selection to classification accuracy*: It is important to understand how the classification accuracy in general is affected by feature extraction.

CONCLUSION

Feature extraction is an important part of an intrusion detection system. In the present chapter, we have described the theoretical background as well as the practical applications of various feature extraction methods for intrusion detection systems. A comparison of these methods performed on the public benchmarking data sets was also provided. Moreover, we have revealed several research directions on feature extraction for intrusion detection.

ACKNOWLEDGEMENTS

We would like to thank Gjøvik University College and Norwegian Information Security Laboratory. We would also like to thank Daniel Hartung, Dang Thai Giang and Nguyen Ngoc Toan for their valuable comments.

REFERENCES

Hall, M. (1999). *Correlation Based Feature Selection for Machine Learning*. Unpublished doctoral dissertation, University of Waikato, Department of Computer Science.

Peng, H., Long, F., and Ding, C. (2005). Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 27, No. 8, pp.1226-1238.

Ghiselli, E. E. (1964). *Theory of Psychological Measurement*. New York: Mc GrawHill.

Guyon, I., Gunn, S., Nikravesh, M., & Zadeh, L. A. (2006). *Feature Extraction: Foundations and Applications.* Series Studies in Fuzziness and Soft Computing, Physica-Verlag, Springer.

Liu, H., & Motoda, H. (2008). *Computational Methods of Feature Selection.* Boca Raton : Chapman & Hall/CRC.

Chang, C-T. (2001). On the polynomial mixed 0-1 fractional programming problems, *European Journal of Operational Research, vol. 131, issue 1, pages 224-227.*

Chang, C-T. (2000). An efficient linearization approach for mixed integer problems, *European Journal of Operational Research, vol. 123, pages 652-659.*

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*, Morgan Kaufmann.

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern Classification.* USA: Wiley-Interscience.

Witten, I. H., Frank, E., Trigg, L., Hall, M., Holmes, G., and Cunningham, S. J. (1999). Weka: Practical Machine Learning Tools and Techniques with Java Implementations. In *Proceedings of the ICONIP/ANZIIS/ANNES'99 Workshop on Emerging Knowledge Engineering and Connectionist-Based Information Systems.* pp. 192–196.

Kenneth, H., Edvall, M. M., Göran, A.O. (2003). TOMLAB-for Large-Scale Robust Optimization. In *Proceedings of the Nordic MATLAB Conference*.

Bradley, P., Mangasarian, O.L., (1998). Feature selection via concave minimization and support vector machines. In *Proceedings of the Fifteenth International Conference (ICML)*, pp. 82-90.

Mangasarian, O.L., (2007). Exact 1-Norm Support Vector Machines Via Unconstrained Convex Differentiable Minimization (Special Topic on Machine Learning and Optimization). *Journal of Machine Learning Research*, 7(2):1517-1530.

Vapnik, V. (1995) *The Nature of Statistical Learning Theory*, Springer.

Cortes, C., and V. Vapnik, V., (1995). Support-Vector Networks, *Machine Learning*.

Lippmann, R. P., Fried, D., Graf, I., Haines, J., Kendall, K., Mcclung, D., Webber, D., Webster, S., Wyschograd, D., Cunninghan, R., and Zissman, M. (2000). Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation. In *Proceedings of the on DARPA Information Survivability Conference and Exposition* (DISCEX’00, Hilton Head, South Carolina, Jan. 25-27). IEEE Computer Society Press, Los Alamitos, CA, 12–26.

Lippmann, R. P., Graf, I., Garfinkel, S. L., Gorton, A. S., Kendall, K. R., McClung, D. J., Weber, D. J., Webster, S. E., Wyschogrod, D., and Zissman, M. A. (1998). The 1998 DARPA/AFRL off-line intrusion detection evaluation. Presented to *The First Intl. Workshop on Recent Advances in Intrusion Detection (RAID-98)*, (*No printed proceedings*) Lovain-la-Neuve, Belgium, 14–16 September.

Lee, W. (1999). A data mining framework for building intrusion detection Models. In *IEEE Symposium on Security and Privacy*, pages 120–132, Berkeley, California.

Lee, W. and Stolfo, S. (2000). A Framework for Constructing Features and Models for Intrusion Detection Systems. *ACM Transactions on Information and System Security*, volume 3, pp. 227-261.

Agrawal, R., Imielinski, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of*

Data (SIGMOD ’93, Washington, DC, May 26-28), P. Buneman and S. Jajodia, Eds. ACM Press, New York, NY, 207–216.

Mannila, H., Toivonen, H., and Verkamo, A. I. (1995). Discovering frequent episodes in sequences. In *Proceedings of the First International Conference on Knowledge Discovery in Databases and Data Mining* (Montreal, Canada, Aug. 20-21).

Mannila, H., and Toivonen, H. (1996). Discovering generalized episodes using minimal occurrences. In *Proceedings of the 2nd International Conference on Knowledge Discovery in Databases and Data Mining* (Portland, OR, Aug.).

Nguyen, H., Franke, K., and Petrovi'c, S. (2010a). Improving effectiveness of intrusion detection by correlation feature selection, In *Proceedings of the 2010 International Conference on Availability, Reliability and Security (ARES),* Krakow, Poland, February 2010, pp. 17-24.

Nguyen, H., Franke, K., and Petrovi'c, S. (2010b). Towards a Generic Feature-Selection Measure for Intrusion Detection. In *20th International Conference on Pattern Recognition,* Istanbul, Turkey, pp. 1529-1532.

Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules in large databases. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB, pages 487, Santiago, Chile.

Zaki, M. J. (2000). Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12(3): pp. 372-390.

Chebrolu, S., Abraham, A., Thomas, J. (2005). Feature Deduction and Ensemble Design of Intrusion Detection Systems. *Computers & Security*, Volume 4, pp. 295–307.

Sung, A.H., Mukkamala, S. (2003). Identifying Important Features for Intrusion Detection Using Support VectorMachines and Neural Networks. In *Proceedings of the International Symposium on Applications and the Internet (SAINT)*, pp. 209–217. IEEE Press, Los Alamitos.

Chen, Y., Li, Y., Cheng, X-Q., & Guo, L. (2006). Survey and Taxonomy of Feature Selection Algorithms in Intrusion Detection System. In *Proceedings of Inscrypt 2006, LNCS 4318,* (pp. 153-167).

Koller, D. and Sahami, M. (1996). Toward optimal feature selection. In *Proceedings of International Conference on Machine Learning*.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J, (1984). *Classification and Regression Trees*. Wadsworth, Belmont.

Guyon, I.,Weston, J., Barnhill, S. & Vapnik, V., (2002). Gene selection for cancer classification using support vector machines. *Machine learning,* 46(1):389-422.

Weston, J., Mukherjee, S., Chapelle, O., Pontil, M., Poggio, T. & Vapnik, V., (2001). Feature selection for SVMs. *Advances in neural information processing systems*, pp. 668-674.

Wang, K., Parekh, J., and Stolfo, S. (2006). Anagram: A content anomaly detector resistant to mimicry attack. In *Recent Adances in Intrusion Detection (RAID)*, pages 226-248.

Wang, K., and Stolfo, S. (2004). Anomalous payload-based network intrusion detection. In *Recent Adances in Intrusion Detection (RAID)*, pages 203-222.

Shannon, C.E. (1948). A Mathematical Theory of Communication", *Bell System Technical Journal*, 27, pp. 379–423 & 623–656.

Rodgers, J.L. and Nicewander, W.A. (1988). Thirteen ways to look at the correlation coefficient. *The American Statistician*, 42(1):59–66.

McHugh, J.(2000). Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA off-line intrusion detection system evaluation as performed by Lincoln Laboratory. *ACM Transactions on Information and System Security*, 3(4).

Jain A.K., Duin R.P.W., Mao, J. (2000). Statistical Pattern Recognition: A Review. *IEEE Transactions on Pattern Analysis and Machine Intelligence In Pattern Analysis and Machine Intelligence*, IEEE Transactions on, Vol. 22, No. 1., pp. 4-37.

Northcutt S. (1999). *Network Intrusion Detection: An Analyst's Handbook.* Publisher: Sams.

Mahoney, M.V., and Chan, P.K. (2003). An analysis of the 1999 DARPA/Lincoln laboratory evaluation data for network anomaly detection. *Technical Report TR CS-2003-02*, Computer Science Department, Florida Institute of Technology.

Roesch, M. (1999). Snort - Lightweight Intrusion Detection for Networks. In *Proceedings of the 13th USENIX conference on System administration*, pp 229-238.

Lunt, T., Tamaru, A., Gilham, F., Jagannathan, R., Neumann, P., Javitz, H., Valdes, A., and Garvey, T. (1992). A real-time intrusion detection expert system (IDES) - *final technical report*.

Denning, Dorothy E., (1986). An Intrusion Detection Model. In *Proceedings of the Seventh IEEE Symposium on Security and Privacy*, pages 119–131.

ADDITIONAL READING SECTION

Mitchel, T. (1997). *Machine Learning*. McGraw-Hill, Inc., New York, NY.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2001). *Introduction to Algorithms*, Second Edition. MIT Press & McGraw-Hill.

Gu, G., Fogla, P., Dagon, D., Lee, W., & Skoric, B. (2006). Towards an information-theoretic framework for analyzing intrusion detection systems. In *Proceedings of the 11th European Symposium on Research in Computer Security (ESORICS'06)*, (pp. 527-546). Springer.

Crescenzo, G. D, Ghosh, A., & Talpade, R. (2005). Towards a theory of intrusion detection. In *Proceedings of the 10th European Symposium on Research in Computer Security (ESORICS'05)*, (pp. 267-286). Springer.

Guyon, I., and Elisseeff, A. (2003). An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research*, volume 3, pp. 1157–1182.

Kohavi, R., and Sommerfield, D. (1995). Feature subset selection using the wrapper method: Overfitting and dynamic search space topology. In *Proceedings of First International Conference on Knowledge Discovery and Data Mining*, Morgan Kaufmann, pp. 192–197.

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

Nemhauser G.L., and Wolsey, L.A. (1989). Integer programming. In *Handbooks in Operations Research and Management Science*, Volume 1, Elsevier/North Holland, Amsterdam, Netherlands.

KEY TERMS & DEFINITIONS

Keywords: risk management, intrusion detection system, feature extraction, feature construction, feature selection, pattern recognition system, support vector machine.

Definition of Keyword:

Risk Management: the identification, assessment, and prioritization of risks that are the effect of uncertainty on objectives, whether positive or negative.

Intrusion Detection System: a system (a device or a software application) that monitors network and/or system activities for detecting intrusions in the current networks or computers .

Feature Extraction: the process of extracting the most compact and informative features in a given data.

Feature Construction: the process of constructing representative features from the original data.

Feature Selection: the process of selecting the most relevant feature subset from a set of features.

Pattern Recognition system: a system that assigns an output value (or label) to a given input value (or instance).

Support Vector Machine: a supervised learning technique that is about to construct a hyper-plane, which tends to separate data points in the space.