Reliable Machine Learning Algorithms for Intrusion Detection Systems

Machine Learning for Information Security and Digital Forensics

Hai Thanh Nguyen

Thesis submitted to Gjøvik University College

for the degree of Doctor of Philosophy in Information Security

2012

Reliable Machine Learning Algorithms for Intrusion Detection Systems

Faculty of Computer Science and Media Technology
Gjøvik University College

Reliable Machine Learning Algorithms for Intrusion Detection Systems: Machine Learning for Information Security and Digital Forensics / Hai Thanh Nguyen
Doctoral Dissertations at Gjøvik University College 4-2012
ISBN: 978-82-93269-07-6
ISSN: 1893-1227

This thesis is dedicated to my wife-Sunny and my parents. Without your supports and sacrifies this work would not have been possible.

Declaration of Authorship

I, Hai Thanh Nguyen, hereby declare that this dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except as declared and specified in the text.

Moreover, I also declare that it is not substantially the same as any that I have submitted, or, is being concurrently submitted for a degree, diploma, or other qualification at Gjøvik University College or any other University or similar institution except as declared in the Preface and specified in the text. I further state that no substantial part of my dissertation has already been submitted, or, is being concurrently submitted for any such degree, diploma, or other qualification at Gjøvik University College or any other University of similar institution except as declared in the Preface and specified in the text.

I also declare that I have fulfilled the residency requirements and have successfully completed the Educational Section of my doctoral studies.

I also declare that I have submitted this thesis on or before the statutory submission date or have obtained permission for a deferred submission date, and that I have formally applied for examiners to be appointed.

Signature of Candidate:

(Hai Thanh Nguyen)

Date and Place:

Preface

Intrusion Detection Systems (IDSs) are an important component of security measures protecting computer systems and networks from potential abuse. It has been more than 30 years since John Anderson published one of the earliest papers on IDSs in the Computer Security Threat Monitoring and Surveillance in 1980. Since then many different efficient approaches for IDSs have been proposed and implemented in practice. However, the research on intrusion detection is still an active field and attracts attention of many scientists because of its challenges and the IDSs' necessity for our daily life when using Internet.

Nowadays, Intrusion Detection Systems are facing a new challenge in dealing with big network data and have to operate in changing and adversarial network environments with diverse protocols, services, applications and so on. Existing approaches, such as the manual method using expert knowledge, will be inappropriate for IDSs. Machine learning, which is a field of study that gives computers the ability to learn without being explicitly programmed, is becoming more and more important for solving these challenges. This is because of the efficiency and effectiveness of the automatic learning algorithms, especially when the amount of network data is increasing rapidly. However, one of the important research questions before machine learning can be applied for IDSs in practice is about the reliability of detection results provided by automatic learning algorithms. So far previous research on intrusion detection have not studied this question well.

This disseratation aims to develop new reliable machine learning algorithms including a new reliable feature-selection measure, a new reliable ensemble-feature-selection framework, a general Lp-norm support vector machine and an optimal K-means clustering algorithm. In addition, this disseratation applies the new proposed algorithms in building various reliable intrusion detection systems, such as Web application firewalls (WAFs), network-based intrusion detection systems, botnet-malware detection systems, host-based intrusion detection systems and testing of WAFs.

The experiments and the algorithm designs were mainly conducted at Norwegian Information Security Laboratory, Gjøvik University College. Several results were obtained during long-term cooperation with the Information Security Institute (IFA) at Spanish National Research Council (CSIC) in Spain, with the Department of Computer Science at Universidad Carlos III (UC3M) in Spain, with the Center for Advanced Security Research Darmstadt (CASED) in Germany and with the Machine Learning Group at University of California at Santa Cruz in USA.

The research activities have so far resulted in five scientific journal articles, two book chapters and eleven peer-reviewed conference papers.

Summary

The principal focus of the present dissertation is to develop new machine learning methods for increasing the **reliability**, **efficiency** and **effectiveness** of intrusion detection systems. The dissertation studies (i) feature selection methods, (ii) supervised learning algorithms and (iii) un-supervised learning algorithms. Applications in intrusion detection include (1) general network-based intrusion detection systems, (2) general host-based intrusion detection systems, (3) Web application firewalls, (4) botnet-malware detection systems, and (5) testing systems of Web application firewalls.

For the new machine learning methods, we propose to reformulate (i) a class of feature selection methods, e.g. correlation-based and mutual-information-based feature selection, (ii) Lp-norm support vector machines and (iii) the K-means clustering algorithm as discrete optimization problems and propose to unify them into one framework. We prove that these algorithms can be casted into a mixed 0-1 linear programming problems (M01LP), in which the number of variables and constraints are linear in the number of the input features. The obtained M01LP is solved by means of adequate algorithms, such as the branch and bound algorithm or the D.C. (Difference of Convex Functions) programming approach. The new formulation of machine learning algorithms allows to (a) realize the same representation of many different algorithms, (b) easily combine these algorithms to study their reliability including their optimality, generalization, consistency and robustness and (c) optimize the feature selection process and learning model selection process.

For the applications in intrusion detection systems, we conduct experiments on five different datasets: KDD CUP 1999, UNM audit dataset, CSIC 2010 HTTP dataset, ECML-PKDD 2007 HTTP dataset, and Botnet Malware. The experimental results show that our new proposed approaches (a) decrease the computational efforts due to optimal learning algorithms and optimal feature selection, (b) increase the reliability including the generalization and robustness and (c) increase the efficiency and effectiveness of network-based intrusion detection systems, host-based intrusion detection systems, Web application firewalls, botnet-malware detection systems and testing systems of Web application firewalls.

Sammendrag

Doktorgradsoppgaven fokuserer på utvikling av maskinlæringsmetoder for forbedring av pålitelighet og effektivitet av inntrengingsdeteksjonssystemer (IDS) for datasystemer. Oppgaven studerer (i) egenskaps seleksjonsmetoder (ii) veiledede læringsalgoritmer og (iii) ikke-veiledede læringsalgoritmer. Anvendelser av IDSer inkluderer (1) generelle nettverk-baserte IDSer, (2) generelle vert-baserte IDSer, (3) Web-applikasjonsbrannmurer, (4) botnet skadevareoppdagelsessystemer, (5) ondsinnedee pdf–fil oppdagelsessystemer og (6) systemer for testing av Web-applikasjonsbrannmurer.

For nye maskinlæringsmetodene foreslår vi reformulering av (i) en klasse av egenskaps eleksjonsmetoder, for eksempel korrelasjonsbaserte og felles informasjonsbaserte egenskape seleksjon, (ii) Lp-norm støttevektormaskiner og (iii) K-gjennomsnittsklynger til diskrete optimaliseringsproblemer. Vi foreslår også et felles rammeverk for å formulere disse problemene. Videre beviser vi at disse algoritmene kan konverteres til blandede 0-1 lineærprogrammeringsproblemer (M01LP), hvor antall variabler og betingelser er lineære i antall inputegenskaper. Anskaffede M01LP problemet løses med passende algoritmer, for eksempel "Branch-and-Bound" eller "Difference of Convex Functions" (D.C.) programmeringstilnærminger. Denne reformuleringen av maskinlæringsalgoritmer muliggjør (a) felles representasjon av mange forskjellige algoritmer, (b) enkel kombinasjon av disse algoritmene for å studere deres pålitelighet (inkluderer deres optimalisering, generalisering, konsistens og robusthet) og (c) optimalisering av egenskapsseleksjonsprosessen og læringsmodellseleksjonsprosessen.

Som eksempel på anvendelse i inntrengingsdeteksjons prosessen, gjennomførte vi eksperimenter med 6 forskjellige datasett: KDD CUP 1999, UNM revisjons datasettet, CSIC 2010 http datasettet, ECML-PKDD 2007 http datasettet, botnet skadevare og ondsinnede pdf filer. De eksperimentelle resultatene viser at vår nye tilnærming (a) reduserer behovet for regnekraft på grunn av bruken av optimalisertee læringsalgoritmer og egenskapsseleksjonsmetoder, (b) forbedrer påliteligheten, inkludert generalisering og robusthet og (c) forbedrer effektivitet av nettverk-baserte IDSer, vert-baserte IDSer, Web applikasjonsbrannmurer, botnet skadevaredeteksjonssystemer, ondsinnede pdf-fil oppdagelsessystemer og systemer for testing av Web-applikasjonsbrannmurer.

Acknowledgments

I gratefully acknowledge my advisors Prof. Dr. Katrin Franke and Prof. Dr. Slobodan Petrović for their guidance and help throughout my PhD. My deepest gratitude to all of you for being my mentors during the last four years.

In addition, I acknowledge the other HiG faculty and staffs who have taught and supported me. In particular, I would like to thank Prof. Dr. Patrick Bours, Dean Morten Irgens, Prof. Dr. Christoph Busch, Kathrine Huke Markengbakken, Hilde Bakke, Associate Prof. Dr. Nils Kalstad Svendsen, Jayson David Mackie, Dr. Knut Wold, Jan Kåre Testad, Aneta Laskowska, Stein Runa Olsen, Ingrid von Schantz Bakka.

My academic experience has been greatly enriched by all my collaborators and friends: Carmen Torrano Giménez, Prof. Dr. Gonzalo Álvarez Marañón, Sergio Pastrana Portillo, Associate Prof. Agustín Orfila, Prof. Dr. Manfred K. Warmuth, Prof. Dr. Harald Baier, Dr. Daniel Hartung, Dr. Giang Dang Thai, Dr. Bian Yang, Dr. Mark Seeger, Sukalpa Chanda, Lisa Rajbhandari, Mohammad Derawi, Takashi Watanabe.

I would also like to thank my other thesis-committee members, Prof. Dr. Klaus-Robert Müller, Prof. Dr. Milan M. Milosavljević and Prof. Dr. Ivar Farup, for their acceptance to review my work and Prof. Dr. Stephen D. Wolthusen for being the head of the committee.

I have benefited greatly from financial supports of the Norweigian Information Security Laboratory (NISLAB), Gjøvik University College, Norway throughout the past of four years. I would also like to express my appreciation to the Center for Advanced Security Research Darmstadt (CASED), Germany, the Information Security Institute (IFA) at Spanish National Research Council (CSIC), Spain and the Computer Science Department at University of California at Santa Cruz (UCSC), USA for their administrative and technical supports during my internships. I would like to thank The Research Council of Norway for the financial support of the IS-BILAT project.

Finally, I am indedted to my wife, my parents and brother for their enduring support. Without whose love and great supports, this Ph.D would have been meaningless.

Contents

List of Figures

List of Tables

List of Theorems

List of Definitions

List of Algorithms

Chapter 1

Introduction

Simplicity is prerequisite for Reliability.

EDSGER DIJKSTRA

In this chapter, Section 1.1 first introduces an overview on perspectives of machine learning techniques to build intrusion detection systems. This section then provides a detailed discussion on reliability, efficiency and effectiveness issues of these systems. Research questions and motivations to develop new computational machine learning algorithms are described in Section 1.2. Section 1.3 summarizes the contributions of this work. Finally, Section 1.4 presents the structure of the dissertation.

1.1 Motivation and General Considerations

Intrusion Detection Systems (IDSs) have become an important security tool for managing risk and an indispensable part of overall security architecture [105]. An IDS 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. Examples of IDSs are general network intrusion detection systems, Web application firewalls, botnet-malware detection systems, and so on.

The two main intrusion detection approaches are misuse detection and anomaly detection [43]. Misuse detection systems, for instance, SNORT [116], 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 to 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 [78], can overcome the shortcoming of the misuse detection systems. An anomaly detector assumes that normal behavior are different from abnormal behavior. Therefore, abnormal activities can be detected by looking at normal activities only. In fact, in these systems, 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 potential of detecting novel attacks, it is difficult to produce effective models of normal patterns by hand and these systems tend to generate more false positive alerts than the misuse detection systems.

To cope with these problems, the intrusion detection task has been formulated as a statistical pattern recognition task (see, for example, [46, 53, 144]) and machine learning is the core to build these systems due to its efficiency and effectiveness (see Figure 1.1 for the model of a statistical pattern recognition sytem). According to Arthur Samuel [122], machine learning is a field of study that gives computers the ability to learn without being explicitly programmed. In [90], Tom Mitchel provides another definition of machine learning saying that a computer program is said to learn from experience E with respect

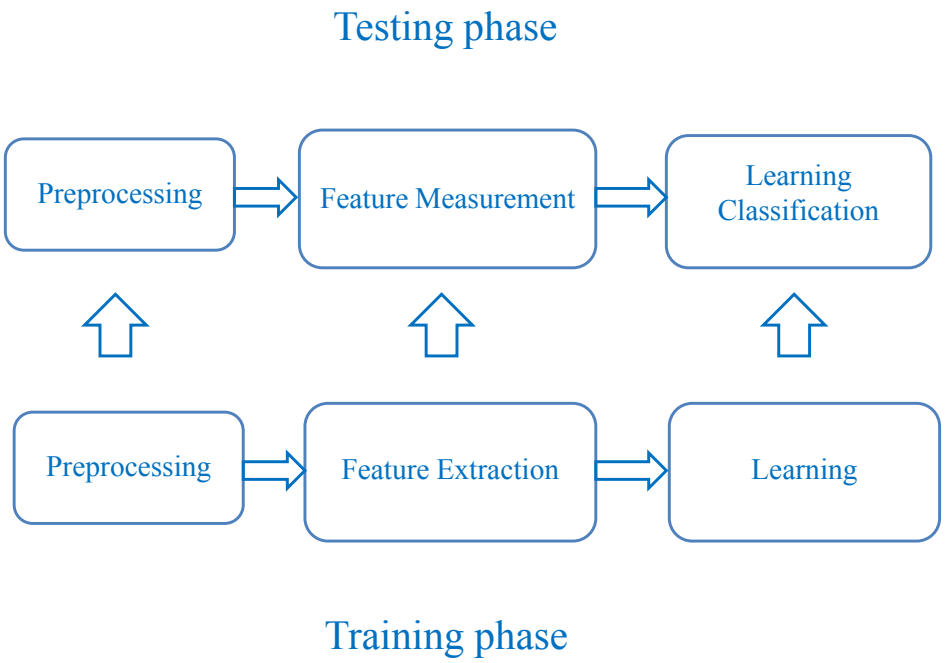


Figure 1.1: Model for statistical pattern recognition [62]

to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E. By means of this approach, an intrusion detection system learns and models the normal and abnormal behavior from a given dataset, which is the experience E. The IDS then uses the gained model to classify new patterns. Figure 1.1 shows the formal model of the IDS, which is essentially a statistical pattern recognition system. The IDS consists of two phases: training and testing. In the training phase, several models learn or, in other words, are built to differentiate normal and abnormal behavior in the given dataset E. The performance of the built models is measured by determining the classification accuracy P. In the testing phase, the best model performed on the testing dataset is selected as a potential intrusion detection system.

In more detail from Figure 1.1, the test and training patterns as raw data are normalized, noise as well as unwanted data are removed by the pre-processing modules. In the training phase, to have an efficient and precise modeling of the normal and abnormal behavior, an automatic process named feature extraction/selection first looks for a representative attribute set from the training patterns. There exist many different ways of selecting attributes by considering the relationships between them, such as correlation-based or mutual information-based methods. Second, depending on whether the given training dataset has the true labels or not, to model the behavior there are two different approaches: supervised and unsupervised learning models. With supervised learning algorithms, such as neural networks or support vector machines, the true labels of the training dataset are available for learning process. In some cases, the unsupervised learning algorithms, such as the K-means clustering, can still learn the normal and abnormal behavior on the dataset without their true labels. We shall cover these methods in more detail in the next chapter. In the classification phase, the built model or trained classifier is applied to assign the test pattern to one of the pattern classes under consideration of the selected attributes from the

training phase.

Due to the **efficiency** of the automatic learning techniques, the machine-learning-based intrusion detection systems (ML-IDSs) allow to detect quickly the attacks while demanding much less manual work. Because of this reason, the approach is becoming more and more important for computer security [82], especially when the huge amount of network data that needs to be analyzed by intrusion detection systems is increasing rapidly. Moreover, the ML-IDSs have demonstrated to be more **effective** in terms of classification accuracy than domain experts and other existing IDS approaches as shown in previous works (see, for example, [144]).

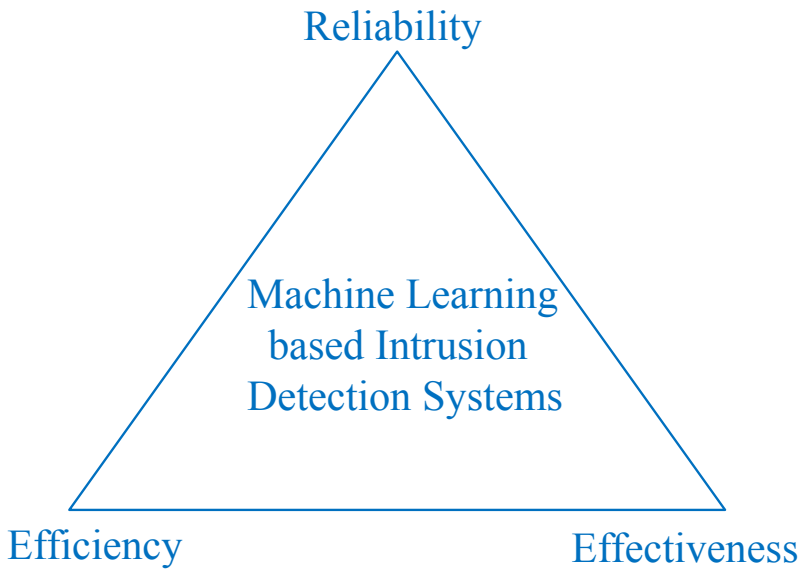


Figure 1.2: The Reliability, Efficiency and Effectivess Principle

However, the ML-IDSs are mostly not being used in practice for information security systems. One of the controversies of this is about the **reliability** issue of intrusion detection results when applying automatic machine learning methods [125]. The most common approach in practice is still the misuse IDSs, which is a domain-experts-based and a highly-reliable approach. Although, they are much less efficient and effective in detecting novel attacks than the ML-IDSs. Therefore, increasing the overall reliability, efficiency and effectiveness of ML-IDSs is important and critical task. See Figure 1.2 for the **REE (Reliability, Efficiency and Effectiveness) principle** of a ML-IDS.

Several works attempt to provide formal estimations and methodologies of reliable classifications with machine learning (see, for example, [70]). However, it is still difficult to define what exactly the reliablity is. This dissertation aims to analyze and get a better understanding of the reliability of a pattern recognition process in general. From this, we develop new reliable machine learning algorithms. We apply our new algorithms to build reliable intrusion detection systems.

From Figure 1.1, it can be seen that the reliability of a ML-IDS is first involved in the training phase, particularly in (1) feature extraction/selection and (2) in learning steps. When the ML-IDS operates in a changing or an adversarial network environment, (3) the reliable detection results in the presence of potential attacks targeted to it are also important.

Ad. 1 Reliability in feature extraction/selection includes the **appropriateness, consistency, steadiness and optimality**. First, a feature extraction/selection method needs to be appropriately chosen for a particular dataset. For example, within a dataset that has many features linearly correlated to each other, a correlation-based feature selection method [59] should be applied. Second, the feature selection results should be consistent in each time the algorithm is run. This requirement depends strongly on the search methods that we use to select features. For example, the genetic algorithm using to search for the best feature subsets is not consistent with different initial populations, thus providing different feature sets or unreliable feature selection results. The exhaustive search is a completely reliable method as it finds the best feature subset from scanning all possibilities. However, it becomes inefficient when the number of features is large. Therefore, a near optimal search strategy, such as branch and bound algorithm or D.C. programming, is a good candidate for reliable feature selection. We will show this in the next chapters. Third, a feature selection method should provide as much steady performance of a classifier as possible. That means the difference between the detection rates before and after selecting features should not be large. We will formalize these requirements and provide new reliable feature selection methods in the Section 3.1 and the Section 3.2.

Ad. 2 Reliability in learning includes the **appropriateness, generalization and optimality**. First, a learning algorithm to model normal and attack behavior needs to be appropriately applied. For example, given a training dataset without labels, an unsupervised learning algorithm, such as K-means [63], should be used. Second, a machine-learning-based intrusion detection system should have a generalization ability to detect future attacks. This can be done through simple models with a good performance on training data to avoid the over-fitting phenomenon, in which the model is fit perfectly to the training dataset, but cannot detect well the future pattern with slight differences from the training data. Third, a learning process is normally to solve an optimization problem. Therefore, an optimal search strategy, such as branch and bound or langrangian methods, is necessary. We will provide new reliable learning algorithms in the Section 3.3 and the Section 3.4.

Ad. 3 Reliability of a machine-learning-based intrusion detection system in a changing or an adversarial network environment includes the **adaptability** and the **robustness**. First, the ML-IDS is required to be adaptive when the type of attacks is changing all the time. This can be solved by running several ML-IDSs at the same time to detect attacks. However, the difficulty lays on combining their outputs. We will discuss about several combination approaches, such as ensemble learning and online learning, in Section 5.2.1. Second, in the presence of training data noise (including label and feature noise), the ML-IDS might learn wrong patterns, thus providing unreliable classification results. Therefore, the robustness of ML-IDSs to ignore the data noise in the training phase is necessary. We will provide several research directions, such as the use of non-convex loss functions instead of convex ones, in the Section 5.2.2.

1.2 Objectives of the Dissertation

The goal of the dissertation is to answer the following research questions:

1. How to **generalize, optimize various feature-selection methods and select an appropriate method** for a particular case in such an efficient way that provides reliable features for effective intrusion detection?

2. How to **combine feature-selection methods** in such an efficient way that provides reliable features for effective intrusion detection?

3. How to **generalize and optimize existing supervised machine-learning algorithms** in such an efficient way that provides reliable and effective intrusion detection systems?

4. How to **generalize and optimize the existing unsupervised machine-learning algorithms** in such an efficient way that provides reliable and effective intrusion detection systems?

5. How to **integrate expert knowledge** into automatic feature selection and learning processes in such an efficient way that provides much more reliable and effective intrusion detection results?

6. How to **apply new methodologies to different intrusion detection systems**, such as general intrusion detection, Web application firewalls, testing of Web application firewalls, and botnet-malware detection systems ?

1.3 Contributions of the Dissertation

The contributions of the dissertation are visualized in Figure 1.3 and summarized as follows:

1. **Generalization of feature-selection methods-New reliable feature selection process:** In this dissertation, we first analyze the main factors that affect the reliability in the feature-selection process: the choice of feature-selection methods and the search strategies for relevant features. We introduce a formal definition of a reliable feature-selection process. The definition provides formal measurements of reliability in feature-selection, i.e., the **steadiness** of a classifier's performance and the **consistency** in search for relevant features. Second, we propose new methods to address the main causes of unreliable feature-selection process. In particular, we introduce a new methodology of determining appropriate instances from a class of feature-selection methods. We call this class a generic-feature-selection (GeFS) measure. We also propose a new search approach that ensures the globally optimal feature subset by means of the GeFS measure. The new search approach is based on solving a mixed 0-1 linear programming (M01LP) problem by means of the branch-and-bound algorithm. In this M01LP problem, the number of constraints and variables is linear in the number of full set features. Finally, we validate our new proposed methods by applying the GeFS measure to intrusion detection systems. This contribution has been presented in [93, 94, 98].

2. **Combination of feature-selection methods-New ensemble feature selection framework:** This dissertation studies a phenomenon in feature selection process that affects the reliability of feature selection results: the *over-selecting* phenomenon. A feature selection method is required to be general enough to find representative features from training data, which are then used for classifying test patterns. The situation when the features selected from the training data are quite different from the representative features of the testing data is called *over-selecting*. The main causes of the *over-selecting* phenomenon are: *(i)* non-comprehensive consideration of statistical properties of the training data, *(ii)* heuristic search strategies for feature selection and *(iii)* small sample size of the data set for training. In this dissertation, we show the influence of the *over-selecting* phenomenon on the *over-fitting* phenomenon of machine learning algorithms. We propose a new framework to address principal causes of *over-selecting*, thus reducing the chance of *over-fitting* and providing reliable results. Our new framework that we call ensemble-feature-selection measure (EnFS), allows the consideration of many statistical properties of a given data set at the same time by combining many feature selection methods used in the filter model. From the chosen

feature selection measures, a new combined measure is constructed. We also propose a new search algorithm that ensures the globally optimal feature subsets by means of the constructed measure. Similar to the case of the generic-feature-selection measure, this new search approach is based on solving a mixed 0-1 linear programming (M01LP) problem by means of the branch-and-bound algorithm. In order to evaluate the quality of our EnFS measure, we chose the design of an intrusion detection system (IDS) as a possible application. Experimental results obtained over the KDD CUP'99 benchmarking data set for IDS [73, 72] show that our EnFS measure is capable of reducing *over-fitting* by addressing *over-selecting*, thus providing more reliable feature selection results. This contribution has been presented in [96].

3. **Supervised machine learning-New general Lp-norm Support Vector Machines:** This dissertation analyzes the popular Lp-norm Support Vector Machines (Lp-SVMs with $p = 1$ or $p = 2$) algorithms to deal with small datasets. When the training dataset is small, the distribution of the target variable, which the model is trying to predict, is likely to be changed in the testing data, thus leading to over-fitting phenomenon and unreliable classification results. In this case, it is worth selecting few important features for building compact and simple models to reduce the chance of over-fitting in the future. The filter model for feature selection, such as the GeFS and EnFS measures from previous contributions, might not be a good choice, since there are not enough samples to estimate statistical properties of the dataset. The wrapper and embedded models use the performance of a machine learning classifier, which is a good criteria to select important features and at the same time to keep the model as accurate as possible. In this dissertation, we focus on the embedded feature selection model that is based on the Lp-norm support vector machines (Lp-SVMs) to cope with small datasets. We realize that the Lp-SVMs do not comprehensively remove irrelevant and redundant features, because the Lp-SVMs consider n full-set features be important for training while skipping other $2^n - 1$ possible feature subsets at the same time. We propose to generalize the Lp-SVMs into a new general Lp-norm Support Vector Machine (GLp-SVM) that takes into account all 2^n possible feature subsets. We represent the GLp-SVM as a mixed 0-1 programming problem (M01LP). We prove that solving the new proposed M01LP optimization problem results in a smaller error penalty and enlarges the margin between two support vector hyper-planes, thus possibly giving more reliable classification results and a better generalization capability of SVMs than solving the traditional Lp-SVMs. Moreover, by following the new general formulation we can easily integrate expert knowledge into the GLp-SVMs by adding the constraints $x_1 + x_2 + .. + x_n = T, x_i = 1$, where T is the pre-defined number of selected features and x_i is the pre-defined important feature, to the proposed M01LP optimization problem. In order to reduce the computational complexity of directly solving the M01LP problem, we propose to equivalently transform it into a mixed 0-1 linear programming (M01LP) problem if $p = 1$ or into a mixed 0-1 quadratic programming (M01QP) problem if $p = 2$. The M01LP and M01QP problems can then be solved by using the branch and bound algorithm. Experimental results obtained over the UNM and MIT Lincoln Lab benchmark datasets for host-based intrusion detection systems show that our new proposed GLp-SVM outperforms the traditional Lp-SVMs by improving the classification accuracy by more than 7.48%. This contribution has been presented in [97, 100].

4. **Unsupervised machine learning-New optimization approach for K-means clustering** It has been shown that the K-means clustering problem is an $NP-$hard optimization problem, even if K is fixed to 2. That means there are no algorithms running in polynomial time to find globally optimal K-means clustering. Many heuristic approaches were proposed and applied to different application domains. However, the question on how to efficiently find more accurate and more reliable K-means clustering for large-scale data are still open. In this dissertation, we propose a new search

method for optimal K-means with $K = 2$. The main ideas are to cast the K-means problem into a mixed 0-1 linear programming problem which can be solved by using the D.C. (Difference of convex functions) programming approach. This contribution has been present in [92].

5. **Expert knowledge-New way of integration of expert knowledge** This dissertation provides a new efficient way of integrating expert knowledge to the automatic feature selection and learning processes. It can easily be done by assigning values to binary variables in the mixed 0-1 linear programming problems, which are reformulated forms of feature selection and learning problems, to indicate the importance of features. Our approach is a combination of automatic machine learning methods and expert knowledge, thus the obtained intrusion detection results are much more reliable. This contribution has been presented in [94, 95, 100].

6. **Application of new algorithms-Applications to information security and digital forensics problems** We apply successfully the new proposed methods to enhance the reliability, efficiency and effectiveness of the general network intrusion detection systems, Web application firewalls, botnet-malware detection systems, and tesing of Web application firewalls. This contribution has been presented in [26, 94, 95, 96, 97, 98, 99, 100, 101, 102, 106].

In summary, this dissertation combines five journal articles, two book chapters and eleven peer-reviewed conference articles in a partially rewritten format.

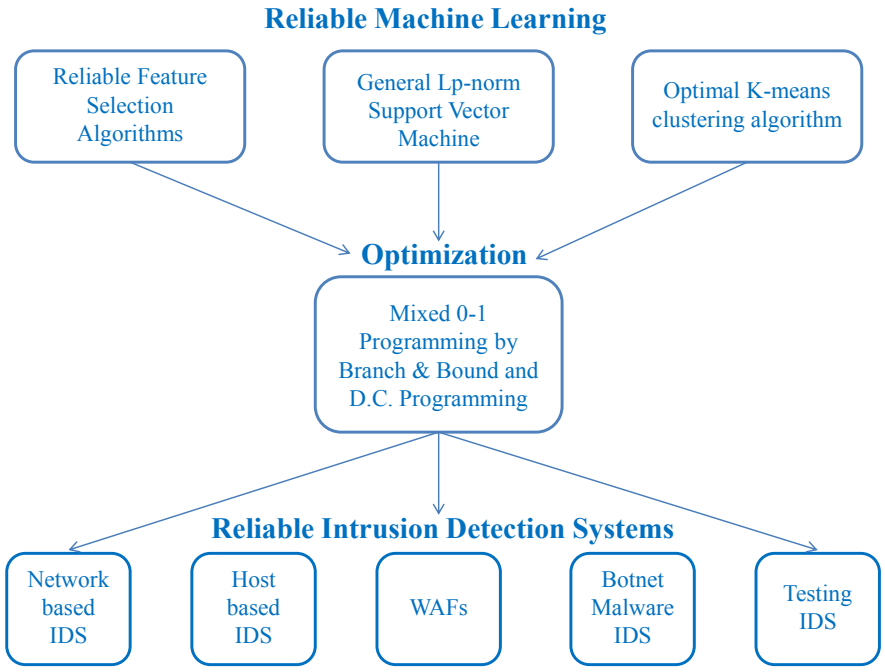


Figure 1.3: Contributions of the dissertation

1.4 Structure of the Dissertation

The structure of the dissertation is organized as follows:

1. **Introduction** This general introduction is followed by an overview on perspectives of machine learning techniques to build intrusion detection systems. A detailed discussion on reliability, efficiency and effectiveness issues of machine-learning-based intrusion detection systems is given. Research questions and motivations to develop new computational machine learning algorithms are described.

2. **Theoretical Background** This chapter discribes basic machine learning techniques in more detail: feature selection, supervised and unsupervised algorithms. Two popular operational research approaches for machine learning are provided: convex and non-convex optimization techniques.

3. **New Reliable Machine Learning Algorithms** The new proposed algorithms are about feature selection, supervised learning and unsupervised learning. In particular, this chapter introduces a new reliable feature-selection process, a new ensemble feature-selection framework, general Lp-norm support vector machines, and optimal K-means clustering via D.C. (Difference of Convex Functions) Programming.

   Major contributions have been published in five international journal articles, two book chapters and eleven peer-reviewed conference and workshop articles.

4. **Applications to Intrusion Detection Systems** This part describes successful applications of new developed machine learning algorithms to enhance the reliability, efficiency and effectiveness of network-based intrusion detection systems, host-based intrusion detection systems, Web application firewalls, botnet-malware detection systems, and testing systems for Web application firewalls.

5. **Conclusions** The theoretical and empirical findings are discussed in detail: new reliable machine learning methods via an operational research approach and their applications to security and forensics. Future works on the reliability of machine-learning-based intrusion detection systems in changing and adversarial network environments are provided.

In summary, the dissertation introduces (1) new reliable machine-learning algorithms including (i) a new reliable feature-selection process, (ii) a new reliable ensemble feature-selection framework, (iii) a general Lp-norm support vector machines and (iv) an optimal K-means clustering algorithm. Moreover, the dissertation shows (2) the sucessful applications of the new algorithms to intrusion detection systems. Finally, the dissertation provides (3) several research directions on reliability of machine-learning-based intrusion detection systems in a changing or an adversarial network environments.

Chapter 2

Theoretical Background

Nothing happends in the universe that does not have a sense of either certain maximum or minimum.

LEONHARD EULER

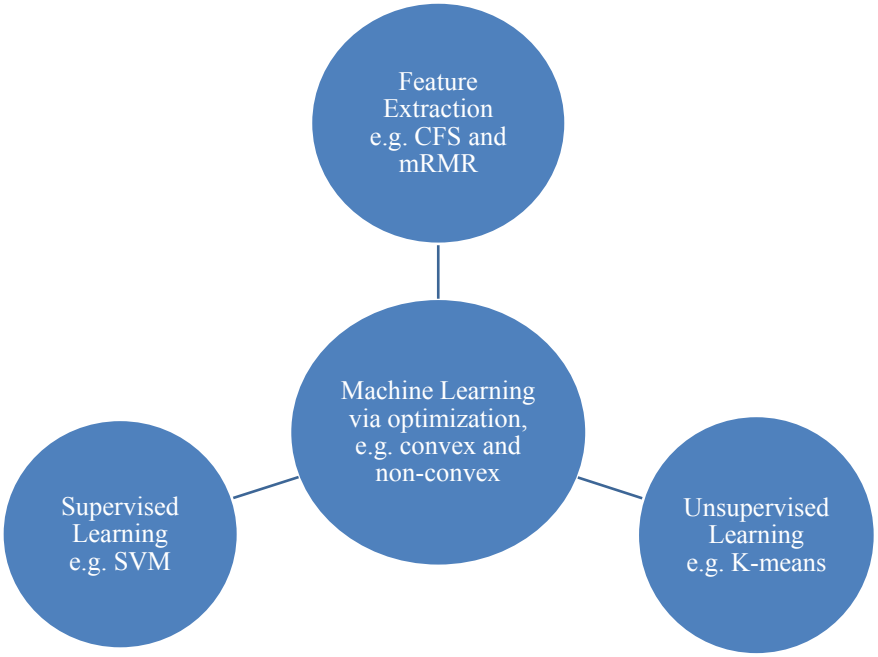


Figure 2.1: Machine learning algorithms addressed in the dissertation

This chapter provides a background on various machine learning algorithms, which is integral to and motivates this dissertation. First, Section 2.1 describes three main groups of machine learning techniques in more detail: feature extraction/selection, supervised learning and unsupervised algorithms with concrete illustrations. In fact, we introduce various feature extraction methods, such as correlation feature selection, the support vector machines from supervised learning and the k-means from unsupervised learning. We also analyze their disadvantages to motivate our research in the next chapter. Finally, Section 2.2 presents two popular operational research approaches for machine learning: convex and non-convex optimization techniques. The content of this chapter is visualized in Figure 2.1.

2.1 Machine Learning

2.1.1 Feature Construction

A feature, which is a synonym for input variable or attribute, is any representative information that is constructed from the raw data set. A special pattern, which is directly selected from the data set, can be considered a feature, for example, a representative substring of SQL injection code in an HTTP request. The distributions of characters or groups of characters are features. In some cases, the structures or semantics of data sets are considered 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 to be constructed from the data. The process of determining the most compact and informative features of a given data set is called Feature Construction. By means of this process not only the efficiency of data storage is improved, but also the processing performance of a statistical pattern recognition system, such as an intrusion detection system, is increased.

A feature construction algorithm consists of two steps: feature extraction and feature selection. Feature extraction 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 extract 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 extraction, several approaches, such as n-grams, association rule learning and frequent episode extraction, are usually applied. These methods will be introduced in more detail in the next sections.

At the feature extraction stage, one should bear in mind that no information should be lost from the original data set. A common idea is to take into account all the possible informative features. However, adding more features seems to come at a price: it increases the dimensionality of the data that is considered, thus increasing the complexity of a pattern recognition system. Moreover, the irrelevant and redundant features are possibly contained in the set of features. The main focus of this section is to determine methodologies for deciding whether or not a feature is relevant. We call the methodologies feature selection methods. In general, feature selection can provide the following benefits [56]:

- 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. The automatic feature selection methods, which include filter, wrapper and embedded models from machine learning, will be emphasized later in this section.

2.1.1.1 Feature Extraction

In this section, we present feature extraction methods for intrusion detection systems that include 1) association rule learning, 2) frequent episode extraction and 3) n-grams extraction.

Association Rule Learning: Association rule learning [12] 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 a base 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 (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 to 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 if a normal user attempts to carry out 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 dataset. Each record r_i contains a subset of features in I. A rule is defined as an implication of the form:

$$X \Rightarrow Y, \, where \, X, Y \subseteq I, X \cap Y = 0$$

When interpreting the above example in intrusion detection according to this definition, the following 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: First, 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. Second, 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 [13] and Eclat [150] algorithms.

Frequent Episode Extraction: Frequent episodes [84, 85] are normally utilized for representing sequential audit data. In fact, frequent episodes are collections of events occurring frequently together. For example, in the sequence of Figure 2.2, the episode "E is followed by F" occurs several times:

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

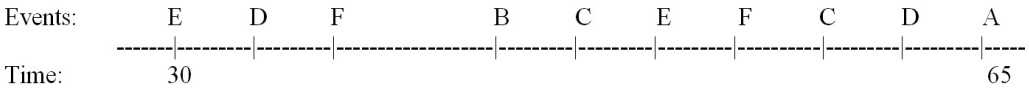


Figure 2.2: A sequence of events

as statistical nature of many attacks and normal users' behavior. From that, additional features will be extracted for detecting incoming traffic.

Let $I = \{i_1, i_2, .., i_n\}$ be a set of n features called items of system audit data. Let $D = \{r_1, r_2, .., r_m\}$ be a set of records in this dataset. Each record r_i contains a subset of feature 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 to 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 [114, 136, 137]. 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:

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

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 $\mathrm{n-grams_i}$ in p.

2.1.1.2 Feature Selection

This section presents the main idea of automatic feature selection methods for intrusion detection systems that include the wrapper model, the filter model and the embedded model from machine learning [56, 74].

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. Later in this dissertation, one of the most popular machine learning algorithms, which are usually applied in the wrapper model -Support Vector Machine [133], 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 [56]. 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 [56], in the following, two measures are considered: the correlation feature selection (CFS) measure [59] and the minimal-redundancy-maximal-relevance (mRMR) measure [108]. It will be shown, in the next chapter, that these two measures can be fused and generalized into a generic feature selection (GeFS) measure for intrusion detection and it will also 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 the 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 [45], in which at each branching node, a feature has to be selected. Another example of the embedded model are SVM-based feature selection methods [57, 140], 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.

Correlation based Feature Selection: The Correlation Feature Selection (CFS) measure proposed by Hall in 1999 [59] 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"*. This hypothesis gives rise to two concepts. One is the feature-classification (r_{cf_i}) correlation and another is the feature-feature ($r_{f_if_j}$) correlation. 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 $r_{f_if_j}$ is, as the very name says, the correlation between two features f_i, f_j. The following equation from [52] used in [58] gives the merit of a feature subset S consisting of k features:

$$\mathrm{Merit_S(k)} = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}}. \tag{2.1}$$

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

$$\overline{r_{cf}} = \frac{r_{cf_1} + r_{cf_2} + \ldots + r_{cf_k}}{k}$$

$$\overline{r_{ff}} = \frac{r_{f_1f_2} + r_{f_1f_3} + \ldots + r_{f_kf_1}}{\frac{k(k-1)}{2}}$$

Therefore, we can rewrite (2.1) as follows:

$$\mathrm{Merit_S(k)} = \frac{r_{cf_1} + r_{cf_2} + \ldots + r_{cf_k}}{\sqrt{k + 2(r_{f_1f_2} + r_{f_1f_3} + \ldots + r_{f_kf_1})}}. \tag{2.2}$$

In fact, the equation (2.1) 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-correlation among them, together with the magnitude of the correlation between the features and the target variable C. From the equation (2.1), the following conclusions can be drawn: The higher the correlation values between the features of the subset S and the target variable C, the higher the correlation between the feature subset S and the target variable C; The lower the correlation between the features in the subset S, the higher the correlation between the feature subset S and the target variable C.

The task of feature selection by means of the CFS measure is as follows: Suppose that there are n full set features. We need to find the subset S of k features, which has the maximum value of $\mathrm{Merit_S(k)}$ over all 2^n possible feature subsets:

$$\max_S \{\mathrm{Merit_S(k)}, 1 \leq k \leq n\}. \tag{2.3}$$

When the number of features n is small, we apply the brute force method to scan all these subsets. But when this number becomes large, the heuristic and random search strategies, such as the best first search or genetic algorithm, are usually chosen due to their computational efficiency. Consequently, the given results will always be approximate. It is desirable to get optimal subsets of features. In the next chapter, we propose a new method to find these optimal subsets.

In order to apply (2.1) to estimate the merit of a feature subset, it is necessary to calculate the correlation between features. Following is the detail of the correlation calculation for the CFS measure

For discrete class problems when the C variable is discrete, the CFS first discretizes numeric features using the technique of Fayyad and Irani [47]. The correlation of two features X and Y is defined as symmetrical uncertainty $SU(X,Y)$ coefficient by the formula:

$$SU(X,Y) = 2[\frac{H(X) - H(X|Y)}{H(X) + H(Y)}],$$

where $H(X)$, $H(Y)$ are the entropies of variables X and Y, respectively; $H(X|Y)$ is the conditional entropy.

For continuous class problems, the measure for estimating the correlation between random variables is standard linear Pearson's correlation. The formula for correlation when the two features X and Y are both continuous is given below:

$$r_{XY} = \frac{\sum xy}{n\sigma_X\sigma_Y},$$

where σ_X and σ_Y are standard deviations of variables X and Y, respectively.

When the feature X is continuous and the feature Y is discrete, the Pearson's correlation is computed as follows:

$$r_{XY} = \sum_{i=1}^{k} p(X = x_i) r_{X_{b_i}Y},$$

where k is the number of discrete values of X, X_{b_i} is a binary variable that takes value 1 when X has value x_i and 0 otherwise.

When both variables involved are discrete, binary variables are created for both and the correlation is computed as follows:

$$r_{XY} = \sum_{i=1}^{k} \sum_{j=1}^{l} p(X = x_i, Y = y_j) r_{X_{b_i} Y_{b_j}},$$

where k and l are the numbers of discrete values of X and Y, respectively.

Beside the correlation-based feature selection measure, which considers the linear relation between features, there exists nonlinear feature selection method that is based on mutual information from information theory. In the following, we introduce the mutual information based feature selection in more detail.

Mutual Information based Feature Selection By means of mutual information, the feature selection is to find a feature subset S with k features from the full-set of n features, which jointly have the largest dependency on the target class C. This scheme, called Max-Dependency, has the following form [108]:

$$\max_{S_k} \{I(S_k, C)\},$$

where $I(S_k, C)$ is the mutual information between the feature subset $S_k = \{x_i, i = 1...k\}$ and the target class C. The $I(S_k, C)$ value is calculated as follows:

$$I(S_k, C) = \int \int p(S_k, C) \log \frac{p(S_k, C)}{p(S_k)p(C)} dS_k dC$$

$$I(S_k, C) = \int \int p(x_1, ..., x_k, C) \log \frac{p(x_1, ..., x_k, C)}{p(x_1, ..., x_k)p(C)} dx_1 ... dx_k dC$$

Although the Max-Dependency feature selection scheme has the theoretical value and can be applied when the number of features is small. It is difficult to obtain an accurate estimation for multivariate densities $p(x_1, ..., x_k)$ and $p(x_1, ..., x_k, C)$ because of two reasons in the high-dimensional space: 1) number of samples is often not large enough and 2) the multivariate density estimation often involves computing the inverse of the high-dimensional covariance matrix, which is usually an ill-posed problem [131]. Therefore, in 2005 Peng et. al. [108] proposed a heuristic approach, called Minimal-Redundancy-Maximal-Relevance (mRMR), to the optimal Max-Dependency. The main idea of Peng's method is to select features based on maximal relevance, which approximates the $I(S_k, C)$ with the mean value of all mutual information values between individual feature x_i and the target class C.

$$\max D(S_k, C) = \frac{1}{k} \sum_{i=1}^{k} I(x_i, C)$$

The features selected by means of the maximal-relevance can have a rich redundancy, which is the dependence among these features. The class-discriminative power would not change much if we remove redundant features. The mutual information can be utilized to estimate the redundancy of k features in the subset S as follows:

$$\min R(S_k) = \frac{1}{k^2} \sum_{x_i, x_j \in S_k} I(x_i, x_j)$$

The Minimal-Redundancy-Maximal-Relevance (mRMR) measure is defined as follows:

$$\max_{S_k} = D(S_k, C) - R(S_k)$$

$$\max_{S_k} = \frac{1}{k}\sum_{i=1}^{k} I(x_i, C) - \frac{1}{k^2}\sum_{x_i, x_j \in S_k} I(x_i, x_j)$$

The interpretation of the mRMR measure is that we want to maximize the relevance of features to the target class C ($D(S_k, C)$) and at the same time to minimize the redundancy between features ($R(S_k)$). It has been shown in [108] that the mRMR is equivalent to Max-Dependency for the first-order incremental search. Peng et. al. proposed to apply many heuristic search methods, such as incremental search, to solve the mRMR feature selection problem.

In summary, we have introduced the main feature construction methods, which motivate our research in this dissertation. In the next section, we describe the main focused supervised and unsupervised machine leanring algorithms.

2.1.2 Supervised Learning Algorithms

This section is devoted to an introduction of the supervised learning, which is the machine learning task to infer a function from a labeled training dataset. We first discuss about the generalization ability of supervised learning algorithms. Second, we describe one of the most popular supervised learning algorithms: the support vector machines.

Ability of generalization The generalization ability of machine learning algorithms can be predicted by using the well-known bounds based on the Vapnik-Chernovenkis (VC) dimension [133]. Given some machine learning algorithm f and let $\mathrm{TrainErr(f)}$ be its training error or fraction training set misclassified. Under the assumption that all the training samples and testing samples are independently drawn from a common generating distribution, Vapnik and Chernovenkis [133] showed that with probability of $1 - p$, the testing error ($\mathrm{TestErr(f)}$) of the machine learning algorithm f is bounded as follows:

$$\mathrm{TestErr(f)} \leq \mathrm{TrainErr(f)} + \sqrt{\frac{h(log(2m/h) + 1 - log(p/4))}{m}}$$

where m is the number of training samples; h is VC dimension or the measure of the f's power. In fact, h is the maximum number of points that can be arranged so that f shatters them. h does not depend on the choice of the training set. This bound gives us a way to estimate the error on future data based on only the training error and the VC dimension of the algorithm f.

However, the above assumption regarding the common distribution of training and testing samples is not always true in real world, since very often the process generating the training set is not the same as that selecting the testing set. In these cases, there is no overlap between the two sets. Therefore, one has to consider the off-training set (OTS) generalization error, i.e., generalization error for test sets that contain no overlap with the training set [142, 143]. Wolpert [142, 143] has stated that "one can't say: if empirical misclassification rate is low; the Vapnik-Chervonenkis dimension of your generalizer is small; and if the training set is large, then with high probability your OTS error is small.". This statement is an implication of the well known No-Free-Lunch (NFL) theorems [142, 143], which indicate that there are no a priori distinctions between learning algorithms. Also by this statement, there is no evidence of generalization ability of any machine learning if it is trained only on the training samples as a part of the whole data space. However, several machine learning algorithms were used in many applications with limited success.

In the following, we will describe the support vector machine algorithm, which has been applied for the intrusion detection tasks (see, for example, [66]).

2.1.2.1 Support Vector Machine

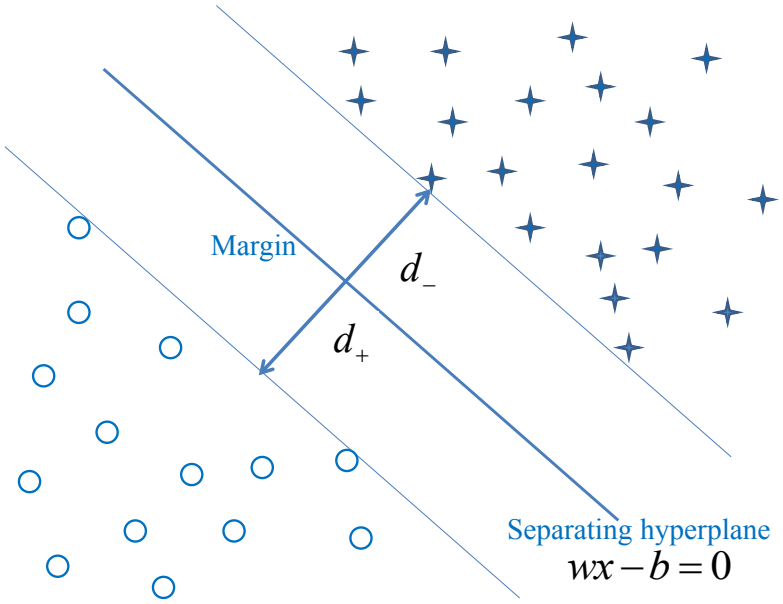


Figure 2.3: Support Vector Machines

We are given a training dataset D that contains m data points:

$$D = \{(a_i, c_i)|a_i \in \mathbb{R}^n, c_i \in \{-1, 1\}\}_{i=1}^m$$

Here a_i is an n-dimensional real vector and c_i is an indicator of the class where the point a_i belongs to.

A hyperplane, which separates the positive ($c = 1$) from negative ($c = -1$) instances, is called separating hyper-plane. The formula of the hyperplane is given as follows: $wx - b = 0$, where w is normal vector to the hyperplane, x is the point of the hyperplane, b is a real value and the $\frac{1}{||w||}$ is the perpendicular distance from the hyperplane to the origin.

Let $d_+(d_-)$ be the shortest distance from the separating hyperplane to the closest positive (negative) data point. Define the Margin of the separating hyperplane to be $((d_+ + d_-))$.

For the separable case when the positive and negative data points are linearly separated, they satisfy the following constraints:

$$\begin{cases} a_i w - b \geq 0, & for \quad c_i = 1, \\ a_i w - b \leq 0, & for \quad c_i = -1. \end{cases}$$

or they can be combined into one set of inequalities:

$$c_i(a_i w - b) - 1 \geq 0, \forall i.$$

Therefore, the margin is simply $d_+ + d_- = \frac{2}{||w||}$. In order to get the highest confidence in classification, we want to choose w and b to maximize the margin. This provides not only the best classification performance on the training data, but also leaves much room for the correct classification of the future data. The linear support vector machines problem can be formulated as follows [40]:

$$\max_{w \in \mathbb{R}^n, b \in \mathbb{R}} \frac{1}{||w||^2}$$

$$\text{such that} \quad \begin{cases} c_i(a_i w - b) \geq 1, \\ i = \overline{1, m}. \end{cases} \tag{2.4}$$

The problem 2.4 is a typical convex optimization problem, which we describe in more detail in the next section. Here we show the main ideas of solving the 2.4.

We first study the Lagrangian formulation [27] of the problem (2.4). The reason for doing this is that the constraints in (2.4) are replaced by constraints on the Lagrange multipliers themselves, which will be much easier to handle. With non-negative Lagrange multipliers $\alpha_i, i = \overline{1, m}$, the problem (2.4) is equivalent to the following problem:

$$\min_{w,b,\alpha} \{L_p = \frac{1}{2}||w||^2 - \sum_{i=1}^{m} \alpha_i [c_i(a_i w - b) - 1]\} \tag{2.5}$$

The problem (2.5) can now be solved by using standard quadratic programming techniques, such as the gradient descent or stochastic gradient descent algorithms [28, 104]. Another popular approach is to apply the Karush-Kuhn-Tucker (KKT) conditions as necessary and sufficient conditions for w, b, α to be a solution [69]. In the next section on the convex optimization for machine learning, we will discuss this method in more detail.

The Karush-Kuhn-Tucker (KKT) conditions of the problem (2.5) is given below:

$$\begin{cases} \frac{\partial}{\partial w_j} L_p = w_j - \sum_{i=1}^{m} \alpha_i c_i a_{ij} = 0, j = \overline{1, n}, \\ \frac{\partial}{\partial b} L_p = -\sum_{i=1}^{m} \alpha_i c_i, \\ c_i(a_i w - b) - 1 \geq 0, \\ \alpha_i [c_i(a_i w - b) - 1] = 0, \\ \alpha_i \geq 0, \forall i = \overline{1, m} \end{cases} \tag{2.6}$$

For the non-separable case that allows for incorrect classified instances, all the data points of D satisfy the following constraints:

$$\{c_i(a_i w - b) \geq 1 - \xi_i, i = \overline{1, m}\}$$

where $\xi_i \geq 0$ is a slack variable that measures the degree of misclassification of the data point a_i. The linear support vector machines problem can then be formulated as follows [40]:

$$\min_{w,b,\xi} \frac{1}{2}||w||^2 + C \sum_{i=1}^{m} \xi_i \tag{2.7}$$

$$\text{such that} \quad \begin{cases} c_i(a_i w - b) \geq 1 - \xi_i, \\ C > 0, \xi_i \geq 0, i = \overline{1, m}. \end{cases}$$

By using Lagrange multipliers, we equivalently transform the problem (2.7) to the following problem:

$$\min_{w,b,\xi,\alpha,\beta} \{H_p = \frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\alpha_i[c_i(a_i w - b) - 1 + \xi_i] - \sum_{i=1}^{m}\beta_i\xi_i\} \tag{2.8}$$

where $\alpha_i, \beta_i \geq 0, i = \overline{1, m}$.
The KTT conditions for the problem (2.8) are as follows:

$$\begin{cases} \frac{\partial}{\partial w_j} H_p = w_j - \sum_{i=1}^{m} \alpha_i c_i a_{ij} = 0, j = \overline{1, n}, \\ \frac{\partial}{\partial b} H_p = -\sum_{i=1}^{m} \alpha_i c_i = 0, \\ \frac{\partial}{\partial \xi_i} H_p = C - \alpha_i - \beta_i = 0, \\ \alpha_i[c_i(a_i w - b) - 1 + \xi_i] = 0, \\ c_i(a_i w - b) - 1 + \xi_i \geq 0, \\ \beta_i \xi_i = 0, \alpha_i, \beta_i, \xi_i \geq 0, \forall i = \overline{1, m} \end{cases} \tag{2.9}$$

2.1.3 Unsupervised Learning Algorithms

This section describes the main idea of unsupervised learning and discuss a concrete sample, which is one of the most popular unsupervised learning algorithm and also the motivation for our new algorithm in the next chapter.

By constrast with supervised learning, in unsupervised learning there are no target output labels in the training and testing datasets. The machine simply receives inputs $x_1, x_2, ...$ and the task is to learn and differentiate them. It seems to be mysterious to image what the machine could possibly learn from the data without knowledge about samples, such as normal and abnormal instances in network intrusion detection. However, in the unsupervised learning it is possible to find and learn the hidden structures inside the unlabeled data. Two groups of the unsupervised learning algorithms are: 1) Dimensionality reduction and 2) Clustering analysis.

Ad.1 Dimensionality reduction [50] is the process of transforming the data in the high-dimensional space into a space of fewer dimensions. This process has a similar goal with the feature selection in section 2.1.1.2, but in this case the input data doesn't have output labels. Approaches to dimensionality reduction includes, for example, Principal Component Analysis (PCA), Independent Component Analysis (IDA) and so on.

Ad.2 Clustering analysis [45] is the task of dividing a set of samples into clusters, in each of which samples are more similar to each other by means of a distance, such as the euclidean distance, than those in other cluster. In some cases, clustering can also be considered as a form of classification in that it derives labels of samples with cluster labels. Various types of clustering are hierarchical, partitional, exclusive, fuzzy clustering algorithms and so on.

In the following, we describe the K-means clustering algorithm, which is one of the most popular unsupervised learning algorithm, in more detail.

2.1.3.1 K-means Clustering Algorithm

K-means clustering algorithm is a method of clustering analysis that aims to partition m input instances into K clusters, in which each instance belongs to the nearest cluster by means of the mean value. More formally, let $S = (a_1, a_2..., a_m)$ be a dataset with m instances, each of which is an $n-$dimensional vector. The objective in K-means clustering is to group these instances into categories $C_1, C_2, ..., C_K$ for the given value K, such that the following objective function is maximized:

$$J_K = \sum_{k=1}^{K} \sum_{i \in C_k} (a_i - \mu_k)^2 \tag{2.10}$$

Here μ_k represents the mean vector of the instances from C_k: $\mu_k = \frac{1}{m_k} \sum_{i \in C_k} a_i$, where $m_k = |C_k|$ is the number of instances in C_k.

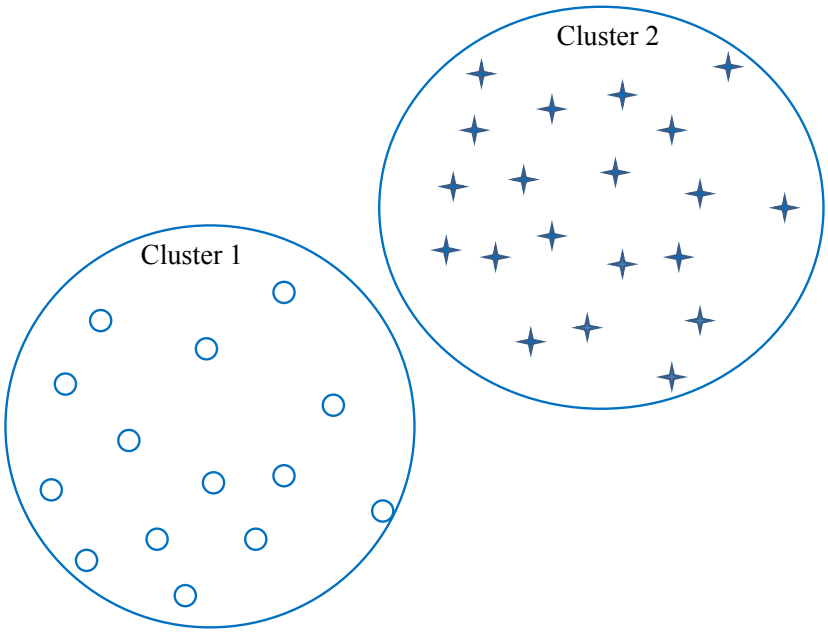


Figure 2.4: Sample of K-means clustering with K = 2

The first idea of the K-means clusteting algorithm was introduced by Hugo Steinhaus in 1956 [127] and the term "K-means" was first used by James MacQueen in 1967 [79]. In 1982, Lloyd [75] proposed a heuristic algorithm for K-means clustering, which then became a standard and widely used approach of unsupervised learning in many different application domains. We describe the Lloyd's algorithm in more detail below.

Given the dataset $S = (a_1, a_2..., a_m)$, the algorithm consists of two following iterative steps:

- **Initialization step:** Select arbitrarily initial cluster centers $C = \{c_1 = a_1, c_2 = a_2, .., c_K = a_K\}$;
- **Assignment step:** Assign every $a_i \in S, i = \overline{1, m}$ to the cluser C_k whose cluster center c_k is closest to it, i.e., $||a_i - c_k|| \leq ||a_i - c_j||$ for all $k \neq j$;
- **Update step:** Calculate the new means to be the centroid of the instances in the cluster: $c_k = \frac{1}{|C_k|} \sum_{i \in C_k} a_i$

The algorithm is converged when the assignments no longer change. See 2.1 for summary of Lloyd's algorithm.

Algorithm 2.1: Lloyd's method for K-means clustering.

1. K Initial cluster centers $C = \{c_1 = a_1, c_2 = a_2, .., c_K = a_K\}$;
2. Assign $a_i \in S, i = \overline{1, m}$ to the cluser C_k if $||a_i - c_k|| \leq ||a_i - c_j||$ for all $k \neq j$;
3. Set $c_k = \frac{1}{|C_k|} \sum_{i \in C_k} a_i$;
4. If clusters or centers have changed, goto step 2. Otherwise, stop.

In summary, previous sections describe the main focused machine learning algorithms, which are feature construction methods, supervised and unsupervised learning. In the next section, we introduce two main optimization techniques for machine leanring: convex and non-convex optimization.

2.2 Optimization for Machine Learning

Many machine learning problems have been formulated as convex and non-convex optimization problems [24]. In this section, we first describe the main concepts and methods of the widely-used convex optimization for machine learning: Lagrange duality and Karush-Kuhn-Turker (KKT) conditions. Second, we introduce the non-convex optimization techniques, which include the branch and bound algorithm and the D.C. programming approach.

2.2.1 Convex Optimization for Machine Learning

Techniques of Convex Optimization (CO) become important tools for machine learning algorithms, such as Support Vector Machines. The uniqueness of the optimal solution allows the understandings of the learning process in the theoretical analysis. Moreover, the use of CO tools permits fast implementations of the learning algorithms in practice. In this section, we first introduce the basic concepts of CO, such as convex set, convex function and convex optimization problems. Second, we provide the duality theorems and the Karush-Kuhn-Turker (KKT) conditions, which are necessary and sufficient conditions for optimality of the convex optimization problem.

Definition 2.1 (Convex set)
A set $S \subseteq \mathbb{R}^n$ is convex if for all $x, y \in S$ and any $\alpha \in [0, 1]$, the point $\alpha x + (1 - \alpha)y$ is in S or

$$\alpha x + (1 - \alpha)y \in S$$

.

Definition 2.2 (Convex function)
A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if for all $x, y \in dom f$ and any $\alpha \in [0, 1]$, we have the following inequality:

$$f(\alpha x + (1 - \alpha y)) \leq \alpha f(x) + (1 - \alpha) f(y)$$

.

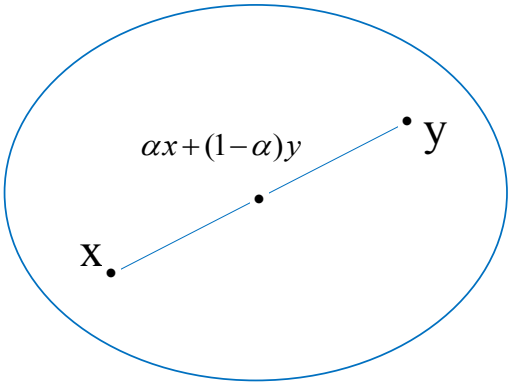


Figure 2.5: Sample of a convex set

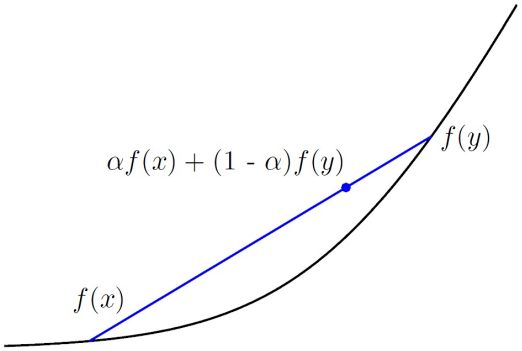


Figure 2.6: Sample of a convex function

Theorem 2.1 (First order convexity conditions)
Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable. Then f is convex if and only if for all $x, y \in domf$ the following inequality is satisfied:

$$f(y) \geq f(x) + \nabla f(x)^T (y - x)$$

Theorem 2.2 (Second order convexity conditions)
Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is twice differentiable. Then f is convex if and only if for all $x \in domf$ the following inequality is satisfied:

$$\nabla^2 f(x) \geq 0$$

Definition 2.3 (Convex optimization problems)
An optimization problem is convex if its objective $f(x)$ is a convex function, the inequality constraints $g_i(x)$ are convex, and the equality constraints $h_j(x)$ are affine:

$$\min_x f(x) \,(\text{Convex function}),$$

$$\text{such that} \quad \begin{cases} g_i(x) \leq 0, i = 1, .., m, (\text{Convex sets}), \\ h_j(x) = 0, j = 1, .., p, (\text{Affine}). \end{cases} \tag{2.11}$$

Theorem 2.3 (Global minimizer)
If x is a local minimizer of a convex optimization problem, it is a global minimizer.

Theorem 2.4 (Condition for a global minimizer)
$\nabla f(x) = 0$ *if and only if x is a global minimizer of $f(x)$.*

2.2.1.1 Lagrange Duality

This section describes a powerful concept in convex optimization theory known as Lagrange duality [104]. We focus on the main intuitions and mechanics of Lagrange duality, in particular, we describe the Lagrangian, its relation to primal and dual problems, and the Karush-Kuhn-Tucker (KKT) conditions as necessary and sufficient conditions for optimality of the convex optimization problem as formulated in Eq. (2.11).

Definition 2.4 (Lagrangian)
Given the convex constrained minimization problem (2.11)*, the Lagrangian is a function* $\boldsymbol{L} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$*, defined as*

$$\boldsymbol{L}(x, \alpha, \beta) = f(x) + \sum_{i=1}^{m} \alpha_i g_i(x) + \sum_{j=1}^{p} \beta_j h_j(x) \tag{2.12}$$

where x is primal variable of the Lagrangian and α, β are dual variables of the Lagrangian.

To show the relationship between the Lagrangian and the original optimization problem (2.11), we introduce the notions of the primal and dual problems associated with the Lagrangian.

Definition 2.5 (Primal problem)

$$\min_x \underbrace{\Big[\max_{\alpha,\beta:\alpha_i \geq 0, \forall i} \boldsymbol{L}(x, \alpha, \beta) \Big]}_{\theta_P(x)} = \min_x \theta_P(x) \tag{2.13}$$

In this equation, the function $\theta_P : \mathbb{R}^n \to \mathbb{R}$ is called the primal objective and a point $x \in \mathbb{R}^n$ is primal feasible if $g_i(x) \leq 0, i = 1, ..., m$ and $h_j(x) = 0, j = 1, ..., p$.

Definition 2.6 (Dual problem)

$$\max_{\alpha,\beta:\alpha_i \geq 0, \forall i} \underbrace{\Big[\min_x \boldsymbol{L}(x, \alpha, \beta) \Big]}_{\theta_D(\alpha,\beta)} = \max_{\alpha,\beta:\alpha_i \geq 0, \forall i} \theta_D(\alpha, \beta) \tag{2.14}$$

Here, the function $\theta_D : \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ is called the dual objective and a point (α, β) is dual feasible if $\alpha \geq 0$.

Theorem 2.5
We call $p^* = \theta_P(x^*)$ and $d^* = \theta_D(\alpha^*, \beta^*)$ the optimal values of primal and dual problems, respectively. If (α, β) is dual feasible, then $\theta_D(\alpha, \beta) \leq p^*$.

The theorem shows that given any feasible (α, β), the dual objective $\theta_D(\alpha, \beta)$ provides a lower bound on the optimal value p^* of the primal problem.

Theorem 2.6 (Weak duality)
For any pair of primal and dual problems, $d^* \leq p^*$.

Theorem 2.7 (Strong duality)
For any pair of primal and dual problems which satisfy certain technical conditions called constraint qualifications, then $d^* = p^*$.

A number of different constraint qualifications exist, of which the most commonly invoked constraint qualification is know as Slater's condition: a primal/dual problem pair satisfy Slater's condition if there exist some feasible primal solutions x for which all inequality constraints are strictly satisfied (i.e., $g_i(x) < 0, i = 1, ..., m$). Below is the necessary and sufficient conditions for optimality of the convex optimization problem (2.11).

Theorem 2.8 (The Karush-Kuhn-Tucher (KKT) conditions)
Suppose that $x^* \in \mathbb{R}^n$, $\alpha^* \in \mathbb{R}^m$ and $\beta^* \in \mathbb{R}^p$ satisfy the following conditions:

1. *(Primal feasibility) $g_i(x^*) \leq 0, i = 1, ..., m$ and $h_j(x^*) = 0, j = 1, ..., p$,*
2. *(Dual feasibility) $\alpha_i^* \geq 0, i = 1, ..., m$,*
3. *(Complementary slackness) $\alpha_i g_i(x^*) = 0, i = 1, ..., m$, and*
4. *(Lagrangian stationarity) $\nabla_x \boldsymbol{L}(x^*, \alpha^*, \beta^*) = 0$*

Then x^* is primal optimal and α^*, β^* are dual optimal. Furthermore, if strong duality holds, then any primal optimal x^* and dual optimal (α^*, β^*) must satisfy the conditions 1 through 4.

Beside the convex optimization formulation, various machine learning algorithms are non-convex optimization problems. In the next section, we describe non-convex optimization techniques for machine learning.

2.2.2 Non-convex Optimization for Machine Learning

This section describes non-convex optimization (NCO) techniques to solve a mixed 0-1 linear programming problem, which is a common reformulated form of many machine learning algorithms that we will study in the next chapter. In particular, we provide the main ideas of two approaches: the branch and bound and D.C. (Difference of Convex Functions) algorithms.

The general formulation of Mixed 0-1 Linear Programming Problem is given as follows:

$$\min f(x, y) = c^T x + d^T y,$$

$$\text{such that} \begin{cases} Ax + By \leq b, \\ x \in [lb_x, ub_x], \\ x \in \mathbb{R}^n, y \in \{0, 1\}^m, \end{cases} \tag{2.15}$$

where the objective function $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is a linear function. The variable vector x is bounded by lb_x and ub_x, which are constants specifying the lower and upper bounds of x. The variable vector y is binary: $y \in \{0,1\}^m$.

The difficulty of solving (2.15) is that the binary variable y destroys the convexity property of the constraint set. Therefore, we cannot apply the convex optimization techniques described in previous section to solve the (2.15). In the following, we describe two popular approaches to non-convex optimization problems in general and to Mixed 0-1 Programming problems in particular: the Branch & Bound and the DC (Difference of Convex Functions) Programming.

2.2.2.1 Branch and Bound

In this subsection, we describe the main principles of the Branch-and-Bound algorithm and its application to solve the problem (2.15).

Let the set of constraints in (2.15) be the region of feasible solutions $S = \{(x,y) \in \mathbb{R}^n \times \{0,1\}^m : Ax + By \leq b, x \in [lb_x, ub_x], y \in \{0,1\}^m\}$. The problem (2.15) can then be expressed as:

$$\min_{(x,y)\in S} f(x,y) = c^T x + d^T y. \tag{2.16}$$

Let P be a set of potential solutions of the (2.15). The set P contains the set S for which the function $f(x,y)$ is still well defined. For example, P can be the set of constraints without limitations on the binary variable y: $P = \{(x,y) \in \mathbb{R}^n \times \mathbb{R}^m : Ax + By \leq b, x \in [lb_x, ub_x], y \in [0,1]\}$. Let $g(x,y)$ be a function defined on S (or P) with the property that $g(x,y) \leq f(x,y)$ for all $(x,y) \in S$ (resp. $(x,y) \in P$). The function $g(x,y)$ is called the bounding function of the function $f(x,y)$. Both the set P and the function $g(x,y)$ are very useful in the Branch-and-Bound context. Figure 2.7 illustrates an example relation between the set S of feasible solutions and the set P of potential solutions and between the objective function $f(x,y)$ and the bounding function $g(x,y)$.

In general, solving a problem with the Branch-and-Bound algorithm is to search through a search tree, in which the root node corresponds to the original problem, and every other node is a subproblem of the original problem. Given a current node N of the search tree, the children of N are subproblems derived from N by adding some new constraints. For example, the problem (2.15) can be divided into the two following subproblems by splitting the constraint of the variable $x \in [lb_x, ub_x]$:

$$\min_{(x,y)\in S_1} f(x,y) = c^T x + d^T y. \tag{2.17}$$

where $S1 = \{(x,y) \in \mathbb{R}^n \times \{0,1\}^m : Ax + By \leq b, x \in [lb_x, \frac{lb_x + ub_x}{2}], y \in \{0,1\}^m\}$

and

$$\min_{(x,y)\in S_2} f(x,y) = c^T x + d^T y. \tag{2.18}$$

where $S2 = \{(x,y) \in \mathbb{R}^n \times \{0,1\}^m : Ax + By \leq b, x \in (\frac{lb_x + ub_x}{2}, ub_x], y \in \{0,1\}^m\}$

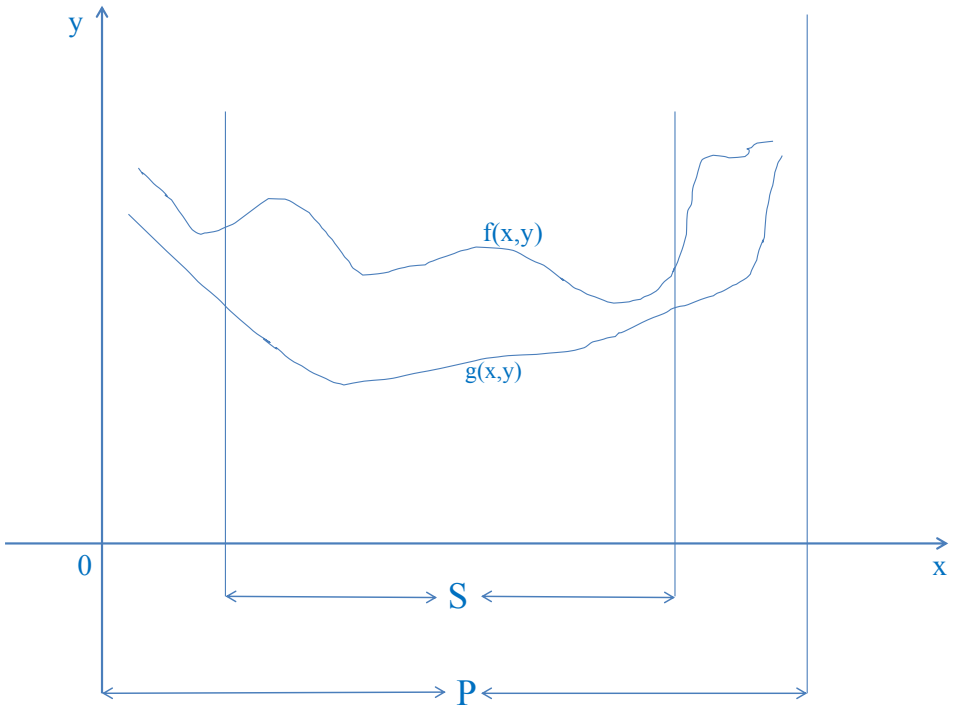


Figure 2.7: Sample of a relation between the set S of feasible solutions and the set P of potential solutions and between the objective function $f(x, y)$ and the bounding function $g(x, y)$ [37]

The search tree is constructed dynamically during the search and consists initially of only the root node as the original problem. To each node in the search tree, a bounding function $g(x, y)$ associates a real number called the bound for the node. For the leaves of the search tree, the bound equals the value of the corresponding solution, whereas for internal nodes the value is a lower bound for the value of any solution of the subproblem corresponding to the node. The general scheme of the Branch-and-Bound algorithm is sketched as follows:

Algorithm 2.2: General scheme of Branch and Bound algorithm for a problem F.

1. Incumbent $U = +\infty$;
2. Select a sub-problem F_i;
3. If (F_i is infeasible), then delete it;
4. Else, compute the lower bound $lb(F_i)$;
5. If ($lb(F_i) \geq U$), then delete F_i;
6. Else, if (the solution to F_i satisfies all the constraints of F), then $U := lb(F_i)$;
7. Else, break F_i into sub-problems and goto step 2.

The Branch-and-Bound algorithm for a minimization problem hence contains three main components [37]:

- A bounding function providing for a given subspace of the solution space a lower bound for the best solution value obtainable in the subspace.
- A strategy for selecting the live solution subspace to be investigated in the current iteration
- A branching rule to be applied if a subspace after investigation cannot be discarded, hereby subdividing the subspace considered into two or more subspaces to be investigated in subsequent iterations.

Since the complexity of the Branch-and-Bound algorithm depends on the number of variables and constraints, in some cases it is unsuitable for large-scale optimization problems. In the next section, we describe an alternative method for the large-scale non-convex optimization problems: The D.C. (Difference of Convex Functions) programming approach.

2.2.2.2 DC Programming Approach

In this subsection, we introduce the DC Programming approach for solving a large-scale Mixed 0-1 Linear Programming problem (2.15).

The DC algorithm (Difference of Convex Functions algorithm or DCA) has been first introduced by Pham in 1985 as an extension of the sub-gradient methods, and extensively developed by Le and Pham since then [16, 17]. The DCA is designed for nonlinear, non-convex, non-smooth, and large-scale programming problems and has successfully been applied for solving many real world non-convex optimization problems [15, 18, 103, 126, 130].

The DCA is a continuous optimization method for solving DC program with convex constraints. To be able to apply the DCA for solving the problem (2.15), we first need to handle the discrete constraint of variable $y \in \{0,1\}^m$ by representing it via a continuous formulation. Second, we can transform the problem (2.15) to a DC program.

Fortunately, the binary set $\{0,1\}^m$ can easily be represented as

$$\{y \in \{0,1\}^m\} \Leftrightarrow \{y \in [0,1]^m : p(y) \leq 0\} \tag{2.19}$$

where the function $p(y) = \sum_{i=1}^{m} y_i(1-y_i)$ is continuous and $p(y) \geq 0, \forall y \in [0,1]^m$ and $p(y) = 0$ if and only if $y \in \{0,1\}^m$.

Let $K = \{(x,y) \in \mathbb{R}^n \times [0,1]^m : Ax + By \leq b, x \in [lb_x, ub_x], y \in [0,1]^m\}$ be a nonempty and compact set. The problem (2.15) can be expressed as:

$$\min\{f(x,y) = c^T x + d^T y : (x,y) \in K, y \in \{0,1\}^m\}. \tag{2.20}$$

By using (2.19), we formulate the problem (2.20) as an equivalent problem:

$$\min\{f(x,y) = c^T x + d^T y : (x,y) \in K, p(y) \leq 0\}. \tag{2.21}$$

Note that the problem (2.21) can't be handled directly by DCA due to the non-convex constraint $p(y) \leq 0$. Fortunately, the problem can be reformulated as a DC program via penalization techniques with a positive penalty parameter t as follows:

$$\min\{F_t(x,y) = c^T x + d^T y + tp(y) : (x,y) \in K\}. \tag{2.22}$$

When t is a large positive number, minimizing $F_t(x,y)$ over the convex set K will either force $p(y)$ to be zero or force y to be binary. According to the general result of the penalty method [77], given a large number t, the minimizer of the (2.22) should be found in a region where $p(y)$ is relatively small.

Definition 2.7 (Cubic neighborhood)
(see [77]) Let $\widetilde{y} \in \{0,1\}^m$. The set $N(\widetilde{y}) = \{y : ||y - \widetilde{y}||_\infty \leq \frac{1}{5}\}$ is called a $\frac{1}{5}$-cubic neighborhood of the binary point $\widetilde{y}$.

Definition 2.8 (Global Minimizer via penalty method)
(see [77]) Suppose that t is large enough, if (x^*, y^*) is a global minimizer of (2.22) and y^* is in a $\frac{1}{5}$-cubic neighborhood of a binary point $\widetilde{y}$, then $(x^*, \widetilde{y})$ is a solution of the problem (2.20).

However, the problem (2.22) is a non-convex, nonlinear optimization problem that is difficult to solve. Fortunately, we can represent $F_t(x,y)$ as a difference of convex functions (DC):

$$F_t(x,y) = g(x,y) - h(y)$$

where $g(x,y) = f(x,y) = c^T x + d^T y$ and $h(y) = -tp(y)$ are convex functions. Therefore, the problem 2.22 can be reformulated as a DC programming problem:

$$\min\{F_t(x,y) = g(x,y) - h(y) : (x,y) \in K\}. \tag{2.23}$$

According to the general framework of DCA [15, 18, 103, 126, 130], we need to construct the two following sequences: $\{X^k = (x^k, y^k)\}$ and $\{Y^k = (u^k, v^k)\}$. To get the $Y^k = (u^k, v^k)$, we compute the subdifferential of the function $h(y)$ at the point $X^k = (x^k, y^k)$, denoted by $\partial h(x^k, y^k)$.

Definition 2.9 (Subdifferential)
(see [16]) Let $\Gamma_0(\mathbb{R}^n)$ denote the convex cone of all lower semi-continuous proper convex functions on $\mathbb{R}^n$. For all $\theta \in \Gamma_0(\mathbb{R}^n)$ and $x_0 \in dom(\theta) = \{x \in \mathbb{R}^n : \theta(x) < +\infty\}$, $\partial\theta(x_0)$ denotes the subdifferential of θ at x_0.

$$\partial\theta(x_0) = \{y \in \mathbb{R}^n : \theta(x) \geq \theta(x_o) + < x - x_0, y >, \forall x \in \mathbb{R}^n\}$$

It is well-known that if θ is differentiable at x_0, then $\partial\theta(x_0)$ reduces to the $\{\nabla\theta(x_0)\}$. In our case, the convex function $h(y)$ is defined as $h(y) = -tp(y) = -t\sum_{i=1}^m y_i(1-y_i)$, which is a continuously differentiable function. Therefore, $\partial h(x^k, y^k) = \{\nabla_{(x,y)} h(x^k, y^k)\}$. The vector $Y^k = (u^k, v^k)$ can be computed as follows:

$$(u^k, v^k) \in \partial h(x^k, y^k) \Leftrightarrow (u^k = 0, v^k = -t\sum_{i=1}^m (1 - 2y_i^k)e^i) \tag{2.24}$$

where e^i is the i-th unit vector of $\mathbb{R}^m$.

Let $g^* = sup\{< x, y > -g(x) : x \in \mathbb{R}^n\}$ be the conjugate function of g. For computing $(X^{k+1} = (x^{k+1}, y^{k+1})) \in \partial g^*(u^k, v^k)$, it is equivalent to solve the following linear program:

$$(x^{k+1}, y^{k+1}) \in argmin\{g(x, y) - < (x, y), (u^k, v^k) >: (x, y) \in K\} \tag{2.25}$$

This DC algorithm can be terminated when one of the following conditions is satisfied:

1. One of the sequences $\{X^k\}$ or $\{Y^k\}$ converges, i.e.,

$$||X^{k+1} - X^k|| \leq \epsilon_1$$

or

$$||Y^{k+1} - Y^k|| \leq \epsilon_1$$

2. The sequence $\{F_t(X^k) = g(x^k, y^k) - h(y^k)\}$ converges, i.e.,

$$||F_t(X^{k+1}) - F_t(X^k)|| \leq \epsilon_2$$

3. If (x^k, y^k) is in a ϵ_3-cubic neighborhood of a binary point $(x^k, \overline{y}^k)$.

The DC algorithm is summarized as follows:

Algorithm 2.3: Difference of Convex Functions algorithm (DCA) [103]

1. **Initialization:**
   - Choose an initial point $X^0 = (x^0, y^0)$
   - Let t be a large enough positive value
   - Let ϵ_1, ϵ_2 and ϵ_3 be sufficiently small positive values
   - Iteration number $k = 0$
2. **Repeat to construct the two following sequences:**
   - $(u^k, v^k) \in \partial h(x^k, y^k) \Leftrightarrow (u^k = 0, v^k = -t\sum_{i=1}^m (1 - 2y_i^k)e^i)$
   - $(x^{k+1}, y^{k+1}) \in argmin\{g(x, y) - < (x, y), (u^k, v^k) >: (x, y) \in K\}$
3. **Until one of the conditions is satisfied:**
   - $||X^{k+1} - X^k|| \leq \epsilon_1$ or $||Y^{k+1} - Y^k|| \leq \epsilon_1$
   - $||F_t(X^{k+1}) - F_t(X^k)|| \leq \epsilon_2$
   - $||(x^k, y^k) - (x^k, \widetilde{y}^k)||_\infty \leq \epsilon_3$

The convergence of the DCA is described in the following theorem (see [16]).

Theorem 2.9 (Convergence properties of the DC algorithm)

1. *DCA generates a sequence $\{(x^k, y^k)\}$ such that the sequence $\{F_t(x^k, y^k)\}$ is decreasing and bounded below.*
2. *If the optimal value of (2.23) is finite and the infinite sequences $\{X^k\}$ and $\{Y^k\}$ are bounded, then every limit point X^∞ of the sequence $\{X^k\}$ is a Karush-Kuhn-Tucker point.*

In summary, this chapter provided a background on various machine learning algorithms, which are integral to and motivate the current dissertation. First, Section 2.1 described three main groups of machine learning techniques in more detail: feature extraction/selection, supervised learning and unsupervised algorithms with concrete illustrations. Second, Section 2.2 presented two popular operational research approaches for machine learning: convex and non-convex optimization techniques. In the next chapter, we introduce the new developed machine learning algorithms.

Chapter 3

New Reliable Machine Learning Algorithms

Entities should not be multiplied unnecessarily.

WILLIAM OF OCCAM, C.1320

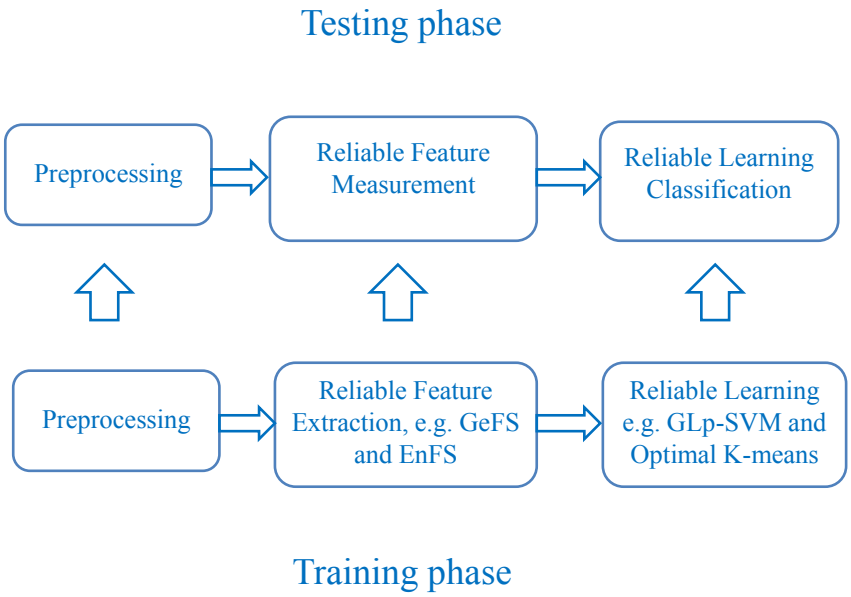


Figure 3.1: Model of a reliable statistical pattern recognition system

A critical question when utilizing automatic pattern-recognition systems for decision making (e.g. intrusion detection) is whether we can trust the outcomes of a classifier. In other words, is the classification accuracy reliable? According to Figure 3.1, the total reliability of a pattern-recognition system is affected by the preprocessing reliability, the feature extraction/selection reliability and the classification reliability. In this section, we describe the new efficient feature selection methods, supervised and unsupervised learning algorithms to address this reliability issue. In particular, we propose new Generic-Feature-Selection-Measure [93, 94], a new Reliable Feature-Selection Method [98], a new Reliable Ensemble-Feature-Selection Framework [96], General Lp-norm Support Vector Machines [97, 100], and Optimal K-means Clustering [92].

3.1 A New Reliable Feature-Selection Process

A feature-selection process consists of a feature-selection method and a search strategy for relevant features, such as exhaustive search, best-first search or genetic search. There exist three categories of feature-selection methods: the filter, the wrapper and the embedded models [56, 74]. The wrapper model utilizes a classifier's performance in assessing and selecting features, whereas the filter model considers statistical properties of the data set without involving any learning algorithms. The embedded model integrates feature selection into the learning process of a classifier.

We distinguish two main factors that affect the reliability in the feature-selection process: (1) the choice of feature-selection methods and (2) the search strategies for relevant features.

Add. 1 Each data set has its own statistical properties. Based on the estimation of these properties, a feature-selection method seeks representative patterns of the data. Thus, the wrong choice of feature-selection methods leads to non-representative patterns and the feature-selection results become unreliable. Further on, the classifier's performance might not be steady as the classifier has learnt on non-representative data set.

Add. 2 The second factor is the search strategies employed in the feature-selection process. Even though we utilize appropriate feature-selection methods, the heuristic-search strategies might provide inconsistent results with different executions yielding different minima. Consequently, the feature-selection results are unreliable as well.

In this section, we first introduce a formal definition of a reliable feature-selection process. Our definition reflects the main factors that affect the reliability in a feature-selection process. Moreover, our definition establishes formal measurements of the feature-selection reliability, i.e., the **steadiness** of a classifier's performance and the **consistency** in search for relevant features.

We then propose new methods in order to address the main causes of the unreliable feature-selection process: (3) the inappropriate choice of feature-selection methods and (4) the heuristic search strategies.

Add. 3 We introduce a new methodology for determining appropriate instances of a class including several feature-selection methods of the filter model, e.g. the correlation-feature-selection measure (CFS) [59] and the minimal-redundancy-maximal-relevance measure (mRMR) [108]. We call this class a generic-feature-selection (GeFS) measure. Following our new proposed methodology, we first analyze the statistical property of a data set. We then choose the CFS measure if the data set has many features that are linearly correlated to the class label and to each other. Otherwise, the mRMR measure is selected.

Add. 4 We propose a new search approach that ensures the globally optimal feature subset by means of the GeFS measure. The new approach is based on solving a mixed 0-1 linear programming problem (M01LP) by using the branch-and-bound search algorithm. In this M01LP problem, the number of constraints and variables is linear in the number of full set features. In addition, the new proposed search method allows us to easily integrate expert knowledge in the feature-selection process. That significantly increases the reliability of the feature relevance estimation.

3.1.1 Definition of Reliability in Feature-Selection Process[1]

In this subsection, we introduce a formal definition of a reliable feature-selection process. As we discussed in the introduction, the reliability of a feature-selection process is affected by the choice of feature-selection methods and the search strategy for relevant features. Thus, we measure the reliability of a feature-selection process via the **steadiness** of a classifier's performance and the **consistency** in search for relevant features as explained below. We first provide several important notations.

Given a data set D, a classifier C and a feature-selection method FS. Suppose that we run the FS algorithm M times to select features from the data set D . With different executions of a search strategy utilized in the feature-selection process, the feature-selection results might be different. Let $X_i, (i = \overline{1, M})$ be the selected feature subset in the i^{th} run. $Acc_i, (i = \overline{1, M})$ is the classification accuracy of the classifier C performed on X_i. Acc_F is the classification accuracy of the C performed on full set of features.

Definition 3.1 (Consistency of feature selection)
A search strategy utilized in feature-selection process is consistent, with level of approximation α or $\alpha_{\text{consistent}}$, if, for a given M,

$$\frac{|X_1 \cap X_2 \cap ... \cap X_M|}{|X_1 \cup X_2 \cup ... \cup X_M|} = \alpha. \tag{3.1}$$

The greater the α is, the more consistent the search strategy is. When $\alpha = 1$ for every M, we say that the search strategy is truly consistent.

Definition 3.2 (Steadiness of feature selection)
A feature-selection method generates $\beta_{\text{steadiness}}$ of the classifier C, if, for a given M:

$$\beta = \frac{Acc_F - \frac{1}{M}\sum_{i=1}^{M} |Acc_F - Acc_i|}{Acc_F}. \tag{3.2}$$

The greater the β is, the better the classifier's steadiness safeguarded by the feature-selection method is. Thus, the wrong choice of feature-selection methods might affect the steadiness of a classifier's performance.

Definition 3.3 (Reliability of feature selection)
A feature-selection method is called $(\alpha, \beta)_{\text{reliable}}$, if, for a given M, the search strategy utilized in the feature-selection process is $\alpha_{\text{consistent}}$ and the feature-selection method generates $\beta_{\text{steadiness}}$ of the classifier C.

In conventional view, the reliability [14] is defined in terms of stability, consistency and equivalence. However, in our case the equivalence condition contradicts our defined consistency condition. The reason is that in terms of equivalence condition determined by Dai [42] a feature-selection process would be considered reliable if different selected feature subsets lead to the same performance of a classifier; whereas, the consistency condition allows only one feature subset to be selected by the search strategy. Thus, in feature selection we do not take the equivalence measurement into account.

In the next subsection, we introduce a generic-feature-selection (GeFS) measure and propose a new methodology for determining appropriate instances of the GeFS measure. A new search method for relevant features by means of the GeFS measure will be described subsequently.

[1] A part of this section is published under the title:

Nguyen, H. T., Franke, K., and Petrović, S. Reliability in a feature-selection process for intrusion detection. In *Reliable Knowledge Discovery*, H. Dai, J. N. K. Liu, and E. Smirnov, Eds. Springer US, 2012, pp. 203-218.

3.1.2 Generic Feature Selection Measure[2]

In this subsection, we give an overview of the generic feature selection ($GeFS$) measure, which belongs to the filter model for feature selection. As discussed in the Section 2.1.1.2, the filter model considers statistical characteristics of a data set directly without involving any learning algorithms [56]. Due to the computational efficiency, the filter method is usually used 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 relationships between features that make certain features redundant. It is well known that the redundant features can reduce the performance of a pattern recognition system. Therefore, the feature-subset-evaluating methods are more suitable for selecting features in many applications. A major challenge in the reliable feature selection process is to choose appropriate measures that can precisely determine the relevance and the relationship between features of a given data set.

Since the relevance and the relationship are usually characterized in terms of correlation or mutual information, we focus on two measures: the correlation-feature-selection (CFS) measure [59] and the minimal-redundancy-maximal-relevance (mRMR) measure [108]. We show that these two measures can be fused and generalized into a generic-feature-selection (GeFS) measure. We reformulate the feature selection problem by means of the GeFS measure as a polynomial mixed 0-1 fractional programming (PM01FP) problem. We improve the Chang's method [33, 34] in order to equivalently reduce this PM01FP problem into a mixed 0-1 linear programing (M01LP) problem. Finally, we propose to use the branch-and-bound algorithm to solve this M01LP problem, whose optimal solution is also the globally optimal feature subset.

Definition 3.4 (Generic feature selection measure)
A generic feature-selection measure utilized in the filter model is a function $GeFS(x)$, which has the following form:

$$GeFS(x) = \frac{a_0 + \sum_{i=1}^{n} A_i(x)x_i}{b_0 + \sum_{i=1}^{n} B_i(x)x_i}, x \in \{0,1\}^n \tag{3.3}$$

In this definition, the vector x is binary: $x = (x_1, \ldots, x_n)$; the binary value of each variable x_i indicates 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 $x_1, \ldots, x_n$; n is the number of features.

Definition 3.5 (Optimization of generic feature-selection measure)
The feature selection problem is to find $x \in \{0,1\}^n$ that maximizes the function $GeFS(x)$.

$$\max_{x \in \{0,1\}^n} GeFS(x) = \frac{a_0 + \sum_{i=1}^{n} A_i(x)x_i}{b_0 + \sum_{i=1}^{n} B_i(x)x_i} \tag{3.4}$$

Following the definition 3.5, it is easy to integrate the expert knowledge into feature selection process. In fact, to get more reliable feature selection results, if domain experts want to select T features from n full-set features and want a particular important feature,

[2] A part of this section is published under the title:

Nguyen, H. T., Franke, K., and Petrović, S. Towards a generic feature-selection measure for intrusion detection. In *International Conference Pattern Recognition* (2010), pp. 15291532.

for example x_i, to appear in the selected feature set, then the problem of feature selection by means of the generic feature selection measure ($GeFS$) is defined as follows:

$$\max_{x\in\{0,1\}^n} GeFS(x) = \frac{a_0 + \sum_{i=1}^{n} A_i(x)x_i}{b_0 + \sum_{i=1}^{n} B_i(x)x_i} \tag{3.5}$$

$$\text{such that} \begin{cases} T \leq n, \\ x_1 + x_2 + ... + x_n = T, \\ x_i = 1. \end{cases}$$

There are several feature selection measures, which can be represented by the form (3.4), such as the correlation-feature-selection (CFS) measure, the minimal-redundancy-maximal-relevance (mRMR) measure, the Mahalanobis distance and so on.

Correlation Feature Selection Measure: As discribed in the Section 2.1.1.2, 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"* [59]. The following equation gives the merit of a feature subset S consisting of k features:

$$Merit_{S_k} = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}}$$

Here, $\overline{r_{cf}}$ is the average value of all feature-classification correlations, and $\overline{r_{ff}}$ is the average value of all feature-feature correlations. The CFS criterion is defined as follows:

$$\max_{S_k} \Big[\frac{r_{cf_1} + r_{cf_2} + ... + r_{cf_k}}{\sqrt{k + 2(r_{f_1f_2} + .. + r_{f_if_j} + .. + r_{f_kf_1})}}\Big] \tag{3.6}$$

Suppose that there are n full-set features. We use binary values of the variable x_i 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 set. We denote the correlation values r_{cf_i}, $r_{f_if_j}$ by constants a_i, b_{ij}, respectively. Therefore, the problem (3.6) can be described as an optimization problem as follows:

$$\max_{x\in\{0,1\}^n} \Big[\frac{(\sum_{i=1}^{n} a_i x_i)^2}{\sum_{i=1}^{n} x_i + \sum_{i\neq j} 2b_{ij}x_ix_j}\Big] \tag{3.7}$$

It is obvious that the CFS measure is an instance of the GeFS measure. We denote this measure by $GeFS_{CFS}$.

Correlation is used to measure the linear relationship between variables (see Figure 3.2 on the left side for example). However, if the variables are non-linearly related to each others when the data points are distributed as clouds (see Figure 3.2 on the right side for example), using correlation values might fail. In the following, we study a more general measure for non-linear relationship estimation-the mutual information-based measure and its application to feature selection problems.

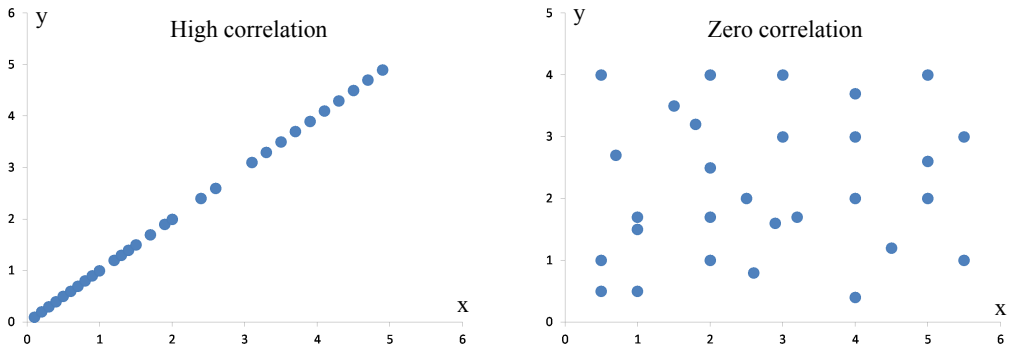


Figure 3.2: Samples of correlation

The mRMR Feature Selection Measure: As discribed in the Section 2.1.1.2, the mRMR [108] is a mutual information based feature selection measure. In this method, relevant features 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 average 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 average value of all mutual information values between the feature f_i and the feature f_j:

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

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

$$\max_S \Big[\frac{1}{|S|}\sum_{f_i \in S} I(f_i;c) - \frac{1}{|S|^2}\sum_{f_i,f_j \in S} I(f_i;f_j)\Big] \tag{3.8}$$

By using binary values of the variable x_i as in the case of the CFS measure to indicate the appearance or the absence of the feature f_i, we can also rewrite the problem (3.8) as an optimization problem as follows:

$$\max_{x \in \{0,1\}^n} \Big[\frac{\sum_{i=1}^n c_i x_i}{\sum_{i=1}^n x_i} - \frac{\sum_{i,j=1}^n a_{ij} x_i x_j}{(\sum_{i=1}^n x_i)^2}\Big] \tag{3.9}$$

It is obvious that the mRMR measure is an instance of the GeFS measure that we denote by $GeFS_{mRMR}$.

The task of feature subset selection by means of the GeFS measure: Suppose that there are n full set features. We need to find a subset of features, which has the maximum value of $GeFS(x)$ over all 2^n possible feature subsets. In other words, we need to solve the following optimization problem:

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

When the number n of features is small, we apply the brute force method to scan all 2^n possible features subsets. But when this number becomes large, the heuristic and random search strategies, such as the best first search or genetic algorithm [107], are usually chosen due to their computational efficiency. Consequently, the given results will always be approximate and unreliable. It is desirable to get the globally optimal subset of features.

In the next subsection, we propose a new method to find these optimal subsets. The main idea is that we consider the optimization problem (3.4) as a polynomial mixed 0-1 fractional programming (P01FP) problem, which will be equivalently converted to a mixed 0-1 linear programming problem and be solved by using the branch and bound algorithm.

3.1.3 Polynomial Mixed 0-1 Fractional Programming

A general polynomial mixed $0-1$ fractional programming ($PM01FP$) problem [33, 34] is represented as follows:

$$\min \sum_{i=1}^{m} \left(\frac{a_i + \sum_{j=1}^{n} a_{ij} \prod_{k\in J} x_k}{b_i + \sum_{j=1}^{n} b_{ij} \prod_{k\in J} x_k} \right) \tag{3.10}$$

(3.11)

$$\text{such that} \begin{cases} b_i + \sum_{j=1}^{n} b_{ij} \prod_{k\in J} x_k > 0, i = \overline{1,m}, \\ c_p + \sum_{j=1}^{n} c_{pj} \prod_{k\in J} x_k \leq 0, p = \overline{1,m}, \\ x_k \in \{0,1\}, k \in J; a_i, b_i, c_p, a_{ij}, b_{ij}, c_{pj} \in \Re. \end{cases}$$

By replacing the denominators in (3.11) by positive variables $y_i (i = \overline{1,m})$, the $PM01FP$ then leads to the following equivalent polynomial mixed $0-1$ programming problem:

$$\min \sum_{i=1}^{m} \left(a_i y_i + \sum_{j=1}^{n} a_{ij} \prod_{k\in J} x_k y_i \right) \tag{3.12}$$

(3.13)

$$\text{such that} \begin{cases} b_i y_i + \sum_{j=1}^{n} b_{ij} \prod_{k\in J} x_k y_i = 1; y_i > 0, \\ c_p + \sum_{j=1}^{n} c_{pj} \prod_{k\in J} x_k \leq 0, p = \overline{1,m}, \\ x_k \in \{0,1\}; a_i, b_i, c_p, a_{ij}, b_{ij}, c_{pj} \in \Re. \end{cases} \tag{3.14}$$

(3.15)

In order to solve this problem, Chang [33, 34] has 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. Based on this technique, the $PM01FP$ becomes then a mixed $0-1$ linear programming ($M01LP$), which can be solved by means of the branch-and-bound method to obtain the global solution.

Proposition 3.1 (Chang's method for linearization)
A polynomial mixed $0-1$ *term* $\prod_{k\in J} x_k y_i$ *from* (3.13) *can be represented by the following program [33, 34], where* C *is a large positive value:*

$$\min z_i$$
$$\text{such that} \begin{cases} z_i \geq 0, \\ z_i \geq C(\sum_{k\in J} x_k - |J|) + y_i \end{cases} \tag{3.16}$$

Proposition 3.2
A polynomial mixed $0-1$ *term* $\prod_{k\in J} x_k y_i$ *from* (3.15) *can be represented by a continuous variable* v_i*, subject to the following linear inequalities [33, 34], where* C *is a large positive value:*

$$\begin{cases} v_i \geq C(\sum_{k\in J} x_k - |J|) + y_i, \\ v_i \leq C(|J| - \sum_{k\in J} x_k) + y_i, \\ 0 \leq v_i \leq C x_k, k \in J. \end{cases} \tag{3.17}$$

We now formulate the feature selection problem (3.4) as a polynomial mixed $0-1$ fractional programming ($PM01FP$) problem.

Proposition 3.3 (GeFS via optimization)
The feature selection problem (3.4) *is a polynomial mixed* $0-1$ *fractional programming (*$PM01FP$*) problem.*

Remark: By applying Chang's method [33, 34], we can transform this $PM01FP$ problem into an $M01LP$ problem. The number of variables and constraints is quadratic in the number n of full set features. This is because the number of terms $x_i x_j$ in (3.4), which are replaced by the new variables, is $n(n+1)/2$. The branch-and-bound algorithm can then be utilized to solve this $M01LP$ problem. But the efficiency of the method depends strongly on the number of variables and constraints. The larger the number of variables and constraints an $M01LP$ problem has, the more complicated the branch-and-bound algorithm is.

In the next subsection, we present an improvement of the Chang's method to get an $M01LP$ problem in which the number of variables and constraints is linear in the number n of full set features.

3.1.4 Optimization of the GeFS measure[3]

By introducing an additional positive variable, denoted by y, we now consider the following problem equivalent to (3.4):

$$\min_{x\in\{0,1\}^n} (-GeFS(x)) = -a_0 y - \sum_{i=1}^{n} A_i(x) x_i y \tag{3.18}$$

[3]This section is published under the title:

Nguyen, H. T., Franke, K., and Petrović, S. Optimizing a class of feature selection measures. *In NIPS 2009 Workshop on Discrete Optimization in Machine Learning: Submodularity, Sparsity & Polyhedra (DISCML)* (2009).

$$\text{such that } \{ \ b_0 y + \sum_{i=1}^{n} B_i(x) x_i y = 1; y > 0. \tag{3.19}$$

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

Proposition 3.4 (Optimization of the GeFS Measure)
A term $A_i(x)x_iy$ from (3.18) can be represented by the following program, where C is a large positive value:

$$\begin{aligned} &\min z_i \\ &\text{such that} \begin{cases} z_i \geq 0, \\ \\ z_i \geq C(x_i - 1) + A_i(x)y, \end{cases} \end{aligned} \tag{3.20}$$

PROOF

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

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

Therefore, the above program on z_i reduces to:

$$\min z_i = \begin{cases} 0, & if \quad x_i = 0, \\ \\ A_i(x)y, & if \quad x_i = 1. \end{cases}$$

which is the same as $A_i(x)x_i = \min z_i$.

Proposition 3.5
A term $B_i(x)x_iy$ from (3.19) can be represented by a continuous variable v_i, subject to the following linear inequality constraints, where C is a large positive value:

$$\begin{cases} v_i \geq C(x_i - 1) + B_i(x)y, \\ \\ v_i \leq C(1 - x_i) + B_i(x)y, \\ \\ 0 \leq v_i \leq Cx_i \end{cases} \tag{3.21}$$

PROOF

(a) If $x_i = 0$, then the constraints become

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

v_i is forced to be zero, because C is a large positive value.

(b) If $x_i = 1$, then the constraints become

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

v_i is forced to be $B_i(x)y$, because C is a large positive value.

Therefore, the constraints on v_i reduce to:

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

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

We substitute each term x_iy that will appear in (3.20),(3.21) by new variables t_i satisfying constraints from Proposition 3.2. The total number of variables for the $M01LP$ problem will be $4n + 1$, as they are x_i, y, t_i, z_i and $v_i(i = \overline{1,n})$. Therefore, the number of constraints on these variables will also be a linear function of n. As we mentioned above, with Chang's method [33, 34] the number of variables and constraints depends on the square of n. Thus our new method actually improves Chang's method by reducing the complexity of the branch and bound algorithm.

We now present a new methodology for determining appropriate instances of the GeFS measure as well as a new search strategy for obtaining subsets of relevant features by means of this measure. The new methodology requires a statistical analysis of properties of the given dataset, i.e. computation of correlation values between features. Depending on these correlation values, the correlation-based or mutual information-based feature selection methods will be selected. In fact, we choose the $GeFS_{CFS}$ measure if the dataset has many features that are linearly correlated to the class label and to each other. Otherwise, the $GeFS_{mRMR}$ measure is selected.

However, before doing this analysis, a visualization of data points into two dimensional space is necessary [19]. The reason is that even though the correlation value between two features is high, their relation can still be non-linear. The Anscombe's quartet [19] illustrates this phenomenon. In fact, in Figure 3.3 we have four different cases of the relation between variables x and y (the dataset is given in Appendix). The correlation value between x and y is 0.816 in each case, which is a very high value. The fourth example (bottom right) shows that one outlier is enough to produce a high correlation value, even though the relationship between two variables is not linear.

The new search strategy for relevant features by means of the $GeFS$ measure is summazied in Algorithm 3.1.

In summary, this section first has analyzed the main factors that affect the reliability in the feature-selection process: the choice of feature-selection methods and the search strategies for relevant features. A formal definition of a reliable feature-selection process were introduced. The definition provides formal measurements of reliability in feature-selection, i.e., the **steadiness** of a classifier's performance and the **consistency** in search for relevant features. Second, this section has proposed new methods to address the main causes of unreliable feature-selection process. In particular, this section has introduced a new methodology of determining appropriate instances from a class of feature-selection methods. This

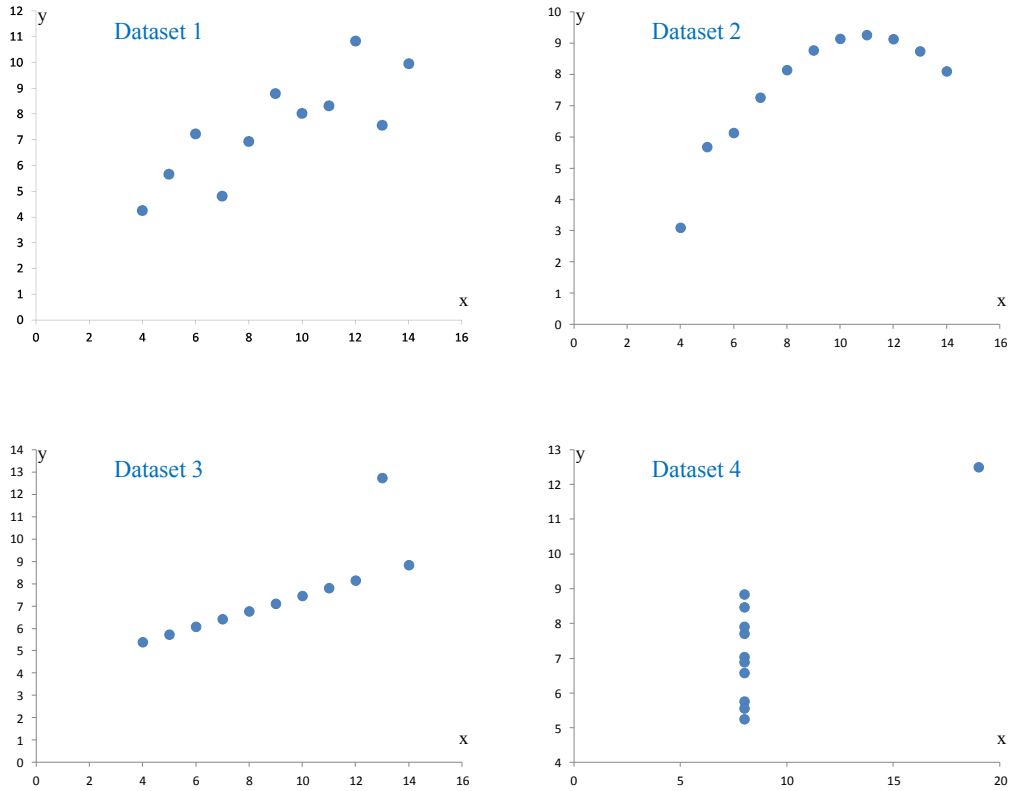


Figure 3.3: Graphs of the Anscombe's quartet.

Algorithm 3.1: The search strategy for relevant features by the GeFS measure.

1. Visualize the dataset into two dimensional space to see whether the relationships between features are linear or non-linear.
2. Analyze the statistical properties of the given dataset in order to choose the appropriate feature selection instance ($GeFS_{CFS}$ or $GeFS_{mRMR}$) from the generic feature selection measure GeFS. We choose the $GeFS_{CFS}$ measure if the dataset has many features that are linearly correlated to the class label and to each other. Otherwise, the $GeFS_{mRMR}$ measure is selected.
3. According to the choice of feature selection instance from *Step 1*, we construct the optimization problem 3.4 for the $GeFS_{CFS}$ measure or for the $GeFS_{mRMR}$ measure. In this step, we can use expert knowledge by assigning the value 1 to the variable if the feature is relevant and the value 0 otherwise.
4. Transform the optimization problem of the GeFS measure to a mixed 0-1 linear programming (M01LP) problem, which is to be solved by means of the branch-and-bound algorithm. A non-zero integer value of x_i from the optimal solution x indicates the relevance of the feature f_i regarding the GeFS measure.

class is called a generic-feature-selection (GeFS) measure. This section has also proposed a new search approach that ensures the globally optimal feature subset by means of the GeFS

measure. The new search approach is based on solving a mixed 0-1 linear programming (M01LP) problem by means of the branch-and-bound algorithm.

So far, we have studied the reliable feature selection methods for the dataset, which has representative linear or non-linear relationships between features. However, in many cases looking at only one statistical relationship is not sufficient to select relevant features, because the dataset can have a mixture of these statistical properties. In the next section, we propose a new ensemble feature selection framework that allows to consider many different statistical properties of a given data set at the same time by combining many feature selection methods used in the filter model. Therefore, the new proposed framework provides much more reliable feature selection results.

3.2 A New Reliable Ensemble Feature-Selection Framework[4]

A typical problem when utilizing filter model for selecting features is the *over-selecting* phenomenon. From the Figure 3.1, in the classification phase, the feature measurement module picks a part of testing data under consideration based on the measured features from the training phase. This assumes that the selected features from the training phase will be representative for the testing data. In fact, this is not always true. The phenomenon happening when the selected features from the training phase are quite different from the representative features of the testing data is called *over-selecting*. In this case, the feature selection results are unreliable and the testing patterns are quite different from the patterns that the system has learned, thus the classification process might perform poorly. Since the model is adapted to another data distribution and might become overfitted, the *over-selecting* phenomenon might imply the *over-fitting* phenomenon of learning algorithms as well.

There are three main causes of the *over-selecting* phenomenon that lead to unreliable feature selection results: (i) non-comprehensive consideration of statistical properties of training data, (ii) heuristic search strategies for feature selection and (iii) small sample size of the data set for training.

Add i. The first cause is that we usually use only one feature selection method for selecting features from a given data set. Since each feature selection measure is considered a representation of a statistical property of the data set, the feature selection results are not general at all, because a data set usually has a set of statistical properties. Therefore, in many cases we take into account the most representative property of the training data and neglect the less representative ones, which might appear as the most representative properties in the testing data. Consequently, the selected features from the training data and the representative features of the testing data are different. It is necessary to consider as many statistical properties as the data has.

Add ii. The second cause of over-selecting is that many feature selection methods use heuristic search strategies, such as genetic algorithms, to find locally optimal feature subsets. Therefore, in the case of many local optima, with different settings of heuristic search methods, different locally optimal feature subsets may be selected. For example, when using genetic algorithm for selecting the best feature subsets by means of the correlation-feature-selection (CFS) measure [59], with different initial populations or with different probabilities of mutation, different solutions may be found. Since each selected feature subset can give a good accuracy of a classifier, domain experts might confuse on choosing the best one for the feature measurement module in the classification phase. Thus the

[4]This section is published under the title:

Nguyen, H. T., Franke, K., and Petrović, S. A new ensemble-feature-selection framework for intrusion detection. In 11th *International Conference on Intelligent Systems Design and Applications (ISDA)*, Spain, 2011, pp. 213-218.

finally selected feature subset might be quite different from the representative one of the testing data.

Add iii. The third reason of the *over-selecting* phenomenon is the small sample size of a data set for training. In this case, not all the statistical properties are presented in the training data set.

In this section, we propose a new framework to address the principal causes of the over-selecting phenomenon: (1) non-comprehensive consideration of statistical properties of training data and (2) heuristic search strategies for feature selection. Next section will describe a new general Lp-norm Support Vector Machines to deal with small datasets.

Add 1. Our new framework that we call Reliable Ensemble Feature Selection measure (EnFS), allows to consider many statistical properties of a given data set at the same time by combining many feature selection methods used in the filter model. Unlike the ensemble feature selection techniques proposed by Saeys et. al. in [120] and Tuv in [56], with our approach, a new formula of feature selection score is constructed from chosen feature selection measures for combining.

Add 2. We also propose a new search algorithm that ensures the globally optimal feature subset by means of the constructed measure. The new search approach is based on solving a mixed 0-1 linear programming (M01LP) problem by means of the branch-and-bound algorithm. In this M01LP problem, the number of constraints and variables is linear in the number of full set features.

3.2.1 Reliable Ensemble Feature-selection Measure

This subsection introduces the definition of a new reliable ensemble feature-selection measure.

Definition 3.6 (Ensemble feature-selection measure)
An ensemble-feature-selection measure ($EnFS$) is defined as a linear combination of m instances of the generic-feature-selection measure 3.3 as follows:

$$EnFS(x) = \frac{1}{m}\sum_{k=1}^{m}\frac{a_{0k} + \sum_{i=1}^{n} A_{ik}(x)x_i}{b_{0k} + \sum_{i=1}^{n} B_{ik}(x)x_i} \tag{3.22}$$

Definition 3.7 (Optimization of ensemble feature-selection measure)
The problem of feature selection by means of the ensemble-feature-selection measure is to find x that maximizes the function $EnFS(x)$:

$$\max_{x\in\{0,1\}^n} EnFS(x) = \frac{1}{m}\sum_{k=1}^{m}\frac{a_{0k} + \sum_{i=1}^{n} A_{ik}(x)x_i}{b_{0k} + \sum_{i=1}^{n} B_{ik}(x)x_i} \tag{3.23}$$

Among m chosen instances, some of them can be univariate methods used in the filter model, such as information gain, distance measures [56], etc. The others can be multivariate methods, such as the correlation-feature-selection (CFS) measure [59], the minimal-redundancy-maximal-relevance ($mRMR$) measure [108], etc. This framework is able to consider many feature selection methods at the same time while keeping the spirit of the filter model by maximizing the defined score $EnFS(x)$.

To get more reliable results, if domain experts want to select T features from n full-set features and want a particular important feature, for example x_i, to appear in the selected feature set, then the problem of feature selection by means of the ensemble-feature-selection measure ($EnFS$) is defined as follows:

$$\max_{x\in\{0,1\}^n} EnFS(x) = \frac{1}{m}\sum_{k=1}^{m}\frac{a_{0k}+\sum_{i=1}^{n}A_{ik}(x)x_i}{b_{0k}+\sum_{i=1}^{n}B_{ik}(x)x_i} \tag{3.24}$$

$$\text{such that}\begin{cases} T \leq n, \\ x_1 + x_2 + ... + x_n = T, \\ x_i = 1. \end{cases}$$

In the next subsection, we consider the optimization problems (3.23), (3.24) as polynomial mixed $0-1$ fractional programming ($P01FP$) problems and show how to solve them.

Proposition 3.6 (EnFS via optimization)
The feature selection problem (3.23) *or* (3.24) *is a polynomial mixed* $0-1$ *fractional programming* ($PM01FP$) *problem.*

Remark: By applying Chang's method [33, 34], this $PM01FP$ problem can be transformed into an $M01LP$ problem. The number of variables and constraints is quadratic in the number n of full set features. This is because the number of terms x_ix_j in (3.23), which are replaced by the new variables, is $n(n+1)/2$. The branch-and-bound algorithm can then be used to solve this $M01LP$ problem. But the efficiency of the method depends strongly on the number of variables and constraints. The larger the number of variables and constraints an $M01LP$ problem has, the more complicated the branch-and-bound algorithm is.

In the next subsection, we present an improvement of the Chang's method to get an $M01LP$ with a linear number of variables and constraints in the number of full set variables. We also give a new search strategy to obtain the relevant subsets of features by means of the $EnFS$ measure.

3.2.2 Optimization of the EnFS measure

By introducing additional positive variables, denoted by $y_k (k = 1, \ldots, m)$, the following problem equivalent to (3.23) is considered:

$$\min_{x\in\{0,1\}^n} (-EnFS(x)) = -\frac{1}{m}\sum_{k=1}^{m}\left(a_{0k}y_k + \sum_{i=1}^{n}A_{ik}(x)x_iy_k\right) \tag{3.25}$$

$$\text{such that}\begin{cases} y_k > 0, k = 1, \ldots, m \\ b_{0k}y_k + \sum_{i=1}^{n}B_{ik}(x)x_iy_k = 1 \end{cases} \tag{3.26}$$

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

Proposition 3.7 (Optimization of the EnFS measure)
A term $A_{ik}(x)x_iy_k$ *from* (3.25) *can be represented by the following program, where* M *is a large positive value:*

$$\min z_{ik}$$

$$\text{such that}\begin{cases} z_{ik} \geq 0, \\ z_{ik} \geq M(x_i - 1) + A_{ik}(x)y_k, \end{cases} \tag{3.27}$$

PROOF The proof of this proposition is in the same line as in the proposition 3.4. The main idea is to consider two cases when $x_i = 0$ and $x_i = 1$, thus the program $\min z_{ik}$ reduces to:

$$\min z_{ik} = \begin{cases} 0, & if \quad x_i = 0, \\ A_{ik}(x)y_k, & if \quad x_i = 1. \end{cases}$$

which is the same as $A_{ik}(x)x_iy_k = \min z_{ik}$. ■

Proposition 3.8
A term $B_{ik}(x)x_iy_k$ from (3.26) can be represented by a continuous variable v_{ik}, subject to the following linear inequality constraints, where M is a large positive value:

$$\begin{cases} v_{ik} \geq M(x_i - 1) + B_{ik}(x)y_k, \\ v_{ik} \leq M(1 - x_i) + B_{ik}(x)y_k, \\ 0 \leq v_{ik} \leq Mx_i \end{cases} \tag{3.28}$$

PROOF The proof of this proposition is in the same line as in the proposition 3.5. The main idea is to consider two cases when $x_i = 0$ and $x_i = 1$, thus the constraints on v_{ik} reduce to:

$$v_{ik} = \begin{cases} 0, & if \quad x_i = 0, \\ B_{ik}(x)y_k, & if \quad x_i = 1. \end{cases}$$

which is the same as $B_{ik}(x)x_iy_k = v_{ik}$. ■

Each term x_iy_k in (3.27), (3.28) is substituted by a new variable t_{ik} satisfying constraints from Proposition 3.2. Then the total number of variables for the $M01LP$ problem will be $\big((3m + 1)n + m\big)$, as they are x_i, y_k, t_{ik}, z_{ik} and $v_{ik}(i = 1, \ldots, n; k = 1, \ldots, m)$. Therefore, the number of constraints on these variables will also be a linear function of n. As we mentioned above, with Chang's method [33, 34] the number of variables and constraints depends on the square of n. Thus our method actually improves Chang's method by reducing the complexity of the branch and bound algorithm.

Remark: To solve the optimization problem (3.24), we can apply the same technique for solving the problem (3.23).

In summary, this section has studied a phenomenon in feature selection process that affects the reliable feature selection results: the *over-selecting* phenomenon. A feature selection method is required to be general enough to find representative features from training data, which are then used for classifying test patterns. The situation where the features selected from the training data are quite different from the representative features of the testing data is called *over-selecting*. The main causes of the *over-selecting* phenomenon are: non-comprehensive consideration of statistical properties of the training data, heuristic search strategies for feature selection and small sample size of the data set for training. In this section, we have shown the influence of the *over-selecting* phenomenon on the *over-fitting* phenomenon of machine learning algorithms. We have proposed a new framework to address principal causes of *over-selecting*, thus reducing the chance of *over-fitting* and providing reliable results. Our new framework that we call Ensemble Feature Selection measure

(EnFS), allows to consider many statistical properties of a given data set at the same time by combining many feature selection methods used in the filter model. From the chosen feature selection measures, a new combined measure is constructed. We have also proposed a new search algorithm that ensures the globally optimal feature subset by means of the constructed measure. Similar to the case of the generic feature selection measure, this new search approach is based on solving a mixed 0-1 linear programming (M01LP) problem by means of the branch-and-bound algorithm.

In the next sections, we first analyze the limitations of the new GeFS and the EnFS methods for feature selection in some special cases, such as when the dataset is quite small or highly imbalanced. We then propose new methods to overcome these limitations.

3.2.3 The limitations of the GeFS and the EnFS methods for feature selection

The Generic Feature Selection (GeFS) measure and the Ensemble Feature Selection (EnFS) framework may fail if the dataset is (1) highly imbalanced or (2) quite small in size, but (3) has many features. We give a more detailed analysis as below.

Add 1. For the case of highly imbalanced datasets, the portion of instances of a class is much larger than the other ones. This is a typical problem in many application domains, such as in network security. The number of normal communication traffic patterns observed in a network exceeds the number of attack traffic patterns. When we calculate the correlation value between a feature X and the class label Y ($corr(X,Y)$, see the calculation below), the $Portion\ A$ of attacks might not affect much to the value $cor(X,Y)$. That means only the $Portion\ N$ of normal instances will contribute to estimate relevance and redundancy of features. It might happen that two features X_1, X_2, which have the same portion N but have quite different portion A, provide the same high correlation values $cor(X_1,Y) = cor(X_2,Y)$ and one of them will be selected due to redundancy. But it turns out that both of them are needed to differentiate types of attacks.

The correlation value between two random variables X and Y is defined as follows:

$$cor(X,Y) = \frac{cov(X,Y)}{\delta_X \delta_Y} = \frac{E(X-\mu_X)(Y-\mu_Y)}{\delta_X \delta_Y} = \frac{\sum_{i=1}^{m}(x_i-\mu_X)(y_i-\mu_Y)}{\delta_X \delta_Y}$$

If we normalize the dataset so that $\mu_X = \mu_Y = 0; \delta_X = \delta_Y = 1$, then the correlation value is computed as follows:

$$cor(X,Y) = \sum_{i=1}^{m}(x_i y_i) = \overbrace{\sum_{i=1}^{Normal}(x_i y_i)}^{Portion\ N} + \underbrace{\sum_{i=1}^{Attack}(x_i y_i)}_{Portion\ A}$$

Add 2. When the dataset is small, it might happen that there are not enough instances to exactly estimate the joint probabilities $p(x,y)$ between features in the formula 3.29 of mutual information computation (see below). Therefore, the mutual information-based feature selection method cannot be utilized to select features.

$$I(X,Y) = \sum_{y\in Y}\sum_{x\in X} p(x,y) log(\frac{p(x,y)}{p(x)p(y)}) \tag{3.29}$$

Add 3. To select features in highly imbalanced and small datasets with many features, the wrapper and embedded methods, which use a classifier's performance as a measurement, are a good choice. The reason is that the selected features contribute mostly to differentiate (classify) instances in the datasets. It is also the main objective of a feature selection

process. In the next section, we focus on the one of the most popular embedded methods for feature selection as well as for classification task: Lp-norm (p=1 or 2) Support Vector Machines.

3.3 General Lp-norm Support Vector Machines[5]

In this section, we propose a new general Lp-norm Support Vector Machine, which is particularly useful in combat with small training datasets.

When the training dataset is small, the distribution of the target variable, which the model is trying to predict, might likely be changed in the testing data, thus leading to over-fitting phenomenon and unreliable classification results. In this case, it is worth selecting few important features for building compact and simple models to reduce the chance of over-fitting in the future. The filter model for feature selection might not be a good choice, since there are not enough samples to estimate statistical properties of the dataset. The wrapper and embedded models use the performance of a machine learning classifier, which is a good criterion to select important features and at the same time to keep the model as accurate as possible. Due to the computational efficiency when dealing with small datasets, several machine learning algorithms were applied in previous work, such as neural networks (see, for example, [61, 149]). In this section, we focus on the embedded feature selection model that is based on the support vector machine (SVM) to cope with small datasets. In fact, we propose a new general Lp-norm SVM (GLp-SVM) with a formal proof that the GLp-SVM selects a lower number of important features while providing better and more reliable accuracy than the traditional Lp-SVM. According to Occam' razor principle [45] saying that "other things being equal, a simpler explanation is better than a more complex one", thus our new GLp-SVM is to be preferred because the simplicity, which is understood as the number of features being used for building the model, is desirable in itself. Below we first analyze the drawback of the traditional Lp-norm SVMs.

The traditional Lp-norm Support Vector Machines (Lp-SVMs, with $p = 1$ or $p = 2$) were studied in many previous work and were demonstrated to be efficient in solving a broad range of different practical problems (see, for example, [30, 40, 57, 83, 140, 91, 113, 134, 151, 138]). As other machine learning classifiers, the performance of Lp-SVMs strongly depends on the quality of features from a dataset. The existence of irrelevant and redundant features in the dataset can reduce the accuracy of Lp-SVMs. We realize that the traditional Lp-SVMs do not comprehensively consider irrelevant and redundant features. In fact, the Lp-SVMs consider all n full-set features to be important for training. However, there probably exist irrelevant and redundant features among n full-set features. Furthermore, in many cases L2-SVM was shown not to select any features [30]. The L1-SVM provides some relevant features, but it cannot remove redundant features [30]. In order to improve the performance of the Lp-SVMs by looking at important features, it is necessary to test all 2^n possible combinations of features for training. In this section, we generalize the Lp-SVMs into a new general Lp-norm Support Vector Machine (GLp-SVM) that takes into account all 2^n possible feature subsets. In the formulation of the GLp-SVM, we encode the data matrix by using the binary variables x_k ($k = \overline{1,n}$) to indicating the appearance of the k^{th} feature ($x_k = 1$) or the absence of the k^{th} feature ($x_k = 0$). Following this proposed encoding scheme, our GLp-SVM can be represented as a mixed 0-1 nonlinear programming problem (M01NLP). The objective function of this M01NLP is a sum of the inverse value of margin by means of Lp-norm and the error penalty. We prove that the minimal value of the objective function from the GLp-SVM is not greater than the one

[5]This section is published under the title:

Nguyen, H. T., Franke, K., and Petrović, S. On general definition of l1-norm support vector machines for feature selection. *International Journal of Machine Learning and Computing* 1, 3 (2011), 279-283.

from the traditional Lp-SVMs. As a consequence, solving our new proposed M01NLP optimization problem results in a smaller error penalty and enlarges the margin between two support vector hyper-planes, thus possibly giving better generalization capability or more reliable classification results of SVM than those obtained by solving the traditional Lp-SVMs. Moreover, by following the new general formulation we can easily integrate expert knowledge into the GLp-SVMs by adding the constraints $x_1+x_2+..+x_n = T, x_i = 1$, where T is the pre-defined number of selected features and x_i is the pre-defined important feature, to the proposed M01NLP optimization problem.

In order to reduce the computational complexity of directly solving the M01NLP problem, we apply Chang's method [33, 34] to equivalently transform it into a mixed 0-1 linear programming (M01LP) problem if $p = 1$ or a mixed 0-1 quadratic programming (M01QP) problem if $p = 2$. The obtained M01LP and M01QP problems can then be efficiently solved by using the branch and bound algorithm.

3.3.1 A General Lp-norm Support Vector Machine

Our goal is to develop a new SVM-based method capable of identifying the best feature subset for classification. To achieve this goal, our first contribution is a novel general formulation of the Lp-norm Support Vector Machines (Lp-SVMs). In the standard Lp-SVMs, we are given a training data set D with m instances: $D = \{(a_i, c_i)|a_i \in \mathbb{R}^n, c_i \in \{-1, 1\}\}_{i=1}^m$, where a_i is the i^{th} instance that has n features and c_i is a class label. a_i can be represented 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 the instance a_i.

For the two-class classification problem, a Support Vector Machine (SVM) learns a separating hyper-plane $w \cdot a_i = b$ that maximizes the margin distance $\frac{2}{||w||_p^p}$, where $w = (w_1, w_2, ..., w_n)$ is the weight vector and b is the bias value. The standard form of the SVM is given below [40, 133]:

$$\min_{w,b,\xi} \frac{1}{p}\|w\|_p^p + C\sum_{i=1}^{m}\xi_i,$$

$$\text{such that} \quad \begin{cases} c_i(\sum_{j=1}^{n} a_{ij}w_j - b) \geq 1 - \xi_i, \\ \xi_i \geq 0, i = \overline{1, m}. \end{cases} \tag{3.30}$$

Above, ξ_i is slack variable, which measures the degree of misclassification of the instance a_i, and $C > 0$ is the error penalty parameter; $p = 1$ or $p = 2$. If $p = 1$, then we have the L1-norm support vector machine (L1-SVM) that was first proposed by Bradley and Mangasarian [30]. If $p = 2$, then we have the traditional L2-norm support vector machine (L2-SVM) [40].

From (3.30), we observe that the traditional Lp-norm SVMs consider all n features to be important for training. However, there probably exist irrelevant and redundant features among n features of the dataset [56, 74]. The performance of Lp-SVMs might be reduced because of these features. Therefore, it is necessary to test all 2^n possible feature subsets for training the Lp-SVMs.

When the number of features n is small, we can apply the brute force method to scan all 2^n subsets. However if this number of features becomes large, a more computationally efficient method that also ensures the best feature subset is required. In the following, we first show how to generalize the problem (3.30) into a general Lp-norm SVM (GLp-SVM), which is in fact a mixed 0-1 nonlinear programming (M01NLP) problem. We then describe how to solve this M01NLP optimization problem in order to get globally optimal solution.

Firstly, we use the binary variables x_k ($k = \overline{1,n}$) for indicating the appearance of the k^{th} feature ($x_k = 1$) or the absence of the k^{th} feature ($x_k = 0$) to encode the data vector a_i ($i = \overline{1,m}$) as follows:

$$\begin{cases} a_i = (a_{i1}x_1, a_{i2}x_2, ..., a_{in}x_n). \\ x = (x_1, x_2, ..., x_n) \in \{0,1\}^n. \end{cases} \tag{3.31}$$

With the encoding scheme (3.31), the problem (3.30) can be generalized into the following mixed 0-1 nonlinear programming (M01NLP) problem:

$$\min_{w,\xi,b,x} \frac{1}{p}\|w\|_p^p + C\sum_{i=1}^{m}\xi_i,$$

$$\begin{cases} c_i\big(\sum_{j=1}^{n} a_{ij}w_jx_j - b\big) \geq 1 - \xi_i, \\ \xi_i \geq 0, i = \overline{1,m}; C > 0, \\ x_j \in \{0,1\}, j = \overline{1,n}. \end{cases} \tag{3.32}$$

Proposition 3.9 (Generalization capacity of the GLp-SVMs)
Suppose that $S1$ and $S2$ are the minimal values of the objective functions from (3.30) and (3.32), respectively. The following inequality is true:

$$S2 \leq S1 \tag{3.33}$$

PROOF

It is obvious, since the problem (3.30) is one case of the problem (3.32) when x equals to $(1,1,..,1)$ or $x = (x_1, x_2, .., x_n) = (1,1,..,1)$. □

Remark:

1. As a consequence of the Proposition 3.9, solving the problem (3.32) results in a smaller error penalty and enlarges the margin between two support vector hyper-planes, thus possibly giving a better generalization capability of SVM than solving the traditional Lp-norm SVMs in (3.30).

2. To get more reliable classification results, if domain experts want to select T features from n full-set features and want a particular important feature, for example x_i, to appear in the selected feature set, then the problem of feature selection by means of the GLp-SVM is defined as follows:

$$\min_{w,\xi,b,x} \frac{1}{p}\|w\|_p^p + C\sum_{i=1}^{m}\xi_i,$$

$$\begin{cases} c_i\big(\sum_{j=1}^{n} a_{ij}w_j x_j - b\big) \geq 1 - \xi_i, \\ \xi_i \geq 0, i = \overline{1,m}; C > 0, \\ x_j \in \{0,1\}, j = \overline{1,n}, \\ x_1 + x_2 + ... + x_n = T, \\ x_i = 1. \end{cases} \tag{3.34}$$

3. Normally, we can apply popular optimization techniques, such as the branch and bound algorithm, to directly solve a mixed 0-1 non-linear programming problem. However, with non-linear constraints, the problem (3.32) becomes even more difficult to solve. In the next subsection, we propose a new approach to linearize the constraints in (3.32), thus reducing the computational complexity of solving (3.32).

3.3.2 Optimizing the General Lp-norm Support Vector Machine

The main idea of our new proposed method is to linearize mixed $0-1$ terms $w_j x_j$ in (3.32) by applying Chang's method. In this way, we equivalently transform the optimization problem (3.32) into a mixed 0-1 linear programming (M01LP) problem if $p = 1$ or into a mixed 0-1 quadratic programming (M01QP) problem if $p = 2$. The obtained M01LP and M01QP problems can then be efficiently solved by using the branch and bound algorithm.

Proposition 3.10 (Optimization of the GLp-SVMs)
A mixed $0-1$ term $w_j x_j$ from (3.32) can be represented by a continuous variable z_j, subject to the following linear inequalities [33, 34], where M is a large positive value:

$$\begin{cases} z_j \geq M(x_j - 1) + w_j, \\ z_j \leq M(1 - x_j) + w_j, \\ 0 \leq z_j \leq M x_j. \end{cases} \tag{3.35}$$

PROOF The proof of this proposition is in the same line as in the Proposition 3.5. The main idea is to consider two cases when $x_i = 0$ and $x_i = 1$. Thus the constraints on z_j reduce to:

$$z_i = \begin{cases} 0, & if \quad x_j = 0, \\ w_j, & if \quad x_j = 1. \end{cases}$$

which is the same as $w_j x_j = z_i$.

Remark:

1. All the constraints in (3.32) are now linear. We consider the first case when $p = 1$. We define $w = p - q$ with $p = (p_1, p_2, ..., p_n) \geq 0, q = (q_1, q_2, ..., q_n) \geq 0$ and $e_n^T = (1, 1, ..., 1)$. The problem (3.32) is then equivalent to the following problem [25]:

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

$$\begin{cases} c_i\big(\sum_{j=1}^{n} a_{ij}(p_j - q_j)x_j - b\big) \geq 1 - \xi_i, \\ \xi_i \geq 0, i = \overline{1,m}; C > 0, \\ x_j \in \{0,1\}, \\ p_j, q_j \geq 0, j = \overline{1,n}. \end{cases} \tag{3.36}$$

By applying the Proposition 3.10, we substitute terms $p_j x_j$ and $q_j x_j$ in (3.36) by new variables t_j and v_j, respectively, satisfying linear constraints. By performing this substitution, we in fact transform the problem (3.36) into a mixed 0-1 linear programming (M01LP) problem. The total number of variables for the M01LP problem will be $6n + m + 1$, as they are b, ξ_i, x_j, p_j, q_j, t_j, z_j and $v_j (i = 1, ..., m; j = 1, ..., n)$. Therefore, the number of constraints on these variables will also be a linear function of n. We propose to use the branch and bound algorithm to solve this M01LP problem.

2. If $p = 2$, then it is obvious that the optimization problem (3.30) is equivalent to the following mixed 0-1 quadratic programming problem, which can be solved by using the branch and bound algorithm:

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

$$\begin{cases} c_i\big(\sum_{j=1}^{n} a_{ij} z_j - b\big) \geq 1 - \xi_i, \\ z_j \geq M(x_j - 1) + w_j, \\ z_j \leq M(1 - x_j) + w_j, \\ 0 \leq z_j \leq M x_j, \\ \xi_i \geq 0, i = \overline{1,m}; C > 0, \\ x_j \in \{0,1\}, j = \overline{1,n}. \end{cases} \tag{3.37}$$

In summary, this section has analyzed the popular Lp-norm Support Vector Machines (Lp-SVMs with $p = 1$ or $p = 2$) algorithms. We have realized that the Lp-SVMs do not comprehensively consider irrelevant and redundant features, because the Lp-SVMs consider all n full-set features to be important for training while skipping other $2^n - 1$ possible feature subsets at the same time. In this section, we have generalized the Lp-SVMs into a new general Lp-norm Support Vector Machine (GLp-SVM) that takes into account all 2^n possible feature subsets. We represent the GLp-SVM as a mixed 0-1 nonlinear programming problem (M01NLP). We prove that solving the new proposed M01NLP optimization problem results in a smaller error penalty and enlarges the margin between two support vector hyper-planes, thus possibly giving a better generalization capability of SVMs than solving the traditional Lp-SVMs. Moreover, by following the new formulation we can easily control the sparsity of the GLp-SVM by adding a linear constraint to the proposed M01NLP optimization problem. In order to reduce the computational complexity of directly solving the M01NLP problem, we have proposed to equivalently transform it into a mixed 0-1 linear programming (M01LP) problem if $p = 1$ or into a mixed 0-1 quadratic programming (M01QP) problem if $p = 2$. The M01LP and M01QP problems can then be solved by using the branch and bound algorithm.

Previous sections have introduced the new reliable feature selection methods and new supervised learning algorithms. In the next section, we propose a new reliable unsupervised learning algorithm based on the popular K-means clustering: an optimal and large-scale K-means clustering algorithm.

3.4 An Optimal K-means Clustering[6]

To deal with datasets without labels, there exist several unsupervised learning algorithms [45]. In this section, we focus on the K-means clustering algorithm [79], which is one of the most popular unsupervised learning methods. K-means clustering aims to partition m observations into K clusters in which each observation belongs to the cluster with the nearest mean value. As shown in section 2.1.3, the formal description of the K-means clustering is given below.

Let $S = (a_1, a_2..., a_m)$ be a dataset with m instances, each of which is an $n-$dimensional vector: $a_i = (a_{i1}, .., a_{in}) \in \mathbb{R}^n$. The objective in K-means clustering is to group these instances into K categories $C_1, C_2, ..., C_K$ for a given K, such that the following objective function is minimized:

$$J_K = \sum_{k=1}^{K} \sum_{i \in C_k} (a_i - \mu_k)^2 \tag{3.38}$$

Here μ_k is the mean vector of the instances from the category C_k: $\mu_k = \frac{1}{m_k} \sum_{i \in C_k} a_i$, where m_k is the number of instances in C_k: $m_k = |C_k|$.

Even though many other approaches of the K-means algorithm exist and were successfully applied to many application domains (see, for example, [63, 65, 145]), there are still two open research questions: (1) How to properly initialize the K seeds? and (2) How to efficiently find the optimal K-means clustering algorithm for large-scale problems?

In this section, we propose a new initialization method for K-means clustering that allows to automatically select good initial points as well as the number K of clusters. We also propose a new search method for optimal K-means with $K = 2$. The main ideas are to cast these problems into mixed 0-1 linear programming problems which can be solved by using D.C. programming approach. The details of our new methods are described below.

3.4.1 New Initialization Method for K-means Clustering

To initialize the K-means clustering, the number of clusters K needs to be specified first and then the seed points are selected. Several popular approaches are applied to determine the value of K [110]. For example, we use the visualization to indicate the number of clusters because of its simplicity and explanation possibilities. The value of K can also be equal to the number of clusters or can also be specified by the users. To select the seed points for K-means, several methods are widely applied. For instance, Milligan et.al. [88] proposed to use hierarchical clustering algorithms to find the K initial seeds. In [31], Bradley and Fayyad suggested to use boostrap-type procedure, in which several different subdatasets are clustered by using the K-means. Each clustering then provides a different candidate set of centroids from which the seeds are chosen. Another approach [89], which has been shown to be a very effective method for well-separated groups, is to find initial seeds that

[6]This section is published under the title:

Nguyen, H. T. An optimal k-means clustering and application to intrudion detection. Tech. rep., Norwegian Information Security Laboratory, Gjøvik University College, Norway

are well separated from each others by means of the euclidean distance. Because of the effectiveness, in this section we focus on this method. Following describes the detail of the algorithm.

We realize that this algorithm is a greedy search, thus providing suboptimal and unreliable solutions in general. The question on how to get more appropriate or more reliable initialization algorithm is still open. In the following, we propose a new reliable initialization method for K-means clustering that allows to automatically select good initial points as well as the number K of clusters. The objective is to search for the best subset of initial points that has a maximal value of total distances between them. To do this, we first cast this search problem into a mixed 0-1 linear programming (M01LP) problem which can be solved by using D.C. programming approach. The domain knowledge can also be easily intergrated into this search process by adding an additional constraint to the M01LP problem. The details of our new method are described belows.

Let d_j^i be an euclidean distance between two instances a_i and a_j of the set S. Therefore, the total distance between all instances in the set S is: $d_S = \frac{\sum_{i=1}^m \sum_{j=1,j\neq i}^m d_j^i}{2}$. The task is to find the subset P of the set S that has the maximal value d_P. In the following, by using binary variables, which indicate the appearance ($x_i = 1$) or the absense ($x_i = 0$) of the i^{th} instance in the selected subset P, we formalize the task as a mixed 0-1 linear programming (M01LP) problem which can be solved by using D.C. programming approach.

$$\max_{x\in\{0,1\}^m} \frac{\sum_{i=1}^m (\sum_{j=1,j\neq i}^m d_j^i x_j) x_i}{2\sum_{i=1}^m x_i} \tag{3.39}$$

$$\Leftrightarrow \min_{x,y} [-\sum_{i=1}^m (\sum_{j=1,j\neq i}^m d_j^i x_j) x_i y]$$

$$\text{such that} \quad \begin{cases} 2\sum_{i=1}^m x_i y = 1, \\ x \in \{0,1\}^m, y > 0. \end{cases} \tag{3.40}$$

Proposition 3.11 (New initialization of the K-means)
A polynomial mixed $0-1$ *term* $(\sum_{j=1,j\neq i}^m d_j^i x_j) x_i y$ *from* (3.40) *can be represented by the following program:*

$$\min z_i$$

$$\text{such that} \quad \begin{cases} z_i \geq 0, \\ z_i \geq M(x_i - 1) + (\sum_{j=1,j\neq i}^m d_j^i x_j) y, \end{cases} \tag{3.41}$$

where M *is a large positive value.*

PROOF The proof of this proposition is in the same line as in the proposition 3.4. The main idea is to consider two cases when $x_i = 0$ and $x_i = 1$, thus the program $\min z_i$ reduces to:

$$\min z_i = \begin{cases} 0, \quad if \quad x_i = 0, \\ (\sum_{j=1,j\neq i}^m d_j^i x_j) y, \ if \ x_i = 1. \end{cases}$$

which is the same as $(\sum_{j=1,j\neq i}^m d_j^i x_j) x_i y = \min z_i$.

Proposition 3.12
A polynomial mixed $0-1$ *term* $x_i y$ *from the above constraints can be represented by a continuous variable* v_i*, subject to the following linear inequality constraints:*

$$\begin{cases} v_i \geq M(x_i - 1) + y, \\ v_i \leq M(1 - x_i) + y, \\ 0 \leq v_i \leq M x_i \end{cases} \tag{3.42}$$

where M *is a large positive value.*

PROOF The proof of this proposition is in the same line as in the proposition 3.5. The main idea is to consider two cases when $x_i = 0$ and $x_i = 1$, thus the constraints on v_i reduces to:

$$v_i = \begin{cases} 0, \ if \ x_i = 0, \\ \\ y, \ if \ x_i = 1. \end{cases}$$

which is the same as $x_i y = v_i$.

3.4.2 Optimal 2-means Clustering via a Mixed 0-1 Programming Problem

It has been shown that the K-means clustering problem is an $NP-$hard optimization problem, even if K is fixed to 2 [80]. That means there are no algorithms running in polynomial time to find globally optimal K-means clustering. Many heuristic approaches were proposed and were applied to many application domains [63]. However, the question on how to find reliable and optimal K-means clustering for large-scale data is still open.

In this section, we will formalize the K-means clustering problem with $K = 2$ as a mixed 0-1 linear programming problem (M01LPP). The obtained M01LPP can be solved by using the DC programming approach [16, 17].

The K-means method is determined by minimizing the sum of squared errors, where μ_k represents the mean vector of the instances from the cluster C_k, $\mu_k = \frac{1}{m_k} \sum_{i \in C_k} a_i$; $m_k = |C_k|$ is the number of instances in C_k.:

$$\min_{C=(C_1,C_2,..,C_K)} J_K = \sum_{k=1}^{K} \sum_{i \in C_k} (a_i - \mu_k)^2$$

$$\min_{C=(C_1,C_2,..,C_K)} J_K = \sum_{k=1}^{K} \sum_{i \in C_k} (a_i^2 - 2a_i\mu_k + \mu_k^2)$$

$$\min_{C=(C_1,C_2,..,C_K)} J_K = \sum_{k=1}^{K} \sum_{i \in C_k} (a_i^2) + \sum_{k=1}^{K} \sum_{i \in C_k} (-2a_i\mu_k) + \sum_{k=1}^{K} \sum_{i \in C_k} (\mu_k^2)$$

$$\min_{C=(C_1,C_2,..,C_K)} J_K = \sum_{k=1}^{K} \sum_{i \in C_k} (a_i^2) + \sum_{k=1}^{K} \mu_k \sum_{i \in C_k} (-2a_i) + \sum_{k=1}^{K} m_k(\mu_k^2)$$

$$\min_{C=(C_1,C_2,..,C_K)} J_K = \sum_{k=1}^{K} \sum_{i \in C_k} (a_i^2) + \sum_{k=1}^{K} \mu_k \sum_{i \in C_k} (-2a_i) + \sum_{k=1}^{K} m_k(\frac{\sum_{i \in C_k} a_i}{m_k})\mu_k$$

$$\min_{C=(C_1,C_2,..,C_K)} J_K = \sum_{k=1}^{K} \sum_{i \in C_k} (a_i^2) + \sum_{k=1}^{K} \mu_k \sum_{i \in C_k} (-2a_i) + \sum_{k=1}^{K} \mu_k \sum_{i \in C_k} a_i$$

$$\min_{C=(C_1,C_2,..,C_K)} J_K = \sum_{k=1}^{K}\sum_{i\in C_k}(a_i^2) - \sum_{k=1}^{K}\mu_k \sum_{i\in C_k} a_i$$

When $K = 2$ then,

$$\min_{C=(C_1,C_2)} J_K = \sum_{k=1}^{2}\sum_{i\in C_k}(a_i^2) - \sum_{k=1}^{2}\mu_k \sum_{i\in C_k} a_i \tag{3.43}$$

The above (3.43) problem is equivalent to the following problem, since the sum $\sum_{k=1}^{2}\sum_{i\in C_k}(a_i^2)$ is constant:

$$\min_{C=(C_1,C_2)} J_K = -\sum_{k=1}^{2}\mu_k \sum_{i\in C_k} a_i = -\sum_{k=1}^{2}\frac{(\sum_{i\in C_k} a_i)^2}{m_k} \tag{3.44}$$

$$\min_{C=(C_1,C_2)} J_K = -\frac{(\sum_{i\in C_1} a_i)^2}{m_1} - \frac{(\sum_{i\in C_2} a_i)^2}{m_2}$$

We use binary values of the variable x_i in order to indicate the appearance ($x_i = 1$) or the absence ($x_i = 0$) of the instance a_i in the cluster C_1. Therefore, the problem (3.44) is equivalent to the following discrete optimization problem:

$$\min_{x\in\{0,1\}^m}[-\frac{(\sum_{i=1}^{m} a_i x_i)^2}{\sum_{i=1}^{m} x_i} - \frac{(\sum_{i=1}^{m} a_i(1-x_i))^2}{\sum_{i=1}^{m}(1-x_i)}]$$

$$\Leftrightarrow \min_{x\in\{0,1\}^m,y}[-\sum_{i=1}^{m}(\sum_{j=1}^{m} a_j x_j)a_i x_i y_1 - \sum_{i=1}^{m}(\sum_{j=1}^{m} a_j(1-x_j))a_i(1-x_i)y_2]$$

$$\text{such that} \quad \begin{cases} \sum_{i=1}^{m} x_i y_1 = 1, y_1 > 0 \\ \sum_{i=1}^{m}(1-x_i)y_2 = 1, y_2 > 0 \end{cases} \tag{3.45}$$

Proposition 3.13 (Optimization of the K-means)
A polynomial mixed $0-1$ *term* $(\sum_{j=1}^{m} a_j x_j)a_i x_i y_1$ *from* (3.45) *can be represented by the following program:*

$$\min z_i$$

$$\text{such that} \quad \begin{cases} z_i \geq 0, \\ z_i \geq M(x_i - 1) + a_i \sum_{j=1}^{m} a_j x_j y_1, \end{cases} \tag{3.46}$$

where M *is a large positive value.*

PROOF The proof of this proposition is in the same line as in the proposition 3.4. The main idea is to consider two cases when $x_i = 0$ and $x_i = 1$, thus the program $\min z_i$ reduces to:

$$\min z_i = \begin{cases} 0, & if \quad x_i = 0, \\ a_i \sum_{j=1}^{m} a_j x_j y_1, & if \quad x_i = 1. \end{cases}$$

which is the same as $(\sum_{j=1}^{m} a_j x_j)a_i x_i y_1 = \min z_i$.

Proposition 3.14
A polynomial mixed $0-1$ *term* $(\sum_{j=1}^{m} a_j(1-x_j))a_i(1-x_i)y_2$ *from* (3.45) *can be represented by the following program:*

$$\min t_i$$

$$\text{such that} \quad \begin{cases} t_i \geq 0, \\ \\ t_i \geq M(-x_i) + a_i \sum_{j=1}^{m} a_j(1-x_j)y_2, \end{cases} \tag{3.47}$$

where M *is a large positive value.*

The proof of this proposition is in the same line as in the proposition 3.4. The main idea is to consider two cases when $x_i = 0$ and $x_i = 1$, thus the program $\min t_i$ reduces to:

PROOF

$$\min t_i = \begin{cases} 0, \quad if \quad x_i = 1, \\ \\ a_i \sum_{j=1}^{m} a_j(1-x_j)y_2, \quad if \quad x_i = 0. \end{cases}$$

which is the same as $(\sum_{j=1}^{m} a_j x_j)a_i(1-x_i)y_2 = \min t_i$.

Proposition 3.15
A polynomial mixed $0-1$ *term* $x_i y_j$ *from the above constraints can be represented by a continuous variable* v_{ij}*, subject to the following linear inequality constraints:*

$$\begin{cases} v_{ij} \geq M(x_i - 1) + y_j, \\ v_{ij} \leq M(1-x_i) + y_j, \\ 0 \leq v_{ij} \leq M x_i \end{cases} \tag{3.48}$$

where M *is a large positive value.*

PROOF The proof of this proposition is in the same line as in the proposition 3.5. The main idea is to consider two cases when $x_i = 0$ and $x_i = 1$, thus the constraints on v_{ij} reduces to:

$$v_{ij} = \begin{cases} 0, \quad if \quad x_i = 0, \\ \\ y_1, \quad if \quad x_i = 1. \end{cases}$$

which is the same as $x_i y_j = v_{ij}$.

In summary, this section has proposed a new initialization method for K-means clustering that allows to automatically select good initial points as well as the number K of clusters. This section has also proposed a new search method for optimal and large-scale K-means clustering algorithm with $K = 2$. The main idea is to cast these problems into mixed 0-1 linear programming problems which can be solved by using the D.C. programming approach.

Chapter 4

Applications to Intrusion Detection Systems

Theory without practice is empty;
Practice without theory is blind.

JOHN DEWEL

This part describes successful applications of new developed machine learning algorithms to enhance the efficiency, effectiveness and reliability of intrusion detection systems in different layers: application layer, network layer and operating system layer. In particular, as shown in Figure 4.1 we studied Web Application Firewalls [98, 102] for the application layer. We applied the new methods to the Network-based Intrusion Detection Systems [94, 95, 96, 99] and Botnet-malware Detection Systems [26] in the network layer. For the operating system layer, Host-based Intrusion Detection Systems [97, 100] were analyzed. Finally, we show the application of new proposed machine learning algorithms to increase the efficiency and effectiveness of the testing process of intrusion detection systems, e.g. Web Application Firewalls [106].

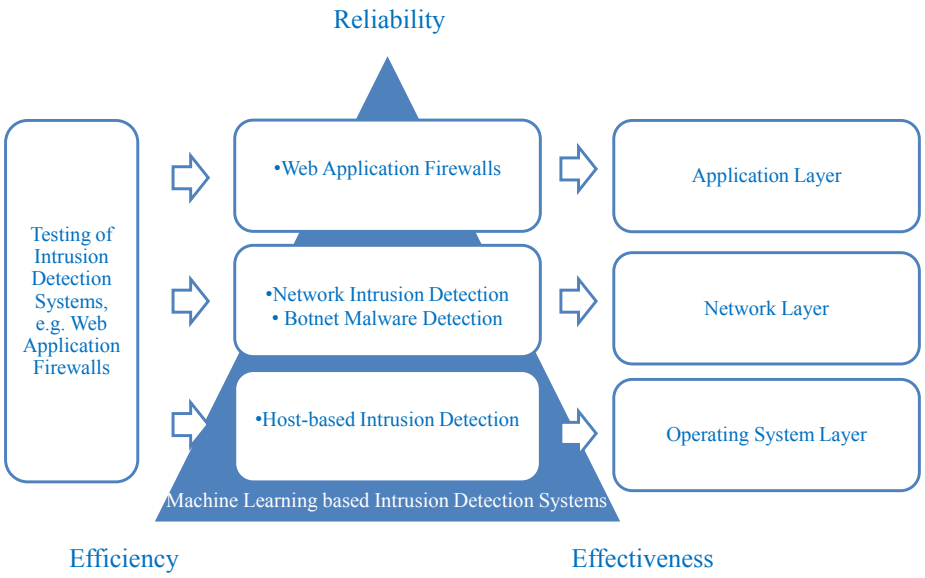


Figure 4.1: Applications to Intrusion Detection Systems

4.1 Web-based Attack Detection[1]

Web attacks pose many serious threats to modern Internet. The number of Web attacks is steadily increasing. Consequently, Web Application Firewalls (WAFs) [23] need to be more and more effective. One of the approaches for improving the effectiveness of WAFs is to apply computational intelligence methods. In fact, it has been shown that these methods enhance the overall performance of many types of intrusion detection systems (see, for example, [38, 95, 118, 129, 144]) and particularly of the WAFs (see, for example, [68, 132, 146]). In this section, we focus only on the feature selection methods for WAFs. Achieving reduction of the number of relevant traffic features without negative effect on detection accuracy is a goal that greatly increases the available processing time of WAFs and reduces the required system resources. As there exist many feature selection algorithms (see for example [56, 74]), the question that arises is which ones can be applied for intrusion detection in general and for Web attack detection in particular. The most of the feature selection work in intrusion detection practice is still done manually and the quality of selected features depends strongly on expert knowledge. For automatic feature selection, the wrapper and the filter models from machine learning are frequently applied [56, 74]. The wrapper model assesses the selected features by learning algorithm's performance. Therefore, the wrapper method requires a lot of time and computational resources to find the best feature subsets. The filter model considers statistical characteristics of a dataset directly without involving any learning algorithms. Due to the computational efficiency, the filter method is usually utilized to select features from high-dimensional datasets, such as intrusion detection systems. Moreover, this method allows to estimate feature subsets not only by their relevance, but also by the relationships between features that make certain features redundant. A major challenge in the IDS feature selection process is to choose appropriate measures that can precisely determine the relevance and the relationship between features of a given dataset.

Since the relevance and the relationship are usually characterized in terms of correlation or mutual information [56, 74], we focus on our new proposed generic feature selection (GeFS) measure for intrusion detection [94] (see Section 3.1.2). This measure consists of two instances that belong to the filter model from machine learning: the correlation feature selection ($GeFS_{CFS}$) measure [59] and the minimal-redundancy-maximal-relevance ($GeFS_{mRMR}$) measure [108]. As described in Section 3.1.2, in a given dataset if there are many features that are linearly correlated to each other, then the $GeFS_{CFS}$ measure is recommended for selecting features. Otherwise, the $GeFS_{mRMR}$ measure is alternatively chosen as it considers non-linear relations through the analysis of mutual information between the features.

In this section, we propose to use the GeFS measure (see subsection 3.1.2) for selecting features in Web attack detection. We conducted experiments on the ECML/PKDD 2007 dataset, which was generated for the ECML/PKDD 2007 Discovery Challenge. However, the attack requests of this dataset did not target any real Web application. Therefore, we additionally generated our new CSIC 2010 dataset, which contains the traffic directed to an e-commerce Web application. From our expert knowledge about Web attacks, we listed 30 features that we considered relevant for the detection process. Then, we extracted the values of these 30 relevant features from the datasets. By applying the GeFS measure, we wanted to know within the particular datasets which features among the 30 extracted features are the most important for the Web attack detection process. In order to do that, we analyzed the statistical properties of the datasets to see whether they had linear correlation or non-linear relations between features. To do that, the data points of the datasets were visualized in the two-dimensional space and the correlation coefficients were computed.

[1] This section is published under the title:

Nguyen, H. T., Torrano-Gimenez, C., Alvarez, G., Franke, K., and Petrović, S. Enhancing the effectiveness of web application firewalls by generic feature selection. *Logic Journal of IGPL* (2012).

We then chose the $GeFS_{CFS}$ measure for feature selection from the CSIC 2010 dataset and the $GeFS_{mRMR}$ measure for the ECML/PKDD 2007 dataset. The detection accuracies obtained after the feature selection by means of four different classifiers were tested.

Furthermore, in this section we validate our new proposed methods in addressing the reliability issue in feature-selection process as described in Section 3.1. The aim is to show that in order to yield the steadiness of a classifier's performance, analyzing the statistical property of a data set before determining appropriate instances of the GeFS measure is necessary. We also show that our new proposed search approach is consistent in search for relevant features, thus providing more reliable feature-selection results than heuristic search strategies do.

The section is organized as follows. Section 4.1.1 describes two datasets for the experiment in more detail: the ECML/PKDD 2007 and the CSIC 2010 HTTP. We also introduce the features from expert knowledge for Web attack detection. Experimental setting is given in Section 4.1.2 and experimental results are discussed in Section 4.1.3.

4.1.1 Datasets

We conducted experiments on the ECML/PKDD 2007 data set, which was generated for the ECML/PKDD 2007 Discovery Challenge [112]. In fact, we utilized the training set, which is composed of 50,000 samples including 20% of attacks (i.e. 10,000 attacks and 40,000 normal requests). The requests are labeled with specifications of attack classes or normal traffic. The classes of attacks in this data set are: Cross-Site Scripting, SQL Injection, LDAP Injection, XPATH Injection, Path traversal, Command Execution and SSI attacks. However, the attack requests of this data set were constructed blindly and did not target any real Web application. Therefore, we additionally generated our new CSIC 2010 data set for experiments.

The CSIC 2010 data set [2] contains traffic targeted to a realistic Web application developed for this purpose. It consists of an e-commerce Web application running on an Apache server and it is composed of several Web pages that allow users to do actions such as buying items with a shopping cart or registering by providing their personal data. Some of the Web pages require certain parameters, for example the user's name and address for the registration process or the name of the product that the user wants to buy.

The traffic of the CSIC 2010 dataset is automatically generated and it contains normal and anomalous requests to all the Web pages composing the e-commerce Web application, with different values for those Web pages including parameters. In total, 36,000 normal requests and more than 25,000 anomalous requests are included in the dataset. As the traffic is generated, all the requests are labeled (either as normal or as anomalous). The HTTP dataset CSIC 2010 includes modern Web attacks such as SQL injection, buffer overflow, information gathering, CRLF injection, XSS, server side include and parameter tampering.

The traffic is generated as explained in the sequel: First, real data is collected for all the parameters of the Web application. All the data (names, surnames, addresses, etc.) is extracted from real databases. These values are stored in two databases: one for the normal values and other for the anomalous ones. Additionally, all the public available pages of the Web application are listed. For every Web page, normal and anomalous requests are generated. In the case that normal requests have parameters, the parameter values are filled out with data taken from the normal database randomly. The process is analogous for anomalous requests, where the values of the parameters are taken randomly from the anomalous database. In Figure 4.2 it can be observed a scheme of the traffic generation process. The dataset is available in [2], where further details about the dataset can also be found.

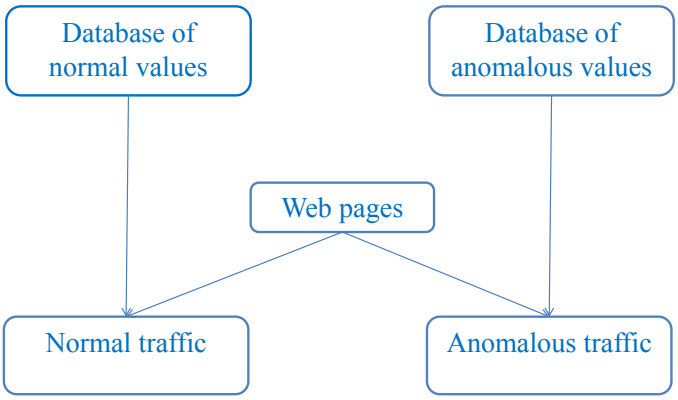


Figure 4.2: Scheme of the HTTP traffic generation process.

Table 4.1: Names of 30 features from expert knowledge for Web attack detection [18]. $\star$ refers to features selected by the $GeFS_{CFS}$ from CSIC-2010 data set; † refers to features selected by the $GeFS_{mRMR}$ from CSIC 2010 data set; $\bullet$ refers to features selected by the $GeFS_{CFS}$ from ECML/PKDD 2007 data set; and $\diamond$ refers to features selected by the $GeFS_{mRMR}$ from ECML/PKDD 2007 data set.

Feature Name	Feature Name
Length of the header "Accept-Charset" †	Length of the path $\star$
Length of the arguments $\star$ $\diamond$	Length of the header "Accept" †
Length of the header "Accept-Encoding" †	Length of the request $\star$ $\diamond$
Length of the header "Accept-Language" †	Length of the header "Cookie" †
Length of the header "Content-Length" †	Length of the header "Content-Type"
Length of the Host †	Length of the header "Referer" †
Length of the header "User-Agent" †	Method identifier
Number of arguments $\star$	Number of letters in the arguments $\star$
Number of digits in the arguments $\star$	Number of distinct bytes
Number of other char in the arguments $\bullet$ $\diamond$	Number of letters char in the path $\star$
Number of digits in the path $\star$ †	Number of 'special' char in the path $\star$
Number of other char in path †	Number of cookies †
Maximum byte value in the request $\star$ †	Minimum byte value in the request $\diamond$
Number of 'special' char in the arguments $\star$ † $\bullet$ $\diamond$	Entropy $\diamond$
Number of keywords in the path	Number of keywords in arguments

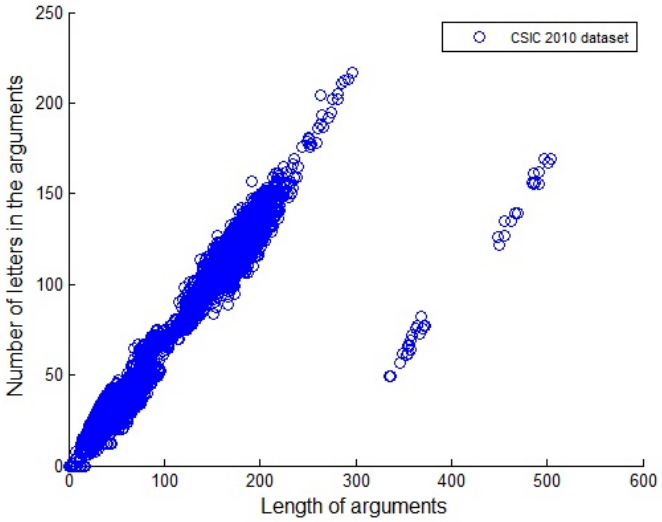


Figure 4.3: Sample of distributions of data points from the CSIC 2010 dataset.

4.1.2 Experimental Setting

By utilizing our expert knowledge about Web attacks, we listed 30 features that we considered relevant for the detection process (see Table 4.1). Some features refer to the length of the request, the length of the path or the headers, as length is important for detecting buffer-overflow attacks. We also observed that the non-alphanumeric characters were present in many injection attacks. Therefore, we took four types of characters into account: letters, digits, non-alphanumeric characters and other characters. As the non-alphanumeric characters have a special meaning in a set of programming languages, in Table 4.1 we refer to them as 'special' char. We analyzed the appearance of the characters in the path and in the argument's values. We also studied the entropy of the bytes in the request. Additionally, we collected the keywords of several programming languages that were often utilized in the injection attacks and counted the number of their appearances in different parts of the request as a feature.

We analyzed statistical properties of the data sets to see whether they had linear or non-linear relations between features. From this analysis, the appropriate feature selection instance from the GeFS measure was chosen for each data set according to Algorithm 3.1 described in Section 3.1. In order to do that, we first visualized the whole data sets in the two-dimensional space to get a plot matrix. In this plot matrix, each element was the distribution of data points depending on the values of a feature and the class label or the values of two features. Figure 4.3 and Figure 4.4 show the sample distributions of data points of the CSIC 2010 data set and the ECML/PKDD 2007 data set, respectively. We then calculated the correlation coefficients between the features. From these, we observed that the CSIC 2010 data set has many features that are linearly correlated to each other, whereas in the ECML/PKDD 2007 data set the non-linear relations between features are more representative. In fact, in the CSIC 2010 data set, more than 63 % of the correlation coefficients are greater than 0.5, whereas in the ECML/PKDD 2007 data set more than 83% of the correlation coefficients are less than 0.09. Therefore, we chose the $GeFS_{CFS}$ measure for selecting features from the CSIC 2010 data set, and the $GeFS_{mRMR}$ measure for selecting features from the ECML/PKDD 2007 data set. Moreover, the $GeFS_{CFS}$ and the $GeFS_{mRMR}$ measures were also applied to the ECML/PKDD 2007 and to the CSIC 2010 data sets, respectively, to see how the wrong choices of feature selection methods would negatively affect the detection performance.

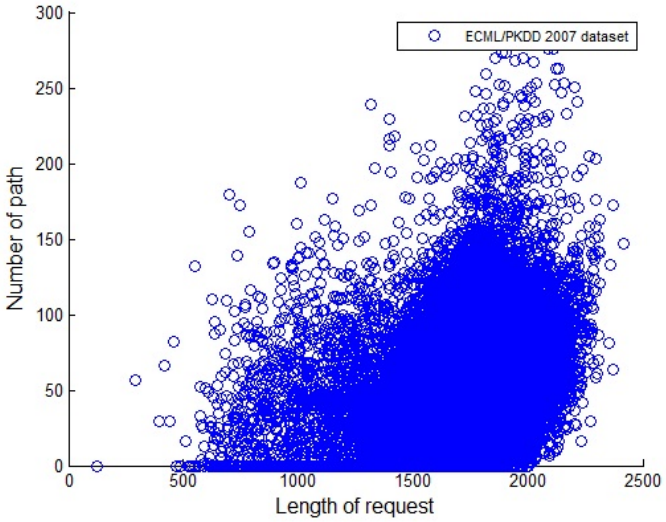


Figure 4.4: Sample of distributions of data points from the ECML/PKDD 2007 data set.

We applied the new proposed search method described in Section 3.1.2 to find globally optimal feature subsets by means of the $GeFS_{CFS}$ and the $GeFS_{mRMR}$ measures. We compared our new global search method with heuristic search strategies in term of reliability. In order to do that, we utilized the genetic search and the Peng's method [108] to select features from the data sets. Four classifiers with 10-fold cross validation were utilized to evaluate detection performances before and after feature selection: C4.5, CART, RandomTree and RandomForest [45]. All the obtained results are listed in the tables 4.2, 4.3 and 4.4.

4.1.3 Experimental Results and Discussion

Table 4.2 shows the number of full-set features and the number of features selected by the $GeFS_{CFS}$ measure and the $GeFS_{mRMR}$ measure (Table 4.1 shows which features were selected). Table 4.3 summarizes the detection rates of four different classifiers performed on the CSIC 2010 data set and the ECML/PKDD 2007 data set. Table 4.4 indicates the consistency and steadiness values of four different feature-selection methods, i.e., the $GeFS_{CFS}$, the $GeFS_{mRMR}$, the GA-CFS (genetic search with the CFS measure) and the mRMR (Peng's method).

Table 4.2: Full-set features and the number of selected features.

Data Set	**Full − set**	$\mathbf{GeFS_{CFS}}$	$\mathbf{GeFS_{mRMR}}$
CSIC 2010	30	11	14
ECML/PKDD 2007	30	2	6

Table 4.3: Detection rates of four different classifiers performed on the CSIC 2010 data set and the ECML/PKDD 2007 data set. The Ge-CFS and Ge-mRMR are notations of $GeFS_{CFS}$ and $GeFS_{mRMR}$, respectively.

Classifiers	**CSIC 2011 data set**			**ECML/PKDD 20007 data set**		
	Full-set	**Ge-CFS**	**Ge-mRMR**	**Full-set**	**Ge-CFS**	**Ge-mRMR**
C4.5	94.49	94.06	79.80	96.37	86.45	91.62
CART	94.12	93.71	79.85	96.11	86.45	91.54
RandomTree	92.30	92.70	71.36	96.89	86.39	93.41
RandomForest	93.71	93.68	71.70	98.80	86.39	95.18
Average	**93.65**	**93.53**	**75.67**	**97.04**	**86.42**	**92.93**

Table 4.4: Consistency and steadiness values of different feature-selection methods.

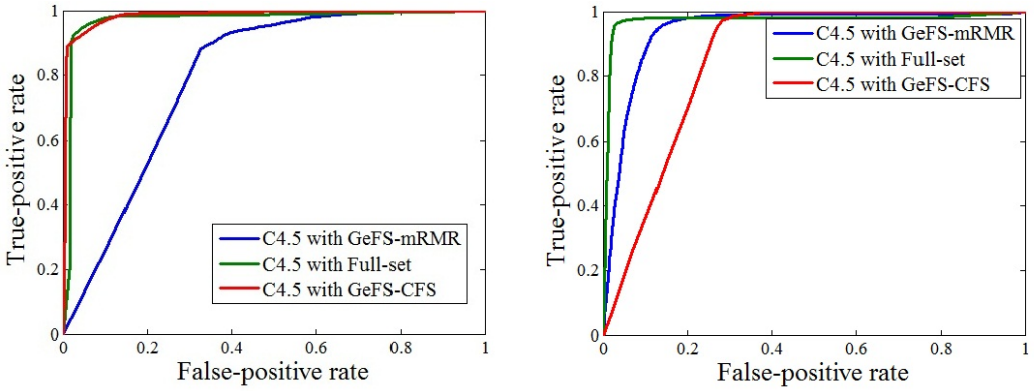
Measurements	**CSIC 2011 data set**			**ECML/PKDD 20007 data set**		
	Ge-CFS	**Ge-mRMR**	**GA-CFS**	**Ge-CFS**	**Ge-mRMR**	**mRMR**
Consistency(%)	100	100	25	100	100	27
Steadiness(%)	99.87	80.80	97.33	89.05	95.76	92.14

It can be observed from Table 4.3 that following our new proposed methodology the choice of appropriate instances of the GeFS measure leads to steady performance of classifiers. In fact, the $GeFS_{CFS}$ measure performed well on the CSIC 2010 data set and provided better results than the $GeFS_{mRMR}$ measure. In more detail, with the $GeFS_{CFS}$ measure we almost keep the detection accuracies with only 0.12% of difference, whereas with the $GeFS_{mRMR}$ measure we reduce detection rates by 17.89%. On the ECML/PKDD 20007 data set, the $GeFS_{mRMR}$ measure provides better results than the $GeFS_{CFS}$ measure.

Following the definitions 3.1 and 3.2, we calculated the consistency and steadiness values of the feature-selection methods. These values are shown in Table 4.4. We observed that our new proposed search method for relevant features is consistent with 100% consistency. The heuristic search strategies provided inconsistent feature-selection results with only 25% consistency. At the same time, the steadiness values generated by the heuristic search strategies are less than the ones obtained by utilizing our global search approach.

Finally, the overall performance, which include the detection performance and run-time performance, of the WAFs has been improved by the appropriate choices of feature selection measures. As shown in Figure 4.5(a), for example, the C4.5-based WAF trained on important features by means of the $GeFS_{CFS}$ measure is comparative with the one trained on the full-set features. At the same time, more than 50% of run-time for training the WAF is reduced.

In summary, this section has proposed to apply the generic feature selection (GeFS) measure for Web attack detection. Statistical properties of the generated CSIC 2010 dataset and the ECML/PKDD 2007 dataset were first analyzed. Based on this analysis, the $GeFS_{CFS}$ measure and the $GeFS_{mRMR}$ measure were chosen for selecting features from the CSIC 2010 dataset and the ECML/PKDD 2007 dataset, respectively. The detection accuracies obtained after the feature selection by means of four different classifiers were tested. The experiments show that by choosing appropriate instances of the GeFS measure, 63% of irrelevant and redundant features were removed from the original dataset, while reducing



(a) CSIC 2010 dataset

(b) ECML/PKDD 2007 dataset

Figure 4.5: Detection performance (ROC curves) of different C4.5 classifiers.

only 0.12% the detection accuracy of WAFs. At the same time, the new proposed methods provide reliable feature-selection results and outperform the heuristic approaches in term of reliability.

In the next section, we introduce the application of our new proposed reliable machine learning algorithms to the network-based intrusion detection systems and to the botnet-malware detection systems in the network layer (see Figure 4.1).

4.2 General Network-based Intrusion Detection[2]

Network-based Intrusion Detection Systems (NIDSs) have become important security tools applied in many contemporary network environments. They gather and analyze information from various sources on hosts and networks in order to identify suspicious activities and generate alerts for an operator. The task of network intrusion detection is often analyzed as a pattern recognition problem-an NIDS has to tell normal from abnormal behaviour. It is also of interest to further classify abnormal behaviour in order to undertake adequate counter-measures. An NIDS can be modeled in various ways (see for example [41], [55]). A model of this kind usually includes a representation algorithm (for representing incoming data in the space of selected features) and a classification algorithm (for mapping the feature vector representation of the incoming data to elements of a certain set of values, e.g. normal or abnormal etc.). Some NIDSs, like the models presented in [55], also include the feature selection algorithm, which determines the features to be used by the representation algorithm. Even if the feature selection algorithm is not included in the model directly, it is always assumed that such an algorithm is run before the very intrusion detection process.

The quality of the feature selection algorithm is one of the most important factors that affect the effectiveness of an NIDS. The goal of the algorithm is to determine the most relevant features of the incoming traffic, whose monitoring would ensure reliable detection of abnormal behaviour. Since the effectiveness of the classification algorithm heavily depends on the number of features, it is of interest to minimize the cardinality of the set of selected

[2]This section is published under the title:

Nguyen, H. T., Franke, K., and Petrović, S. Improving effectiveness of intrusion detection by correlation feature selection. *International Journal of Mobile Computing and Multimedia Communications (IJMCMC)* 3, 1 (2011), 21-34.

features, without dropping potential indicators of abnormal behaviour. Obviously, determining a good set of features is not an easy task. The most of the work in practice is still done manually and the feature selection algorithm depends too much on expert knowledge. Automatic feature selection for reliable intrusion detection remains therefore a great research challenge.

In this section, we consider automatic feature selection procedures from the filter model [56] for network intrusion detection purposes. The filter method directly considers statistical characteristics of the data set, such as correlation between a feature and a class or inter-correlation between features, without involving any learning algorithms. Thus, the filter method is suitable to select features from high-dimensional data sets for NIDSs. We focus on the new proposed generic feature-selection (GeFS) measure from Section 3.1

This section is organized as follows. Subsection 4.2.1 describes the KDD CUP 1999 dataset in more detail. We provide the procedure to extract features from the captured 1998 DARPA dataset. Experimental setting is given in subsection 4.2.2 and experimental results are discussed in subsection 4.2.3. In subsection 4.2.4 we show the comparison of our new proposed algorithm with the existing methods. Finally, we apply the new ensemble-feature-selection framework (see Section 3.2) for network-based intrusion detection systems in subsection 4.2.5.

4.2.1 Datasets

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 [73]. 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 to take place. The generated traffic was similar to that on a government site 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 e-mails, browsed Web sites, sent and received files using FTP or used Telnet to log into remote computers and performed works and so on. More details of the simulated network are described in the sequel. 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 E-mail traffic was generated between the inside PC'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 e-mail message contents were created using statistical bigrams frequencies to preserve word and two-word sequence statistics from a sampling of roughly 10,000 actual E-mail 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

Table 4.5: Sample of network connection records [71]

Timestamp	Duration	Service	src_bytes	Flag
1.1	1	HTTP	10	S0
13.4	60	FTP	1000	SF
15.3	600	SMTP	100	S0

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 Web sites 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 moving to other 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. smurf) 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 or 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 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 gains 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. [71, 72] 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 were then 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 were fist summarized into network connection records, in which a set of intrinsic features from domain knowledge was utilized. Examples of network connection records and the list of intrinsic features are given in Table 4.5 and Table 4.6

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. Then they 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 4.7, there is an example of frequent SYN flood patterns that they found in the audit 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.

The additional features automatically constructed by Lee's proposed method [71] are summarized below:

Table 4.6: Intrinsic features of individual TCP connections [71]

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

Table 4.7: Sample of intrusion pattern [71]

Frequent Episode	Meaning
(flag=S0, service=http,dst_host=victim), (flag=S0, service=http,dst_host=victim) (flag=S0, service=http,dst_host=victim) $\rightarrow$ [0.93, 0.03, 2]	93% of the time, after two http connections with S0 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).

- 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 number 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 4.8 and Table 4.9, 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 [71]: 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 attempts to access control files (e.g., ”/etc/passwd”,

Table 4.8: Traffic features computed using a two-second time window [71]

ID	Feature Name	Description	Type
10	Count	number of connections to the same host in the past two seconds	continuous
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 in the past two seconds	continuous
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 4.9: Traffic features computed using a window of 100 connections [71]

ID	Feature Name	Description	Type
19	dst_host_count	number of connections to the same host in the past two seconds	continuous
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 in the past two seconds	continuous
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

".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 an ftp session. These features are summarized in Table 4.10.

4.2.2 Experimental Setting

This section demonstrates the application of the generic-feature-selection (GeFS) measure for network-based intrusion detection systems. According to Algorithm 3.1 described in Section 3.1, it is necessary to analyze the statistical property of the KDD CUP 1999 dataset to see whether it has linear or non-linear relations between features. From this analysis, an appropriate feature-selection instance from the GeFS measure is chosen. In order to do that, we first visualize the whole KDD CUP 1999 dataset in the two-dimensional space. Figure 4.6 shows the sample distributions of data points of the KDD CUP 1999 dataset. We then calculate the correlation coefficients between the features. From these, we observe that the KDD CUP 1999 dataset has many features that are linearly correlated to each other. In fact, in the KDD CUP 1999 dataset, more than 74 % of the correlation coefficients are greater than 0.5. Therefore, we chose the $GeFS_{CFS}$ measure for selecting features from the KDD

Table 4.10: Content features within a connection suggested by domain knowledge [71]

ID	Feature Name	Description	Type
29	host	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_cmd	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

CUP 1999 dataset.

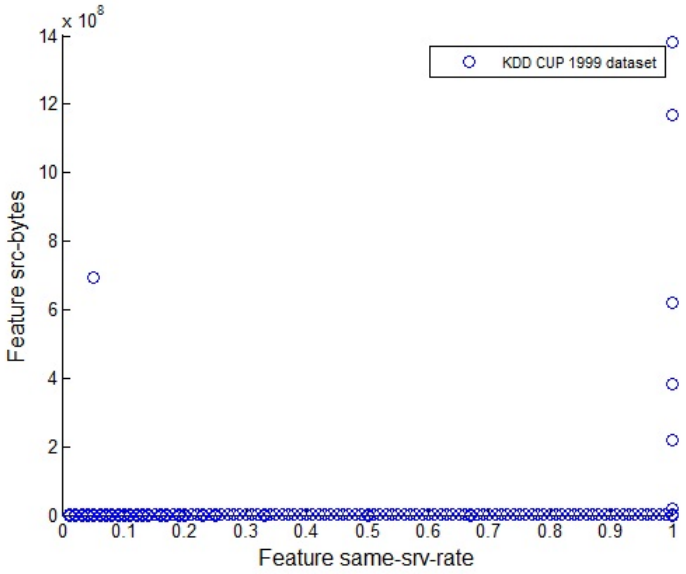


Figure 4.6: Sample of distributions of data points from the KDD CUP 1999 dataset.

For evaluating the performance of our new $GeFS_{CFS}$ approach, two different available feature selection methods based on the CFS measure [36] are implemented. One is the best-first-CFS method, which uses the best first search strategy to find the locally optimal feature subset. The other one uses the genetic algorithm for search. Note that the best first search and genetic algorithm may not guarantee to find the globally optimal solution. However, we can overcome this issue with our new method. We did not choose the exhaustive search method since it is not feasible for feature selection from data sets with a large number of features. We applied machine learning algorithms for evaluating the classification accuracy on selected features, since there is no standard network-based intrusion detection systems.

We performed our experiment using 10% of the overall (5 millions) KDD CUP'99 IDS benchmarking labeled data [72]. This data set described in the previous section contains 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 numbers of instances for the four attack classes and normal class are quite different, e.g the relation of the number of U2R to the number of DoS is $1.3 * 10^{-4}$. Details of the number of class instances are given in Table 4.11 .

Table 4.11: The partition of KDD CUP'99 data set used in the experiment

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%

We tested the performance of our newly proposed $GeFS_{CFS}$ feature selection method in more detail below:

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 used 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, we might miss relevant characteristics of the less represented classes.

2. As the attack classes distribute so differently, we prefered to process these attack classes separately. With the specific application of IDS we can also formulate four different two-class problems. Four classifiers shall be derived using specific features for each classifier in order to detect (identify) a particular attack. The rationale for this approach is that we predict the most accurate classification if each of the four intrusion detectors (classifiers) is fine-tuned according to significant features. This approach might also be very effective, since the four light-weight classifiers can be operated in parallel.

For understanding the effect, as mentioned in $1)$, we conducted a small experiment. The aim was to show that the classifier highly neglected U2R attack instances. In order to do that, we mixed all attack classes to get only one data set and considered a five-class (normal, DoS, Probe, U2R and R2L) problem. The C4.5 machine learning algorithm was used as a classifier. We applied 5-fold cross-validation for evaluating the detection accuracy of the

C4.5. The result of the experiment is given in Table 4.12. It can be seen from Table 4.12 that the C4.5 highly misclassified U2R attack instances with 34.6% error.

Table 4.12: Misclassified instances (UI) by the C4.5 classifier

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%

In order to perform the experiment 2), we added normal traffic into each attack class to get four data sets: Nor&DoS, Nor&Probe, Nor&U2R and Nor&R2L. With each data set, we ran three feature selection algorithms: our new $GeFS_{CFS}$ method, the best-first CFS-based and the genetic algorithm CFS-based methods. The numbers of selected features and their identifications are given in Table 4.13 and Table 4.14, respectively. We then applied the C4.5 and the BayesNet machine learning algorithms on each original full set as well as each newly obtained data set that includes only those selected features from feature selection algorithms. By applying 5-fold cross-validation evaluation on each data set, the classification accuracies and false-positive rates are reported in Table 4.15 and in Table 4.16, respectively.

Our new $GeFS_{CFS}$ method was compared with the best-first-CFS and genetic-algorithm-CFS methods regarding the number of selected features and regarding the classification accuracies of 5-folds cross-validation of BayesNet and C4.5 learning algorithms. The Weka tool [139] was used for obtaining the results. In order to solve the $M01LP$ problem for optimizing the GeFS (see Section 3.1.2), we used TOMLAB tool [60]. All the obtained results are listed in tables 4.13, 4.14, 4.15 and 4.16.

4.2.3 Experimental Results and Discussions

Table 4.13: Number of selected features (GA: genetic algorithm)

Data Set	Full-set	Our-method	Best-first	GA
Nor&Dos	41	3	6	11
Nor&Probe	41	6	7	17
Nor&U2R	41	1	4	8
Nor&R2L	41	2	5	8

Table 4.13 shows the number of features selected by our approach and those selected by using the best-first and GA search strategies. The identification of selected features is given in Table 4.14 (for feature names, see tables 4.6, 4.8, 4.9 and 4.10). Table 4.15 and Table 4.16 summarize the classification accuracies and false-positive rates, respectively, of the BayesNet and the C4.5 performed on four data sets (see above).

Table 4.14: Identifications of Selected Features

Data Set	Identifications
Nor&Dos	4, 5, 31
Nor&Probe	4, 5, 13, 26, 27, 31
Nor&U2R	33
Nor&R2L	29, 41

It can be observed from Table 4.13 that our $GeFS_{CFS}$ approach selects the smallest number of relevant features in comparison with the full and the feature sets selected by the best-first and GA search strategies. Especially in some cases, our new method compresses the full set of features extremely. For example, only one feature was selected out of 41 features of the Nor&U2R data set.

Table 4.15: Classification accuracies of C4.5 and BayesNet performed on KDD'99 dataset

Data Set	C4.5				BayesNet			
	Full-Set	GeFS	BF	GA	Full-Set	GeFS	BF	GA
Nor&DoS	97.80	98.89	96.65	96.09	99.99	98.87	99.09	99.72
Nor&Probe	99.98	99.70	99.71	99.89	98.96	97.63	97.65	99.19
Nor&U2R	99.97	99.96	99.97	99.95	99.85	99.95	99.97	99.93
Nor&R2L	98.70	99.11	99.01	98.86	99.33	98.81	98.95	99.28
Average	**99.11**	**99.41**	**98.84**	**98.69**	**99.53**	**98.82**	**98.91**	**99.52**

Table 4.16: False-positive rates of C4.5 and BayesNet performed on KDD'99 dataset

Data Set	C4.5				BayesNet			
	Full-Set	GeFS	BF	GA	Full-Set	GeFS	BF	GA
Nor&DoS	0.019	0.1	0.133	0.03	0.2	1.03	1.27	0.002
Nor&Probe	0.034	0.37	0.38	0.025	1.07	3.2	3.21	0.8
Nor&U2R	0.005	0.016	0.002	0.004	0.126	0.021	0.004	0.048
Nor&R2L	0.012	0.011	0.021	0.01	0.67	0.38	0.31	0.047
Average	**0.018**	**0.12**	**0.13**	**0.017**	**0.51**	**1.15**	**1.19**	**0.22**

In the Table 4.15, it can be observed that with our approach the average classification accuracies are slightly different from the ones obtained by using the best-first search or the genetic algorithm. The absolute difference between them does not overcome 0.69%. In the case of the C4.5 classifier, we got better performance. In the Table 4.16, with our method the false-positive rates are better than the ones obtained by applying the best-first search, yet worse than the full-set and GA search.

Even though the gained detection performances are comparative to other methods, the overall gain of the feature selection classification procedure lies in significantly improved efficiency (see Figure 4.7 for the training time of the NIDSs) and in obtaining the classification results due to reduced number of relevant features.

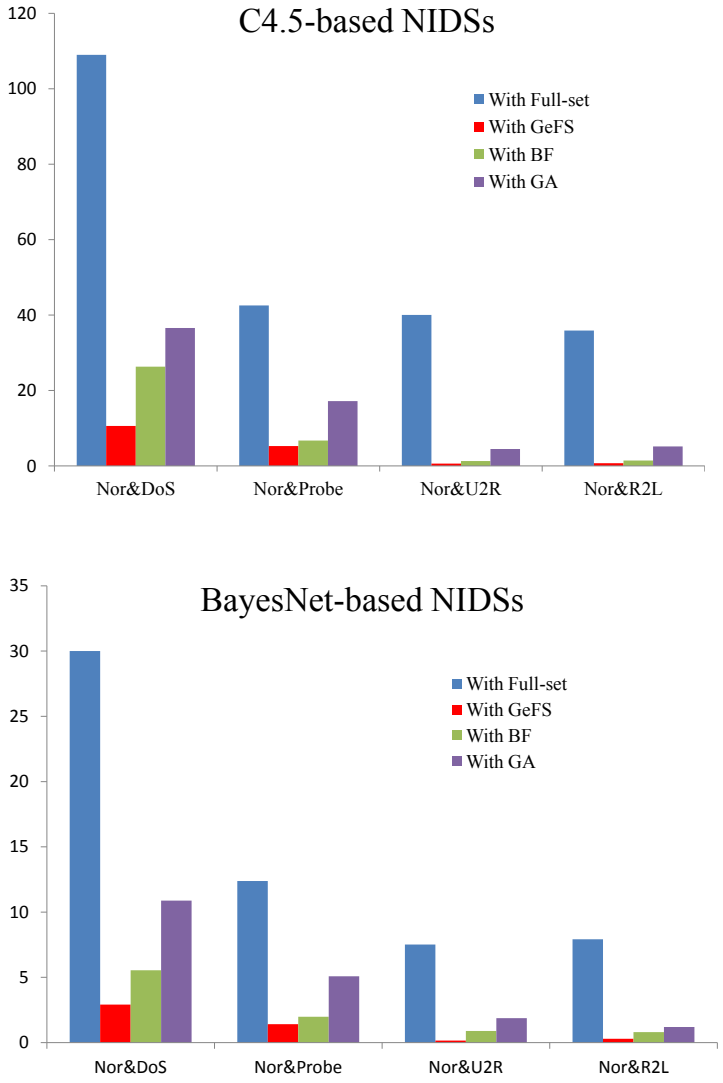


Figure 4.7: Training time of C4.5-based and BayesNet-based NIDSs performed KDD'99 dataset.

Therefore, based on all these experiments we can say that in general our new GeFS measure outperforms the best-first-CFS and genetic-algorithm-CFS methods by removing much more redundant features and still keeping the classification accuracies or even getting better performances. Thus it can be used to find optimal subsets of relevant features by means of the CFS measure for intrusion detection systems.

4.2.4 A Comparison of Feature Selection Methods for Intrusion Detection[3]

In previous subsection, we have showed that our proposed search method outperforms the heuristic search strategies by removing much more redundant features from the KDD CUP 1999 data set and still keeping the classification accuracies or even getting better performances. In this subsection, we validate our proposed feature selection measures by comparison with various automatic feature-selection algorithms for intrusion detection. The feature-selection algorithms involved in this comparison are previously known SVM-wrapper [129], Markov-blanket [35] and Classification & Regression Trees (CART) algorithms [35]. The details of these methods are given below.

SVM-wrapper for feature selection: Sung and Mukkamala [129] used the ranking method 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 used for the training and testing of the classifier Support Vector Machine (SVM) [40]. Then the SVMs performance is 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 will be ranked as "important", "secondary" or "insignificant" according to the following rules:

- 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.

[3]This section is published under the title:

Nguyen, H. T., Petrovic, S., and Franke, K. A comparison of feature-selection methods for intrusion detection. In 5th *International Conference on Mathematical Methods, Models and Architectures for Computer Network Security, MMM-ACNS* (Russia, 2010), pp. 242-255.

Table 4.17: Performance of SVM using selected features (SF) [129]

Classes	Number-of-SF	Accuracy
Normal	25	99.59%
DoS	19	99.22%
Probe	7	99.38%
U2R	8	99.87%
R2L	6	99.78%

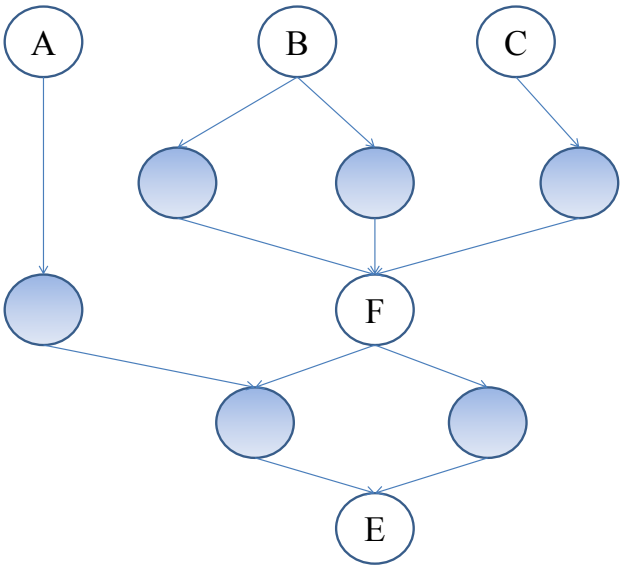


Figure 4.8: Sample of Markov blanket

Markov blanket for feature selection: Markov blanket MB(F) of the output variable F is defined as the set of input variables such that all other variables are probabilistically independent of F. Knowledge of MB(F) is sufficient for perfectly estimating the distribution of F and thus for classifying F. Markov blanket has been applied for feature selection in many domains [56]. In 2004, Chebrolu et. al. [35] proposed to use Markov blanket for selecting important features for intrusion detection. In order to do that, they constructed a Bayesian Network (BN) from the original data set. A Bayesian network $B = (N, A)$ 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 used to compute the conditional probability of one node, given values assigned to the other nodes. From the constructed BN, the Markov blanket of a feature F is the union of F's parents, F's children and eventually other parents of F's children. An example of a Bayesian Network is given in Figure 4.8. The blue-filled nodes constitute the MB(F):

For conducting the experiment, Chebrolu et. al. randomly chose 11,982 instances from the overall (5 millions of instances) KDD CUP'99 dataset. 17 features were selected and the Bayesian Network [45] was used for classifying the obtained data set after removing irrelevant features. The results are given in Table 4.18.

Table 4.18: Performance of Bayesian Network using selected features (SF) [35]

Classes	Number-of-SF	Accuracy
Normal	17	99.64%
DoS	17	98.16%
Probe	17	98.57%
U2R	17	60.00%
R2L	17	98.93%

Table 4.19: Performance of CART using selected features (SF) [35]

Classes	Number-of-SF	Accuracy
Normal	12	100%
DoS	12	85.34%
Probe	12	97.71%
U2R	12	64.00%
R2L	12	95.56%

The Classification and Regression Trees for feature selection: The Classification and Regression Trees (CART) approach [45] 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 use 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 $i_{importance} = (p^n) \times i_{improvement}$ in which p is the *surrogate improvement weight* which is a user controlled parameter set between 0 and 1.

Chebrolu et. al. [35] conducted the experiment on the data set, which contains randomly chosen 11,982 instances from the overall (5 millions of instances) KDD CUP'99 data set. 12 features were selected and the CART [45] was used for classifying the obtained data set after removing irrelevant features. The results are given in Table 4.19.

Since different intrusion detection systems used different feature-selection methods and different classifiers with the aim of achieving the best classification results, we compared the 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. For our experiment, we used the decision tree algorithm C4.5 [111] as classifier for the full-set data as well as for the data sets obtained by removing irrelevant features by means of the $GeFS_{CFS}$ and $GeFS_{mRMR}$ measures. All the obtained results are

shown in Figure 4.9 and Figure 4.10.

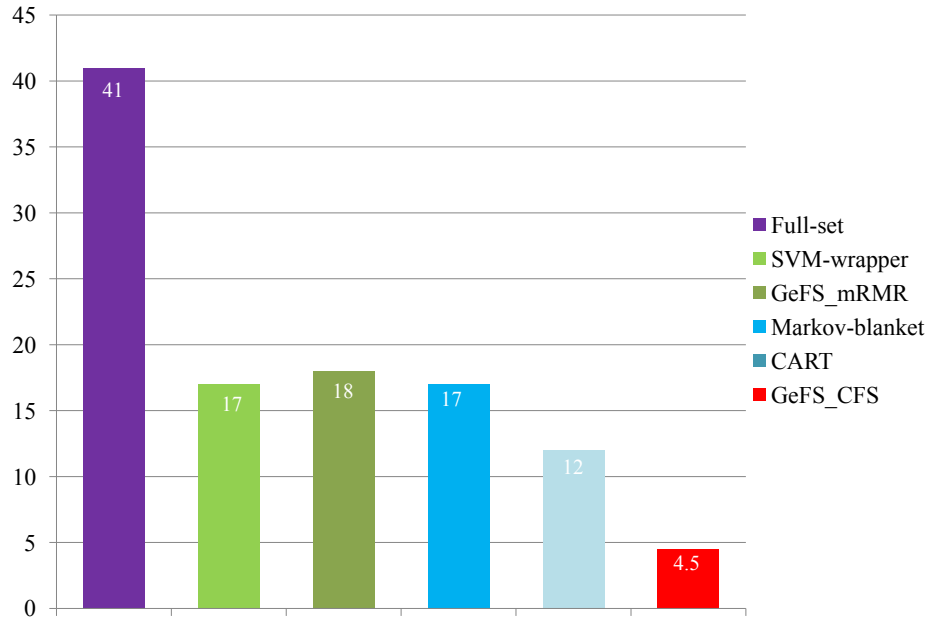


Figure 4.9: Number of selected features (on average)

Figure 4.9 shows the average number of features selected by the $GeFS$ feature-selection method and those selected by existing approaches. Figure 4.10 summarizes the average classification accuracies of chosen machine learning algorithms as classifiers for intrusion detection process.

It can be observed from Figure 4.9 that the $GeFS_{CFS}$ feature-selection method selects the smallest number of relevant features. Figure 4.9 shows that with the recently proposed approach the average classification accuracies are approximately the same or even better than those achieved by applying other methods.

4.2.5 Application of New Reliable Ensemble Feature Selection Framework for Intrusion Detection[4]

So far, in previous subsections we have applied the new reliable feature selection methods for the dataset, which has either a representative linear or non-liear relationships between features. However, in many cases a dataset has a mixture of statistical properties. Considering only one linear or non-linear property by applying a single feature selection method will lead to the over-selecting phenomenon, over-fitting and unreliable intrusion detection results (see the detail discussion in Section 3.2).

In this subsection, (1) we illustrate the over-selecting phenomenon by conducting an experiment on the KDD CUP'99 benchmarking IDSs data set [71, 72]. (2) We also show

[4]This section is published under the title:

Nguyen, H. T., Franke, K., and Petrović, S. A new ensemble-feature-selection framework for intrusion detection. In 11th *International Conference on Intelligent Systems Design and Applications (ISDA)*, 2011 (nov. 2011), pp. 213 218.

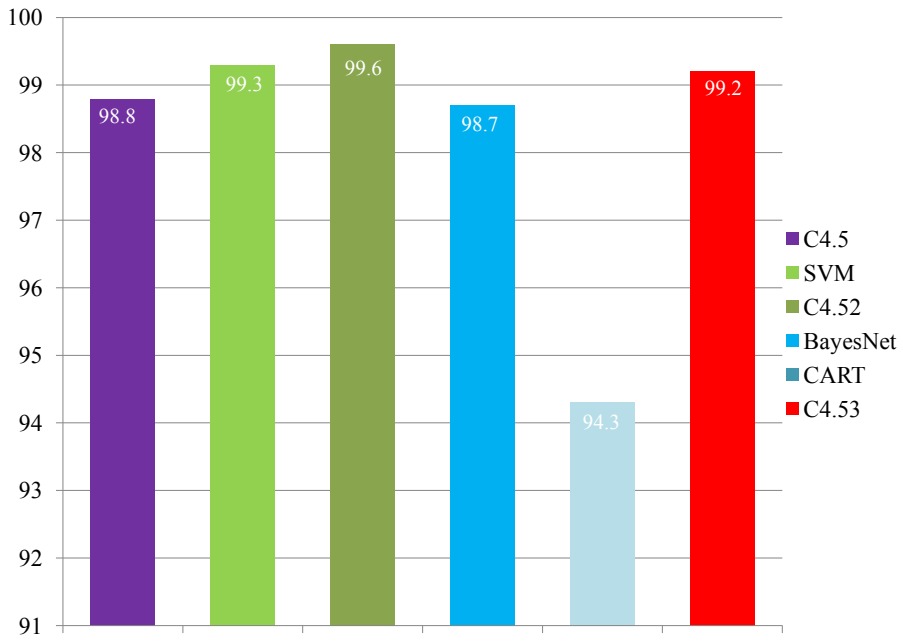


Figure 4.10: Classification accuracies (on average)

the influence of the over-selecting phenomenon on the over-fitting phenomenon of machine learning algorithms. Moreover, (3) we demonstrate how our new ensemble feature selection (EnFS) measure (see Section 3.2) is capable of reducing over-fitting by addressing over-selecting, thus providing more reliable intrusion detection results.

Experimental Setting *Add 1.* In order to illustrate the over-selecting phenomenon when the selected features from the training data are quite different from the representative features of the testing data, we separately implemented the $GeFS_{CFS}$ and the $GeFS_{mRMR}$ measures [94, 98] defined the section 3.1 for selecting features from a part of the KDD CUP 1999 data sets, which is different from the datasets in previous subsections. In fact, we selected 10% of the overall (5 millions of instances) KDD CUP'99 IDS benchmarking labelled data as the training data set. Another 5% of the remained KDD CUP'99 data were randomly selected and used as the testing data. As discussed before, these data sets contain normal traffic and four different attack classes: (i) Denial of Service (DoS) attacks, (ii) Probe attacks, (iii) User to Root (U2R) attacks and (iv) Remote to Local (R2L) attacks. Since the two attack classes U2R and R2L have been criticized [87, 119], we did not consider them for this experiment. Details of the number of class instances are given in Table 4.20.

As the attack classes distribute so differently, the feature selection algorithms might concentrate only on the most frequent class data and neglect the others. Therefore, we chose to process these attack classes separately. In order to do that, we added normal traffic into each attack class to get two data sets: Normal&DoS and Normal&Probe. With each data set, we separately ran two feature-selection algorithms: the $GeFS_{CFS}$ and the $GeFS_{mRMR}$. The feature sets selected from the training data ($Train$) and from the testing data ($Test$) were then compared with each other by means of the Jaccard distance [56].

Table 4.20: The partition of KDD CUP'99 data set used in the experiment

Classes	Training Set		Testing Set	
	Instances	Percentage	Instances	Percentage
Normal	97,278	18.35	56,234	24.73
DoS	391,458	73.88	150,046	65.99
Probe	41,113	7.77	20,987	9.28
Total	529,849	100%	227,357	100%

This distance measures the similarity between two sets $Train$ and $Test$, and is defined as follows:

$$J(Train, Test) = \frac{|Train \bigcap Test|}{|Train \bigcup Test|}$$

The results of this experiment are given in Table 4.21.

Add 2. In order to show the influence of the over-selecting phenomenon on the over-fitting phenomenon of machine learning algorithms, we applied the C4.5 [111] and the BayesNet [45] algorithms on the newly obtained training and testing data sets that include only the selected features from the original training data. The obtained classification accuracies were then compared with each other. The differences between testing and training classification accuracies are calculated as follows:

$$DIFF(Train, Test) = Acc_{Test} - Acc_{Train}$$

Table 4.22 shows the obtained results of this experiment.

Add 3. To validate our new ensemble feature selection measure in addressing the first two main causes of the over-selecting phenomenon as mentioned above, we proposed to combine the $GeFS_{CFS}$ with the $GeFS_{mRMR}$ measures as follows:

$$\max_{x \in \{0,1\}^n} EnFS = \frac{1}{2}\left[GeFS_{CFS} + GeFS_{mRMR}\right] \tag{4.1}$$

The new constructed feature selection measure 4.1 was then used for selecting features from the original training and testing data sets. We applied our new proposed search algorithm and used the TOMLAB tool [60] for solving the $M01LP$ problem. We also applied 10-folds cross-validation of the C4.5 and the BayesNet machine learning algorithms on the newly obtained data sets that include only the selected features from the original training data. The Weka tool [139] was used for determining classification accuracies, which were then compared with each other. All results are given in Table 4.21 and Table 4.22.

Experimental Results Table 4.21 shows the Jaccard distances of feature sets selected from the training data ($Train$) and the testing data ($Test$). The differences between testing and training classification accuracies are given in Table 4.22.

Table 4.21: Jaccard distance of two selected feature sets $Train$ and $Test$

Data Set	$\mathbf{GeFS_{CFS}}$	$\mathbf{GeFS_{mRMR}}$	**EnFS**
Normal&Dos	0.14	0.33	0.75
Normal&Probe	0.29	0.40	0.81

Table 4.22: The differences between Testing and Training accuracies ($\mathbf{Acc_{Test}} - \mathbf{Acc_{Train}}$). The Ge-CFS and Ge-mRMR are the notations of $GeFS_{CFS}$ and $GeFS_{mRMR}$, respectively.

Data Set	**C4.5**			**BayesNet**		
	Ge-CFS	**Ge-mRMR**	**EnFS**	**Ge-CFS**	**Ge-mRMR**	**EnFS**
Normal&DoS	–2.80	–1.98	+2.89	–1.99	+1.36	+1.87
Normal&Probe	–2.01	+0.19	+1.11	+0.03	–0.17	+2.81

It can be observed from Table 4.21 that the over-selecting phenomenon occurs when the $GeFS_{CFS}$ and the $GeFS_{mRMR}$ measures were applied separately. Because the Jaccard distances of the selected feature sets $Train$ and $Test$ are very low, such as in the case of the Normal&DoS data set with the $GeFS_{CFS}$ feature selection measure, the $J(Train, Test) = 0.14$. With the new ensemble feature selection measure $EnFS$, the Jaccard distances of the selected feature sets $Train$ and $Test$ are the greatest. This indicates that the feature sets selected by means of the $EnFS$ from the training and the testing data sets are more similar than those selected by means of the $GeFS_{CFS}$ or the $GeFS_{mRMR}$ measures.

From Table 4.22, it can be observed that the over-fitting phenomenon happened when we applied the C4.5 and the BayesNet algorithms on the data sets, which contain only features selected by means of the $GeFS_{CFS}$ measure or the $GeFS_{mRMR}$ measure. With the $EnFS$, this over-fitting phenomenon did not happen.

In summary, this subsection has illustrated the over-selecting phenomenon by conducting an experiment on the KDD CUP'99 benchmarking IDSs data set. The influence of the over-selecting phenomenon on the over-fitting phenomenon of machine learning algorithms was shown. Moreover, this section has applied the new ensemble feature selection (EnFS) measure (see Section 3.2) to reduce the over-fitting by addressing over-selecting, thus providing more reliable intrusion detection results.

In the next section, we continue to introduce the application of our new proposed reliable machine learning algorithms to the botnet-malware detection systems in the network layer (see Figure 4.1).

4.3 Botnet-Malware Detection[5]

In this section, we show the application of the generic feature selection ($GeFS$) measure (see subsection 3.1.2) in botnet-malware detection. We first describe our own generated

[5]This section is submitted under the title:

Berg, P. E., Franke, K., Nguyen, H.T. Generic Feature Selection Measure for Botnet Malware Detection, *The 12th International Conference on Intelligent Systems Design and Applications (ISDA)* (accepted, 2012).

dataset, on which the experiments were conducted. We then use the static and dynamic approaches [48, 121, 147] to extract features and to generate two separate feature sets. We analyze the statistical properties of the extracted feature sets to choose appropriate instances from the $GeFS$ measure. Since there are no standard botnet-malware detection systems, we apply five different machine learning algorithms (Naive-Bayes, K-nearest neighbors, C4.5, SVM and Bayesian Network [45]) to evaluate the detection rates as well as the false positive rates on datasets containing the selected features.

4.3.1 Datasets

In this section, we describe the dataset, which was acquired in a master thesis [26], and show how to extract important features for the botnet-malware detection task.

Dataset Acquisition: The data acquisition consists of two steps: 1) Setting and 2) Generating.

Ad. 1 Malware and benign software need to be manually downloaded and stored in our database. This is done to be sure that the analyzed malware is utilized in a known botnet and that we can acquire a suitable dataset in a limited time. Furthermore, to appropriately test the classifiers, benign executables should share some similar behavior when they are analyzed with static and dynamic analysis tools, for example some network-related activities. Different portable software, such as mail clients, browsers, network tools, instant message clients and BitTorrent clients [7] were used to generate 50 benign samples. We used the malicious softwares from Web sites, such as vxheavens [11], packetstorm [6] and offensivecomputing [5], to generate 90 malware samples. These malware samples are from three large malware families: *SpyBot* [3], *Torpig* [4] and *SdBot* [128]. They exist in botnets where one of their tasks is to establish and maintain a connection with the botmaster's C&C server and await further commands. The reason for choosing these samples is that we wish to study characteristics of botnet-malware that utilize different behavior on the host. In more detail, three malware families are as follows:

-*SpyBot* [3] is a worm that propagates through P2P-sharing and IRC. This worm can also infect hosts that are already compromised by common backdoor programs. In botnets, various versions of SpyBot have been used for C&C related activities and launched DDoS attacks.

-*Torpig* [4] also known as Sinowal or Anserin, is a Trojan that logs keystrokes and activity to certain bank Web sites. The Trojan employs domain flux in order to communicate with its main C&C servers [128].

-*SdBot* [128] also known as Randex or Agent, is a backdoor that connects to an IRC channel using its own IRC client. From here an adversary can remotely control the infected host to for example perform DDoS attacks against a third party and try to infect other users connected to other IRC channels.

Ad. 2 The second step involves gathering and analyzing the executables's behavior. For this task, we utilize static and dynamic analysis tools (see, for example [22, 81]). The static analysis tool [81] will analyze the dataset without executing the files and therefore avoids to infect the system. Here the portable executable (PE) format need to be exploited by using a PE parser (e.g., pefile), that enables us to retrieve information from the different sections and store this in a static report. The reason for choosing this static analysis approach is that behavior-based characteristics of the files are available without going to a lower abstraction level that requires an analysis of the assembly code. Hence, this approach will not be vulnerable to obfuscation techniques that are applied to the code to confuse the investigator or the disassembler, and have given good results when applied in malware detectors in recent research.

With dynamic analysis approach, a sandbox (e.g., Anubis [22]) is applied to run the executables in a controlled manner utilizing an isolated environment. Thus, the executables, with malicious intent, do not have any propagation opportunities. The sandbox needs to emulate a computer instead of a virtual machine that operates by executing instructions directly on the real processor. This will ensure that malware cannot escape from the emulated environment, and malware cannot detect its presence as easily as with, for example VMWare [10, 21]. The purpose of utilizing a sandbox is that malware will be loaded into memory and executed, to retrieve its system interactions in a controlled manner. Thus, in addition to defeating polymorphism and metamorphism, it will unpack and decrypt malware that is protected by packers and cryptors when it is loaded into memory before execution. Recent research shows that analyzing malware with sandboxes has given satisfactory results (see, for example [115, 141]). When utilizing a sandbox, the executable's actions need to be monitored and logged, to further produce a dynamic report. All actions stored in the dynamic report will reflect the executable's behavior when it is run on an actual system.

Feature Extraction: We apply the static and dynamic approaches [49, 121, 148] to extract features from the originally acquired dataset. Table 4.23 and Table 4.24 summarize the extracted features from both static and dynamic analysis.

Table 4.23: Static features

Feature	Value	Description
DLL import	Boolean(s)	Reflects imported DLL in the PE-header of an executable
Function calls	String(s)	Reflects all functions called within a DLL

The static feature set needs to be built to handle several DLL names and function call names in order to represent the behavior of each sample:

$$feature\_set_{static} = (...dll\_name_i, .., dll\_function\_name_i, ..)$$

Here the dll_name_i represents the imported DLL used in a sample and the $dll\_function\_name_i$ represent the names of the functions called within a DLL. This implies that the feature $dll\_function\_name_i$ holds several feature values:

$$dll\_function\_name_i = (function_{i1}, .., function_{in}).$$

Similarly, we constructed the dynamic feature set from the extracted entities in the dynamic reports. Here each sample is represented by multiple entities:

$$feature\_set_{dynamic} = (entity_1, .., entity_i, .., entity_l)$$

where each $entity_i$ is a set of features as follows: $entity_i = (feature_{i1}, .., feature_{im})$. Since dynamic reports may contain multiple entities of similar type, for example created file, it is needed to store multiple feature values for each feature: $feature = (value_1, .., value_k)$. For more detail, in the following we describe where the extracted features come from.

Table 4.24: Dynamic features

Entity	Description
1:Loaded DLL	Features from loaded DLL dependencies information
2:Created registry key	Features holding information on created registry keys
3:Modified registry key	Features related to registry modifications
4:Read registry value	Features from read registry values information
5:Created file	Features corresponding to name of created files
6:Modified file	Features on modifications done to existing files
7:Deleted file	Features related to deleted files by the executable
8:Read file	Features describing files read by the executable
9:Memory mapped file	Features describing memory mapped files, e.g. DLL usage
10:Driver communication	Features reflecting communication to a system driver
11:Control communication	Features that correspond to file system control operations
12:Thread status	Features related to creation of threads and their status
13:Remote thread created	Features describing processes created by threads
14:Process created	Features that reflect process creation and its purpose
15:Socket	Features describing network socket connections
16:DNS query	Features corresponding to DNS queries to domain names
17:SMTP conversation	Features extracting from e-mail using SMTP
18:HTTP conversation	Features reflecting properties in a HTTP conversation
19:TCP conversation	Features reflecting properties in a TCP transmission
20:UDP conversation	Features reflecting properties in a UDP transmission

DLL Dependencies: DLL dependencies are the type of system libraries the executable requires in order to execute. Since the executables satisfy the portable executable standard, the required libraries will be available on a Windows-based system. However, the type of libraries applied is dependent on the type of Windows version the executable is designed for. Both the static and dynamic analysis tools are able to retrieve type of libraries the executable is dependent on, where the PE parser also extracts the different function calls that are used within each library.

Registry Activities: Retrieving features reflecting the registry activity is needed to reveal configuration settings applied by the executables. This type of activity is especially relevant for malware that makes sure that they will be run after a reboot by adding autorun keys and values to the Windows registry [8]. Additionally, malware can use the registry to make sure that certain services will be disabled/enabled and to open ports; for example, disabling the antivirus service and enabling remote login. Thus, typical registry mechanisms that malware exploits need to be retrieved by the sandbox, see entities 2-4 in Table 4.24.

File Activities: File activities are a common operation performed by executables. A typical example is installers that store temporary installation data when the executable is run-

ning. Malware utilizes file activities for many reasons, for example primitive actions to stay undetected by an average user. Malware tends to copy itself to different directories, often system directories, where its filename is changed to something that does not seem suspicious. Also, malware may hide its tracks by deleting itself from the directory it was launched. See entities 5-9 in Table 4.24.

Process and Thread Activities: A process represents an instance of the executable when it is running, and each process can be made up of several executing threads that perform specific tasks. Retrieving features from these activities is important, because it is from these components all communication to system drivers and libraries are initiated, see entities 10-14 in Table 4.24. Events typically observed from malware are that they spawn additional processes with similar or identical names to running system processes.

Network Activities: Features describing network activities related to protocols such as HTTP, SMTP, TCP and UDP, need to be retrieved in order to observe which actions the executable is performing to other hosts on the network or Internet, see entities 15-20 Table 4.24. Malware that is utilized in a botnet, to expand the botnet, communicate with a C&C server, or launch attacks, may be logged by the sandbox and found in the dynamic reports. A newly infected bot will announce that it exists by trying to connect to the C&C server. Since embedded IP-addresses tend to be blacklisted, a series of DNS queries will be performed to resolve the correct IP-address. Furthermore, bot malware may be propagated by searching for hosts with known vulnerabilities. Port scanning is part of this activity, which is often done by exploiting weaknesses in TCP and/or UDP protocol at the victim to reveal open ports [123, 124]. Note that the fact when these activities are retrieved and later used as features really depends on several aspects. The amount of time the sandbox uses to analyze the executable sets a limit on how much network activity is logged. Also, C&C servers might have been taken offline and therefore not available to the specific bot anymore.

In total, we extracted 1814 static features and 5494 dynamic features from the originally acquired dataset. With these huge amounts of features, it is necessary to utilize feature selection methods for improving the effectiveness of botnet-malware detection. In the next subsections, we will show the experimental setting and results of the application of the generic feature selection ($GeFS$) measure to the extracted feature sets.

4.3.2 Experimental Setting

After extracting the static and dynamic feature sets from the originally generated dataset, we analyzed their statistical properties to see whether they have linear or non-linear relations between features. From this analysis, the appropriate feature selection instance from the $GeFS$ measure was chosen for each dataset according to the Step 1 of the search method 3.1 described in Section 3.1. In order to do that, we first visualized the whole datasets in the two-dimensional space to get a plot matrix, in which each element was the distribution of data points depending on the values of a feature and the class label or the values of two features. For instance, Figure 4.11 shows the sample distributions of data points of the dynamic feature set. We then calculated the correlation coefficients between the features. From these, we observed that these feature sets have many features that were linearly correlated to each other as shown in Table 4.25. Therefore, we chose the $GeFS_{CFS}$ measure for selecting features from the feature sets.

We applied the optimization algorithm proposed in subsection 3.1.4 to find globally optimal feature subsets by means of the measure $GeFS_{CFS}$. We then combined the features that were selected from the static and dynamic feature sets. In this experiment, we

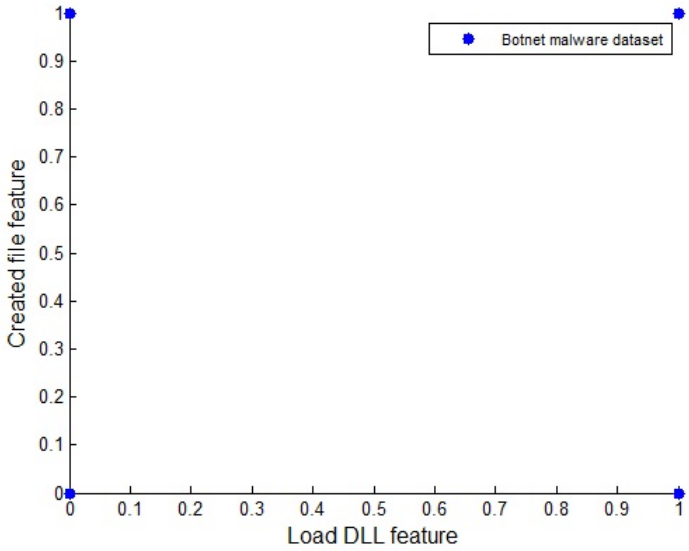


Figure 4.11: Sample of distributions of data points from the dynamic feature set.

Table 4.25: Percent of correlation values between features of the data sets

Feature Sets	**Greater-than 0.1**	**Less-than 0.09**
Static	86.96%	13.04%
Dynamic	97.24%	2.76%

compared the $GeFS_{CFS}$ measure with the GA_{CFS} and the BF_{CFS} methods [59] that used the genetic-algorithm (GA) and the best-first (BF) search, respectively, to find relevant features. Five classifiers with 10-fold cross validation were used to evaluate detection performances before and after feature selection: Naive-Bayes, K-nearest neighbors, C4.5, SVM and Bayesian Network [45]. All the obtained results are given in the tables 4.26 and 4.27.

4.3.3 Experimental Results and Discussions

Table 4.26 shows the number of full-set features before feature selection and the number of features selected by the $GeFS_{CFS}$, the GA_{CFS} and the BF_{CFS} methods. Table 4.27 summarizes the average detection rate as well as the average false positive rate of five different classifiers (Naive-Bayes, K-nearest neighbors, C4.5, SVM and Bayesian Network) performed on the generated datasets.

It can be observed from Table 4.26 that the $GeFS_{CFS}$ removes 99.9% of irrelevant and redundant features from original datasets. Moreover, the $GeFS_{CFS}$ measure outperforms the GA_{CFS} and BF_{CFS} methods by removing much more redundant features and more 18 redundant features, respectively.

Table 4.26: The number of selected features (BF: best first; GA: genetic algorithm).

Feature Sets	Full-set	GeFS	BF	GA
Static	1814	7	11	515
Dynamic	5494	5	19	1956
Combination	7308	12	30	2471

Table 4.27: Experimental results on the dataset.

Feature Sets	Average Detection Rate				Average False Positive Rate			
	Full-set	GeFS	BF	GA	Full-set	GeFS	BF	GA
Static	97.84	97.2	97.85	97.19	20.25	13.71	11.67	20.48
Dynamic	88.60	87.74	87.74	87.30	16.20	9.22	6.62	10.14
Combination	93.76	95.11	93.77	90.74	17.96	9.74	7.89	12.83

In the Table 4.27, it can be seen that the Full-set and GA_{CFS} approaches gave good detection rates, but very high false positive rates. Furthermore, the numbers of features using in these cases are large. Therefore, we are not interested in the experimental results given by the Full-set and GA_{CFS} approaches.

The $GeFS_{CFS}$ and the BF_{CFS} methods provided good detection rates as well as false positive rates in the case of combination feature sets. Those results are comparable with each other. However, the feature selection results given by the $GeFS_{CFS}$ measure are usually more reliable than the ones obtained by applying the BF_{CFS} method (about reliability in feature selection, see section 3.1). The reason is that the BF_{CFS} utilized a heuristic search method, i.e., the best first search to find relevant features by means of the CFS measure. With different settings of the search process, we might select different feature subsets. Furthermore, as the Table 4.26 shows the BF_{CFS} selected more features than the $GeFS_{CFS}$. Thus, the results provided by the $GeFS_{CFS}$ are the best.

In summary, this section has proposed to apply the generic feature selection ($GeFS$) measure for botnet-malware detection systems in the network layer according to Figure 4.1. This section has analyzed statistical properties of the new generated datasets. Based on this analysis, the $GeFS_{CFS}$ measure was chosen for selecting relevant features from the datasets. The experiment in this section has compared the $GeFS_{CFS}$ measure with the genetic-algorithm-CFS and the best-first-CFS methods regarding the feature selection capabilities in botnet-malware detection. The detection accuracies obtained after the feature selection by means of five different classifiers were tested. The experimental results showed that 99.9% of irrelevant and redundant features were removed from the datasets, while keeping or yielding even better classification performances. Moreover, the $GeFS_{CFS}$ measure outperformed the genetic-algorithm-CFS and the best-first-CFS methods by removing much more redundant features and by providing more reliable feature selection results.

In the next section, we introduce the application of our new proposed machine learning algorithms to the host-based intrusion detection system in the operating system layer (see Figure 4.1).

4.4 Host-based Intrusion Detection[6]

If the network-based intrusion detection systems (NIDSs) described in Section 4.2 collect input data by monitoring network traffic (e.g, packets captured by network interfaces), the host-based intrusion detection systems (HIDSs), on the other hand, rely on events collected by the hosts they monitor [135]. Depending on the type of audit data, HIDSs can be categorized into two classes: operating system-level intrusion detection systems and application-level intrusion detection systems. In this section, we focus on the first class of the HIDSs.

HIDSs in this class use information provided by the operating system (OS) to identify attacks. This information can be system calls, such as file system modifications, user logons and so on. For the OS-level auditing data-gathering mechanisms, see [135] for example.

Many different features can be extracted from the captured audit data to detect attacks. However, not all of them are important and relevant for detection task. Therefore, a feature selection process is necessary. In this case, the wrapper model for feature selection is preferable than the filter model for feature selection, which was applied to NIDSs in Section 4.2. The first reason is that the wrapper model assesses features by a learning algorithm's performance, thus providing more reliable and accurate intrusion-detection results. The second reason is that the wrapper method still works efficiently with the audit data captured at the OS-level of a computer, which is normally much smaller than the data that NIDSs can capture in the network.

In this work, we propose to apply our new general Lp-norm SVM (Section 3.3), which is a feature-selection method of the wrapper model, for the host-based intrusion detection systems.

4.4.1 Datasets

We conduct the experiment on two benchmark datasets for host based intrusion detection sytems:*(1)* UNM [9] and *(2)* the MIT Lincoln Lab databases [73].

Ad. 1 The University of New Mexico (UNM) provides a number of system call datasets [9] for testing host-based intrusion detection systems (HIDSs). The system calls were collected when executing different kinds of programs, which vary widely in their size and complexity and contain different kinds of intrusions (buffer overflows, Trojan programs and so on). The normal data are "synthetic" and "live". The synthetic traces were generated in production environments by running a prepared script with chosen program parameters. The live normal data is captured during normal usage of applications. Each trace is the list of system calls issued by a single process from the beginning of its execution to the end. Trace lengths vary widely because of the differences in program complexity and because some traces are daemon processes and others are not.

In the UNM datasets, a feature is defined as the number of occurrence of a system call in an input sequence [64]. Each instance in the UNM datasets, which is labeled as "normal" or "intrusion", has a large number of feature values, such as Xlock has 200 features. We used five different datasets from the UNM database: "L-inetd","Login", "PS","S-lpr" and "Xlock". More detail on the numbers of features of the datasets is given in Table 4.28.

Ad. 2 The datasets generated by MIT Lincoln Lab in 1998 [73] were used for benchmarking of different intrusion detection systems and were introduced in Section 4.2. However, in this section we only consider two system call datasets for testing host-based intrusion

[6]This section is published under the title:

Nguyen, H. T., Petrović, S., and Franke, K. A general l1-norm support vector machine for feature selection. In 3rd *International Conference on Machine Learning and Computing* (Singapore, 2011), pp. 591-595.

detection systems: "RawFriday" and "RawMonday" datasets. The "RawFriday" and the "RawMonday" datasets have 53 and 54 features, respectively.

4.4.2 Experimental Setting

This subsection aimed to apply the embedded model to select important features from the UNM and MIT Lincoln Lab datasets. Particularlly, we ran four different algorithms on the chosen datasets: L1-norm SVM (L1-SVM), L2-norm SVM (L2-SVM), GL1-norm SVM (GL1-SVM) and GL2-norm SVM (GL2-SVM). For the L2-SVM and the traditional L1-SVM, we used the Mangasarian's code from [1]. To implement the new general Lp-SVM (GLp-SVM, p=1 and p=2), the TOMLAB tool [60] was used for solving the mixed 0-1 linear programming problem if $p = 1$ and the mixed 0-1 quadratic programming problem if $p = 2$. The values of the error penalty parameter C used for the experiment were: $2^{-10}, 2^{-9}, 2^{-8}, ..., 2, ..., 2^8, 2^9, 2^{10}$. We applied 10-fold cross validation for estimating the average classification accuracies as well as the average number of selected features. All the best results obtained over those penalty parameters were chosen and are given in the Table 4.28 and the Table 4.29.

4.4.3 Experimental Results and Discussions

Table 4.28: Number of selected features (on average)

Data Sets	**Full-set**	**L2-SVM**	**GL2-SVM**	**L1-SVM**	**GL1-SVM**
RawFriday [73]	53	33.9	33.9	2.4	1.9
RawMonday [73]	54	26	26	1	1
L-inetd [9]	164	33.5	23	13.5	2.4
Login [9]	164	46	30	9.6	2
PS [9]	164	22	22	5	2
S-lpr [9]	182	36.9	36.9	3.2	2
Xlock [9]	200	46.8	46.8	7.5	1
Average	**140.1**	**35.01**	**31.22**	**6.02**	**1.75**

Table 4.29: Classification accuracies (on average)

Data Sets	**L2-SVM**	**GL2-SVM**	**L1-SVM**	**GL1-SVM**
RawFriday [73]	98.40	100	84.2	98.80
RawMonday [73]	100	100	95.65	100
L-inetd [9]	88.33	90.05	85.00	85.83
Login [9]	80.00	85.00	65.00	81.67
PS [9]	100	100	100	100
S-lpr [9]	100	100	87	99.11
Xlock [9]	100	100	96.19	100
Average	**95.24**	**96.43**	**87.57**	**95.05**

Table 4.28 shows the number of features selected by our GLp-norm SVMs and those selected by the traditional Lp-norm SVMs. Table 4.29 summaries the classification accuracies of 10 folds cross-validation of the SVMs performed on 7 datasets.

It can be observed from Table 4.28 that our new method GL1-SVM removes dramatically irrelevant and redundant features from the full-set of features. Surprisingly, in some cases the GL1-SVM selected only one important feature, such as in the Xlock and RawMonday datasets. Moreover, in comparison with other methods the GL1-SVM provided a smallest number of important features. The GL2-norm SVM selects smaller numbers of features than the L2-SVM, but larger than the L1-SVM.

From the Table 4.29, it can be seen that the GL1-SVM provided higher performance than the one given by the traditional L1-SVM. In fact, the GL1-SVM improved the classification accuracy by more than 7.48%. This phenomenon can be explained by the fact that L1-SVM is just a case of the GL1-SVM, thus solving the GL1-SVM results a better classification accuracy than solving the traditional L1-SVM. Moreover, the datasets contain many irrelevant and redundant features that negatively affect the performance of SVMs. These explanations can also be applied to the case of the GL2-SVM and L2-SVM. As shown in Table 4.29, GL2-SVM gave the best classification accuracy on average. In some cases, such as in the Xlock and RawMonday datasets, one feature is enough to identify attacks and normal traffic.

In summary, this section has applied the new general Lp-norm support vector machine (GLp-SVM) for the host-based intrusion detection systems. The experiment was conducted on two datasets: the UNM and the MIT Lincoln Lab. The experimental results showed that the GLp-SVM provided better classification accuracies than the traditional Lp-SVMs do, while using fewer numbers of features. Therefore, the intrusion detection results of HIDS are more reliable than the ones obtained by runing the traditional Lp-SVM.

Previous sections have introduced the applications of our new proposed reliable machine learning algorithms for various intrusion detection systems in three different layers: application layer, network layer and operating system layer. In the next section, we show the last application of our new algorithms in this dissertation to test of Web Application Firewalls (see Figure 4.1).

4.5 Testing of Web-Application Firewalls[7]

This section presents a new methodology to simplify the evasion of an Web Application Firewall (WAF), thus providing an efficient testing of the WAF. State-of-the-art evasion techniques are based on looking for flaws or vulnerabilities in the behaviour of these systems, but this behaviour is normally unavailable or difficult to understand. To cope with this problem, we propose to model WAFs using Genetic Programming (GP) and feature selection methods. The resulting models detect Web attacks similarly to the WAF, but are simpler to understand and process. Accordingly, evasions over the original WAF are found by looking at these models. We run the experiments using HTTP traffic obtained from a simulated environment and an experimental WAF. The results show that feature selection aids to considerably speed up the evasion search.

In the following, we first introduce the data used, the contents and how it is collected. Second, we explain how we construct the experimental WAF, its characteristics and performance.

[7]This section is submitted under the title:

Pastrana, S., Gimenez, C. T., Nguyen, H. T., and Orfila, A. A methodology to simplify the evasion of web application firewalls. *Computers & Security Journal*, Elsevier. (submitted, 2012)

Table 4.30: 89 characters constructed using n-grams from the HTTP dataset CSIC 2010. ⋆ refers to features selected by the $GeFS_{CFS}$ measure and ◇ to features selected by the Consistency measure.

Features Extracted with N-grams																		
!	◇	″		#	◇	$	⋆	LF		Space	⋆							
%	◇	:		&	◇	′	⋆	(		)	⋆							
*	◇	+	◇	,		-	◇	.	◇	/	⋆							
0		1	◇	2		3	⋆◇	4	◇	5	⋆◇							
6	⋆	7	⋆◇	8	◇	9	◇	;		<	⋆◇							
=	◇	>	⋆◇	?		@		A		B								
C	◇	D	⋆◇	E		F	⋆	G	◇	H	⋆◇							
I	⋆	J		K	◇	L	⋆◇	M	◇	N	⋆							
O	⋆	P	⋆	Q		R	⋆	S	⋆◇	T	◇							
U	⋆	V	⋆◇	W	⋆	X	◇	Y	⋆◇	Z	⋆◇							
_		a	⋆	b		c		d		e	⋆							
f	⋆◇	g		h		i		j	◇	k	⋆◇							
l	◇	m	⋆	n	⋆◇	o		p	⋆	q	⋆							
r		s	⋆◇	t		u		v	⋆	w	⋆							
x	◇	y		z	⋆◇	—	⋆	~										

4.5.1 Datasets

A current problem in Web attack detection is the lack of publicly available datasets to test WAFs. The DARPA dataset [73] has been widely used for intrusion detection. However, it has been criticized by the research community [87]. Its Web traffic is out of date and does not include many of the current attacks, which makes it inappropriate for the detection of Web attacks. Therefore, we decided to use the CSIC 2010 HTTP dataset as described in subsection 4.1.1.

For feature construction, we use features from expert knowledge (see Table 4.1 in Section 4.1.1) in addition with the features extracted by using n-grams method as described in subsection 2.1.1.1. An n-gram is a subsequence of n items from a given sequence [32, 86]. In byte sequences, the number of possible subsequences with length n is 256^n. In this work we set $n = 1$ (the simplest case), thus the number of 1-grams is 256. We establish a correspondence between the 256 1-grams and the 256 ASCII characters.

With the assumption that normal requests are different from the anomalous ones, we use the following automatic feature construction method based on the n-gram method for intrusion detection: Given an HTTP request p, a feature vector of p is the vector $x_p = (x_1, x_2, \ldots, x_{2^{8n}})$, where x_i is the number of appearances of the ith n-gram in the request p, which is considered as a string of characters. Therefore, in the case of 1-grams the number of appearances of each character is a feature. We observe that in the CSIC 2010 HTTP dataset only 89 characters are present. Thus, they can be considered features. These 89 characters are listed in Table 4.30.

We combine these 89 features with 28 features from expert knowledge to construct a dataset with 117 features. The dataset is divided into three subsets by randomly selecting instances from it. One third of the dataset is used to train and generate the detection engine of the WAF (below we give details of the construction process). The second subset is used to train the GP models and the third subset to test these models. Each instance of the dataset is selected only once, so it is assured that instances are not repeated among the three subsets.

4.5.2 Experimental WAF

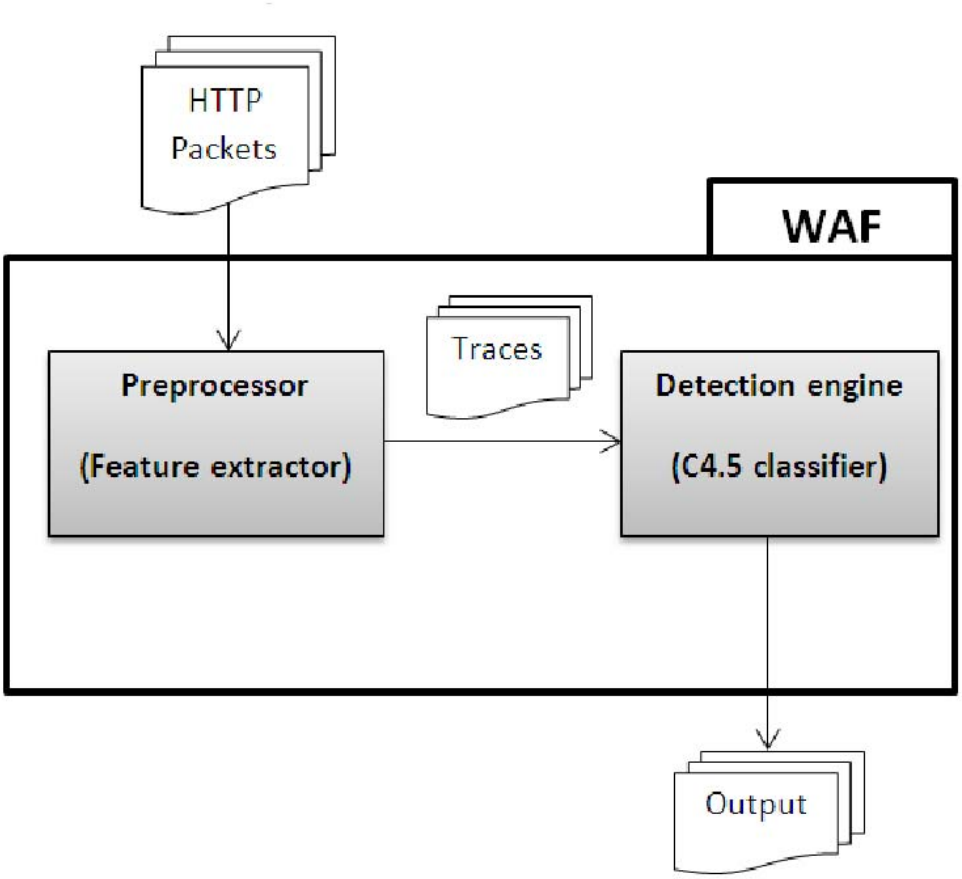


Figure 4.12: Structure of the WAF studied. It is composed of a raw traffic preprocessor and the detection engine.

The WAF that we are going to evade is a detector that, given a trace of data, classifies it as normal or intrusive. The detection engine of the WAF is a decision tree algorithm, the C4.5 [111]. We choose to use a decision tree due to the fact that this kind of algorithms is widely used in the field. In fact, the winner of the famous DARPA intrusion detection contest [73] was an algorithm based on decision trees [109]. We have trained the WAF to classify the HTTP packets using both normal and intrusive packets. In the training mode, we give an initial dataset from which the tree learns how to classify the instances. Figure 4.12 shows the structure of the WAF created. First, the raw traffic is preprocessed as explained in the previous section to extract the features and label each HTTP packet, indicating whether it is normal or intrusive. Then, one third of the dataset is entered to train the detector using the C4.5 algorithm in the WEKA software [139]. As output, the software returns the decision tree that represents the detection engine of the WAF.

In order to evaluate the effectiveness of this WAF, we first define the measures used. Table 4.31 shows the contingency matrix of a detection system. From its notation, several metrics are defined:

- Hit rate (H).

$$H = \frac{TP}{TP + FN} \tag{4.2}$$

Table 4.31: Contingency matrix for binary classification problems: True Negatives (TN), False Positives (FP), False Negatives (FN) and True Positives (TP)

		Detection	
		Negative	Positive
Real	Negative	TN	FP
	Positive	FN	TP

Table 4.32: Contingency matrix obtained testing the C4.5-based WAF performed on the CSIC 2010 HTTP dataset

		Detection	
		Negative	Positive
Real	Negative	23272	1045
	Positive	768	15625

Table 4.33: Detection rate (H), false alarm rate (F) and CID index of the C4.5-based WAF performed on the CSIC 2010 HTTP dataset

H	0.957
F	0.0468
CID	0.734

- False positive rate (F).

$$F = \frac{FP}{FP + TN} \tag{4.3}$$

- CID index [54] measures the amount of uncertainty of the input resolved once the IDS output is obtained, and takes into account the prevalence (B in the following formula) in the dataset besides the hit rate (H) and the false positive rate (F).

$$\begin{aligned} C_{id} = & - BHlog\frac{BH}{BH + HF} \\ & - B(1-H)log\frac{B(1-H)}{B(1-H) + (1-B)(1-F)} \\ & - (1-B)(1-F)log\frac{(1-B)(1-F)}{(1-B)(1-F) + B(1-H)} \\ & - (1-B)Flog\frac{(1-B)F}{(1-B)F + BH} \end{aligned} \tag{4.4}$$

Table 4.32 and Table 4.33 show the effectiveness of the C4.5-based WAF performed on the CSIC 2010 HTTP dataset. These metrics correspond to the detection of new HTTP traces, i.e. the two thirds of the dataset that were not used in the training phase of the WAF. We explain further how these traces are used when modeling the WAF in the next section. The WAF obtains pretty good results detecting the HTTP attacks included in the dataset and has a low false positive rate. Therefore, it is valid to apply our methodology, as we explain below.

4.5.3 Experimental Setting

Figure 4.13 shows a schematic view of the methodology that we use to look for evasions over a given WAF. The required inputs to the methodology are the WAF studied and the la-

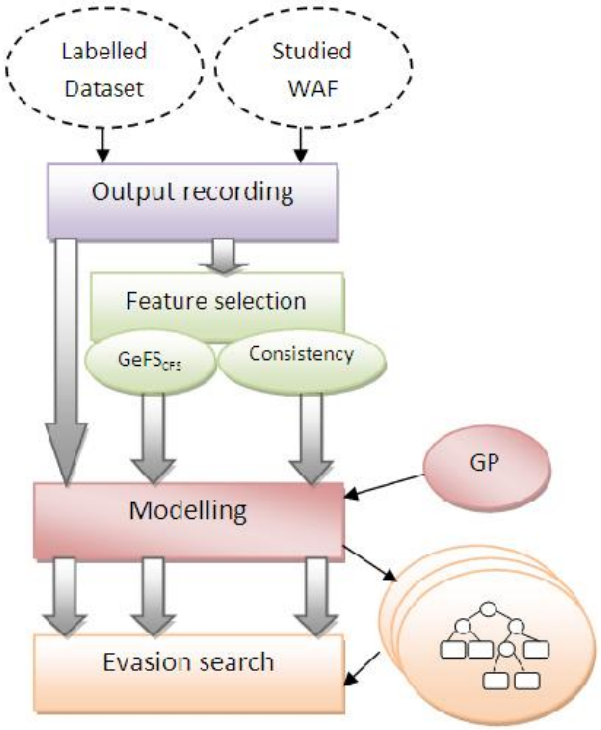


Figure 4.13: Scheme of the methodology followed.

beled dataset (dotted line circles in the figure) used to model it. At this point, labeled means that it is known whether each instance in the dataset is normal or attack. The methodology can be divided into 4 main sub-tasks: the output recording, the feature selection process, the modeling phase and the evasion search.

Output recording: As explained above, Table 4.32 and Table 4.33 show the effectiveness of the WAF implemented with the C4.5 algorithm when classifying new traces. We obtain the output given for each trace exposed to the WAF. We take note of this output and write it at the end of the trace to obtain a final trace with the following format:

$$F_1, F_2, F_3, ..., F_{117}, L, O$$

Here, each F_i is the i^{th} input feature (from the total of 117 features), L is the label indicating whether the trace is actually normal or intrusive and O is the output given by the WAF for this trace. Note that, if the label (L) and the output (O) have the same value, it is a correctly classified instance, otherwise it is a false positive (if L=0) or a false negative (if L=1). After this process the dataset contains information about how the WAF classifies the instances (i.e., its behaviour). This information is used in the modeling phase, as explained below.

Feature Selection: In order to optimize the modeling process and the evasion search, we apply feature selection. Our hypothesis is that looking for evasions over the model obtained from a subset of the most important features may be easier and faster than looking at the model obtained with the complete set of features. Therefore, we apply the GeFS measure for feature selection (see subsection 3.1.2). In order to select the appropriate GeFS instance ($GeFS_{CFS}$ or $GeFS_{mRMR}$), we analyse the statistical properties of the dataset

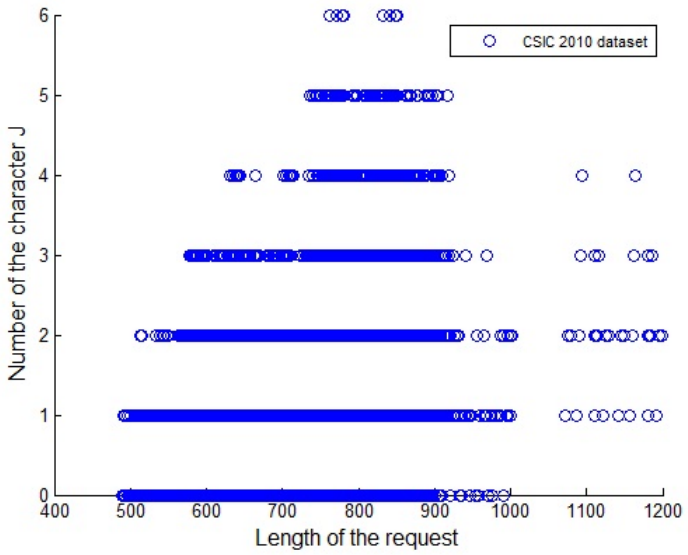


Figure 4.14: Sample of distributions of data points from the HTTP dataset CSIC 2010 with 117 extracted features

with the 117 constructed features to see whether there are linear or non-linear relationships between features (the $GeFS_{CFS}$ measure is recommended in the case where linear relationships exist, and the $GeFS_{mRMR}$ measure otherwise). For the statistical properties analysis, we first visualize the whole dataset in the two-dimensional space to get a plot matrix, where each element is the distribution of data points depending on, either the values of a feature and the output of the WAF, or the values of two features. Figure 4.14 is a sample of data point distributions of the features extracted from the HTTP dataset CSIC 2010. In the figures the axes represent, respectively, the length of the request vs. the number of the 1-gram 'z' and the length of the arguments vs. the number of letters in the argument's values. Blue points represent normal requests and red points, anomalous ones. As it can be seen in the figures, there are linear relations between the features of the dataset. To confirm our observations from the figures, we additionally calculate the correlation coefficients between the features. By observing these coefficients, we see that most of the features of the dataset do not have linear relationships to the label and that many features are linearly correlated to each other. In fact, more than 40% of the correlation values between features are greater than 0.09, which means that between those features there are linear relationships. Therefore, our conclusion is that the appropriate measure for selecting features from the HTTP dataset CSIC 2010 is $GeFS_{CFS}$.

In the tables 4.30, 4.34 and 4.37, the symbols $\star$ and $\diamond$ are used to point out the features selected by the $GeFS_{CFS}$ and Consistency measures from the whole set of features extracted from the CSIC 2010 dataset. Note that indeed many of the features selected are critical for the detection of Web attacks, like the ' character for instance, that is included in many SQL injection attacks.

Modeling: As previously mentioned, the modeling of the WAF is performed using Genetic Programming (GP). Table 4.35 shows the values used for the GP parameters. These values have been obtained using the cross-validation technique [67] and getting the combination of parameters that gives the best results. This technique can be summarized as follows :

Table 4.34: 28 features constructed using expert knowledge from the HTTP dataset CSIC 2010. ⋆ refers to features selected by the $GeFS_{CFS}$ measure and ◇ to features selected by the Consistency measure.

Features Extracted with Expert Knowledge	
Length of the request	⋆ ◇
Length of the path	⋆
Length of the arguments	
Length of the header "Accept"	◇
Length of the header "Accept-Encoding"	
Length of the header "Accept-Charset"	
Length of the header "Accept-Language"	◇
Length of the header "Cookie"	◇
Length of the header "Content-Length"	
Length of the header "Content-Type"	
Length of the Host	
Length of the header "Referer"	◇
Length of the header "User-Agent"	◇
Method identifier	
Number of arguments	
Number of letters in the arguments	⋆ ◇
Number of digits in the arguments	◇
Number of 'special' char in the arguments	⋆ ◇
Number of other char in the arguments	◇
Number of letters char in the path	
Number of digits in the path	⋆ ◇
Number of 'special' char in the path	◇
Number of other char in path	⋆
Number of cookies	◇
Number of distinct bytes	◇
Entropy	⋆ ◇
Number of keywords in the path	⋆
Number of keywords in the arguments	

1. Divide the entire training set into k smaller sets (called folds). Each fold contains different traces; therefore a trace is not repeated among different folds.

2. Establish a random value for each parameter. It is recommended to delimit the valid values to avoid incoherences (for example, an incoherent value is a number of generations lower than two).

3. With these parameter values, and for each fold k:

   a) Merge all the folds but k into a set.

   b) Train a model using as training set the previously merged one.

   c) Test the model using the fold k.

   d) Store the results to be processed later.

4. Return to step 2 and repeat the process as often as desired. To ensure enough variability in the values, we recommend to repeat the process at least $4n$ times, where n is the number of parameters tuned.

Table 4.35: GP (Genetic Programming) parameters used in the experiments

Name	Value
Number of generations	300
Maximal tree depth	15
Population size	1000
Tournament size	8
Crossover rate	90%
Mutation rate	10%

Table 4.36 describes the functions used in the GP algorithm. Most of them outputs a binary value. In order to obtain the models, we run the GP training phase using the second third of the dataset (the first one was used to train the WAF with the C4.5 algorithm). Finally, the models are tested using the last third of the dataset. This test measures the effectiveness of the models classifying data as the WAF did. We perform comparisons in terms of classification error (see Eq. 4.5).

$$class\_error = \frac{incorrectly\_classified\_events}{total\_events} \tag{4.5}$$

This double training-testing phase is repeated 7 times in order to obtain a statistically significant measure which does not depend on the initial random seed. Therefore, as we obtain 7 different models, in Section 4.5.4 we provide the test accuracy for both the best model and the average of these seven models. These 7 experiments are repeated in 3 cases:

1. With the overall set of features (117)
2. With the 47 features selected by $GeFS_{CFS}$
3. With the 54 features selected by Consistency.

We compare the accuracy of the models obtained and discuss the advantages of selecting features before modeling the WAF. As remarked above, our hypothesis is that looking for evasions over the models obtained from a subset of the most important features may be easier or faster than looking at models obtained with the complete set of features. This hypothesis is supported by the results, as we show in Section 4.5.4.

Evasion Search : The evasion search is made by using a special brute-force approach. This search consists of modifying one by one the variables (features) placed at the leaves of the models. These are the features that the original WAF takes into account in the detection process. The objective is to convert true positives of the WAF into false negatives (see Table 4.31). Therefore, for each feature in the leaves of the models, we change its value in every attack trace that was detected as such, generating a new dataset of modified attack traces. This modified dataset is given as input to the original studied WAF to check whether new false negative appears. Each modified attack that is not detected by the WAF is considered an evasion candidate. We say candidate because before considering it a real evasion of the WAF we have to check two requirements:

1. Data obtained with the change must represent valid HTTP traffic.
2. Modifications are performed only in attack traces. It is necessary to confirm that the modified trace still represents the attack.

Table 4.36: List of functions used with GP (Genetic Programming)

Name	Description of Functions
ADD	Addition of two numbers
AND	Logic AND operation between two numbers
GREATER	Compares two values. Returns 1 if the first is greater, otherwise 0
LEAST	Compares two values. Returns 1 if the first is lower, otherwise 0
MULT	Multiplication of two numbers
OR	Logic OR operation between two numbers
MAX	Returns the maximum of two values
MIN	Returns the minimum of two values
RL	Bitwise left rotation in one position of a value
NOT	Logic NOT operation of a value

Table 4.37: Sorted list of the 10 features with more frequency of appearance in the GP models obtained with the overall set of features. The last two columns indicates whether each feature has been selected by $GeFS_{CFS}$ and Consistency respectively

Feature	Frequency	$\mathbf{GeFS_{CFS}}$	Consistency
Number of special char. in arguments	128	⋆	◇
Number of "-"	106	⋆	
Number of "A"	105	⋆	◇
Number of "1"	76	⋆	
Number of "U"	55	⋆	
Number of arguments	50		
Number of "k"	49		
Number of "∼"	42	⋆	◇
Number of "I"	41	⋆	◇
Number of "e"	32		◇

4.5.4 Experimental Results

The first modeling of the WAF has been carried out using the overall set of features. The resulting leaves of the tree models indicate which are the most relevant features considered by the WAF in the detection. Table 4.37 indicates the 10 features that appear with major frequency (in descending order) in the leaves of these models obtained by GP when used the complete set of features. For each of the features, the table shows whether it has been selected by any of the selection methods studied. We can see that five of the most frequent features in the models were also indicated by $GeFS_{CFS}$ (two of them were also selected by Consistency). This means that GP is able to identify the most relevant features to be used in this scenario in a similar way that the $GeFS_{CFS}$ does.

We have also modelled the WAF using the subset of 47 features selected in a previous step by $GeFS_{CFS}$ and the 54 subset selected by Consistency. Using this much smaller subset of features, we have obtained models that classify nearly equal to the model obtained using the overall subset of features (see Figure 4.15). As explained above, the use of GP may perform a similar feature selection. However, the GP process spent a major amount of resources, so if saving time or resources is critical it is better to previously select the fea-

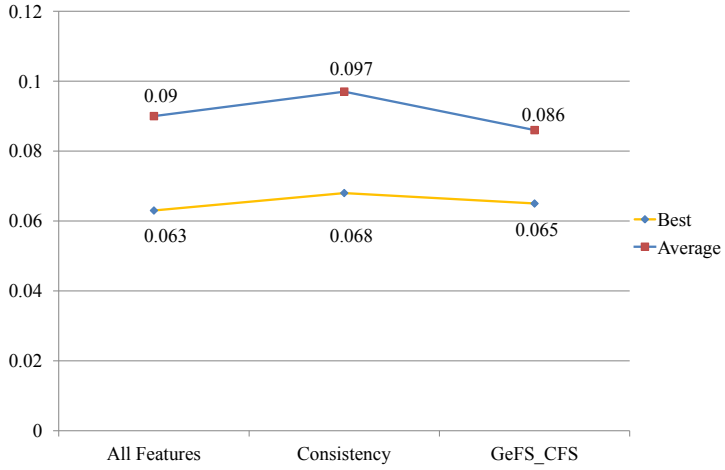


Figure 4.15: Classification error of both the best GP individual and the average of the GP individuals using the complete dataset (All Features), the subset selected by $GeFS_{CFS}$ and the subset selected by Consistency

tures. Therefore, when the only goal is modeling, it seems better to use the overall set of features, as by the nature of GP, it makes an internal selection that nearly coincides with the two other methods. However, in this work our aim is to look for evasions. In the models obtained with a previous feature selection, the number of different variables in the leaves of the trees is lower than in the models obtained using all the features. That means that, when looking for evasions it is better to directly focus on the principal features indicated, so it is easier by looking at the $GeFS_{CFS}$ and Consistency models. In fact, the query is faster looking through these models than looking up on the models obtained without feature selection. Concretely, a 21 % of the time is reduced using the $GeFS_{CFS}$ model and a 15 % is reduced using the Consistency model.

We have obtained around fifty thousand evasion candidates on the dataset, that is, cases where just a feature change makes the WAF to fail the detection. For some traces, we have found several ways to perform the evasion, that is, several candidates. However, for each of these candidates, we have to analyse if it is still an attack and the viability of performing the corresponding change from the HTTP protocol point of view (for instance, it makes no sense to set the "length of the path" feature to zero). In order to give a semantic result of how an evasion may succeed, we have manually analysed some of the candidates. Below we show two illustrative examples of evasion of an SQL Injection attack and an XSS attack.

Figure 4.16 shows an HTTP packet that encapsulates a classical SQL Injection attack. Analysing the models we have realized that part of the detection is based on the length of the path (feature number 2). Originally, the path had 45 bytes:

http://localhost:8080/store1/members/edit.jsp

By brute force search on this feature we have seen that forcing this length path to be greater than or equal to 50, this attack evades the detection. According to the HTTP protocol, this modification can be performed by adding null information, such as "../../". This change also causes that the attack succeeds. The path of the changed packet is:

http://localhost:8080/store1/../../members/edit.jsp

```
GET /store1/members/edit.jsp?mode=register&login=sawyers&password=encUmb2ad3'; DROP
TABLE users; SELECT * FROM data WHERE name LIKE '%&name=Max&surname=Sjostrand
Ampurias&email=tamburi@hispadis.ar&id=30293636Z&direction=Victoria Road Avenue 122
4?A&city=London&cp=48220&country=England&ntc=8898020239162472&B1=Register HTTP/1.1


User-Agent: Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.8 (like Gecko)
Pragma: no-cache
Cache-control: no-cache
Accept:
text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/pn
g,*/*;q=0.5
Accept-Encoding: x-gzip, x-deflate, gzip, deflate
Accept-Charset: utf-8, utf-8;q=0.5, *;q=0.5
Accept-Language: en
Host: localhost:8080
Cookie: JSESSIONID=A1471DF9948232522D0C9431CBA67DAF
Connection: close


');
```

Figure 4.16: HTTP Packet encapsulating a simple SQL Injection attack

```
POST http://localhost:8080/store1/members/edit.jsp HTTP/1.1

User-Agent: Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.8 (like Gecko)
Pragma: no-cache
Cache-control: no-cache
Accept:
text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Encoding: x-gzip, x-deflate, gzip, deflate
Accept-Charset: utf-8, utf-8;q=0.5, *;q=0.5
Accept-Language: en
Host: localhost:8080
Cookie: JSESSIONID=400BDFD43C5F8B30E041F61A078361F7
Content-Type: application/x-www-form-urlencoded
Connection: close
Content-Length: 335

mode=register&login=gib&password=3L29stRE&name=Yaravi&surname=Panella
Villasev&email=switzer@greatinvents.pt&dni=04831932T&address=San Lorenzo, 125&city=Alegria-Dulantzi&
cp=34170& province=%3CSCRIPT%3Ealert%28%22Paros%22%29%3B%3C%2FSCRIPT%3E&ntc=30291475586383096B1
=Register');
```

Figure 4.17: HTTP Packet encapsulating a simple Cross-Site Scripting attack

It is also possible to change the slash character (/) by its URI codification (%2f):

http://localhost:8080%2fstore1%2fmembers%2fedit.jsp

These two modifications provoke the WAF to fail the detection, that is, the SQL Injection attack evades the detection.

Another example is shown in Figure 4.17. The detection of this XSS attack by the WAF takes into account the number of hyphen characters (-) in the method, path and arguments. The packet corresponds to a POST request where the arguments are given in the DATA field. As we can see, the only hyphen in the packet is in the name of the city. We have realized that by adding 4 more hyphens we evade the detection. This can be done by faking part of the information given as arguments. For instance, we can set the surname to be Panella-Villasev (realize that, for example in Spain, people usually have two surnames, so a hyphen is normally used to avoid problems in databases that are expecting just one word), another one into the address (San-Lorenzo), an extra one into the city (Alegria–Dulantzi) and another one into the e-mail address (great-inventions). These changes affect only the fields of the parameters, so the complete packet still represents valid HTTP traffic.

In summary, this section has applied the generic-feature-selection (GeFS) measure to simplify the evasion process of Web Application Firewalls (WAFs), thus providing an efficient testing method of WAFs. The experiment was conducted on the CSIC 2010 dataset. The experimental results showed that the GeFS measure provided a more accurate and simpler modelling of WAFs than the using of full-set of features and the Consistency measure, thus speeding up the evasion search. In particular, the total amount of CPU time spent when using all features is 3 hours and 20 minutes, whereas using GeFS measure reduced the time by 21% (2 hours and 30 minutes of CPU time).

Chapter 5

Conclusions

A conclusion is the place where you get tired of thinking.

ARTHUR BLOCH

The theoretical and empirical findings are discussed in detail: new reliable machine learning methods via an operational research approach and their applications to information security problems (Section 5.1). Future work on the reliability of machine learning based intrusion detection systems in a changing or an adversarial network environment are provided (Section 5.2).

5.1 Summary of Findings

This dissertation is mainly divided in two parts: reliability analysis of a statistical pattern recognition system with new proposed reliable algorithms and applications to information security problems. We summarize each part in turn.

5.1.1 New Reliable Machine Learning Algorithms

The first part of this dissertation is about analyzing the reliability of a statistical pattern recognition system. First, we studied the main factors that affect the reliability in the feature selection process: the choice of feature selection methods and the search strategies for the relevant features. We introduced a formal definition of a reliable feature-selection process. The definition provides formal measurements of reliability in feature-selection, i.e., the **steadiness** of a classifier's performance and the **consistency** in search for relevant features. We proposed new methods to address the main causes of unreliable feature-selection process. In particular, we introduced a new methodology of determining appropriate instances from a class of feature-selection methods. We called this class a generic-feature-selection (GeFS) measure. We also proposed a new search approach that ensures the globally optimal feature subset by means of the GeFS measure. The new search approach is based on solving a mixed 0-1 linear programming (M01LP) problem by means of the branch-and-bound algorithm.

Moreover, we analyzed the influence of the *over-selecting* phenomenon, which is the situation when the features selected from the training data are quite different from the representative features of the testing data, on the *over-fitting* phenomenon of machine learning algorithms. We proposed a new framework to address principal causes of *over-selecting*, thus reducing the chance of *over-fitting* and providing reliable results. Our new framework that we called Ensemble Feature Selection measure (EnFS), allows the consideration of many statistical properties of a given data set at the same time by combining many feature selection methods used in the filter model. From the chosen feature selection measures, a new combined measure is constructed. We also introduced a new search algorithm that ensures the globally optimal feature subsets by means of the constructed measure. Similar to the case of the generic feature selection measure, this new search approach is based on

solving a mixed 0-1 linear programming (M01LP) problem by means of the branch-and-bound algorithm.

Second, we proposed to generalize the traditional Lp-norm Support Vector Machines (Lp-SVMs) into a new general Lp-norm Support Vector Machines (GLp-SVMs) that takes into account all possible feature subsets. We represented the GLp-SVM as a mixed 0-1 nonlinear programming problem (M01NLP). We proved that solving the new proposed M01NLP optimization problem results in a smaller error penalty and enlarges the margin between two support vector hyper-planes, thus possibly giving more reliable classification results and a better generalization capability of SVMs than solving the traditional Lp-SVMs. Moreover, by following the new general formulation we can easily integrate expert knowledge into the GLp-SVMs, thus the classification results are much more reliable.

Third, we proposed a new initialization method for K-means clustering and a new search method for optimal K-means with $K = 2$. In fact, we casted the K-means problem into a mixed 0-1 linear programming probblem, which can be solved by using the D.C. (Difference of convex functions) algorithm. This approach provides more accurate and more reliable K-means clustering than many heuristic approaches do.

5.1.2 Applications for Intrusion Detection Systems

In the second part of this dissertation, we applied our new machine learning algorithms in building reliable intrusion detection systems to detect attacks in three different layers: (1) application layer, (2) network layer and (3) operating system layer.

Ad. 1 In the application layer, we applied the generic-feature-selection (GeFS) measure to select important features for Web attack detection. We conducted the experiment on the CSIC 2010 and ECML/PKDD 2007 datasets. Statistical properties of the datasets were first analyzed. Based on this analysis, the $GeFS_{CFS}$ measure and the $GeFS_{mRMR}$ measure were chosen for selecting features from the CSIC 2010 dataset and the ECML/PKDD 2007 dataset, respectively. The detection accuracies obtained after the feature selection by means of four different classifiers were tested. The experiments show that by choosing appropriate instances of the GeFS measure, 63% of irrelevant and redundant features were removed from the original dataset, while reducing only 0.12% the detection accuracy of WAFs. At the same time, the new proposed methods provide reliable feature-selection results and outperform the heuristic approaches in term of reliability.

Ad. 2 In the network layer, first we applied the generic-feature-selection (GeFS) measure to select important features for network-based intrusion detection systems. The benchmarking KDD CUP 1999 dataset was used in our experiment. We compared our new GeFS measure with various existing feature-selection methods, such as the SVM-based method, the Markov-blanket-based method and so on. Experimental results showed that our approach outperforms the existing ones by removing much more redundant features and still keeping the classification accuracies or even getting better performances. Second, we demonstrated the advantage of the GeFS measure in providing the most important features for the reliable detection of botnet malware.

Ad. 3 In the operating system layer, we analyzed the characteristics of the attack and normal behavior in system calls, such as file system modifications, user logons and so on. We applied the new proposed general Lp-norm support vector machines (GLp-SVMs) for host-based intrusion detection systems. We conducted an experiment on the UNM and the MIT Lincoln Lab datasets. The experiment showed that the GLp-SVMs provided more reliable detection results than the ones obtained by the traditional Lp-norm SVM. Because the GLp-SVMs gave better generalization capabilities, while in many cases selecting fewer features. In fact, the GLp-SVMs improved the classification accuracy of the traditional Lp-SVMs by more than 13.49% and surprisingly, in some cases the GLp-SVMs used only one important feature.

Finally, we applied the generic-feature-selection (GeFS) measure to simplify the evasion process of Web Application Firewalls (WAFs), thus providing an efficient testing method of WAFs. We conducted the experiment on the CSIC 2010 dataset. The experimental results showed that the GeFS measure provided a more accurate and simpler modelling of WAFs than the using of full-set of features and the Consistency measure, thus speeding up the evasion search. In particular, the total amount of CPU time spent when using all features is 3 hours and 20 minutes, whereas using GeFS measure reduced the time by 21% (2 hours and 30 minutes of CPU time).

5.2 Future Work

In terms of future work, our vision is to build highly-reliable machine learning based intrusion detection systems in the changing and adversarial network environments. This section outlines various directions for this vision.

5.2.1 Adaptive Intrusion Detection Systems

Network environments become more and more diverse with the presence of many different network protocols, services, applications and so on. With this diversity, many different types of attacks appear and target at a computer or a network everyday. A single type of intrusion detection systems (IDSs), which has its own advantages and disadvantages, seems to be insufficient to detect all the attacks. For example, misuse IDSs, such as SNORT [117], can only detect attacks, which are described in the databases. This approach is known as a signature-based IDS. The anomaly IDSs [78] can detect novel attacks based on significant deviations of the observed activities from the established profiles of the legitimate users. However, at the same time these IDSs generate more false positives than the misuse IDSs. In addtion, it is difficult to produce effective models of legitimate patterns by expert knowledge. Several data-driven IDSs by using machine learning [144] are developed to cope with these problems. But this approach strongly depends on the utilized classifiers and the IDSs might perform well in a particular network environment with particular attack types. For example, the Decision-Tree-based IDS detects well the denial-of-service attacks, but it fails to identify most of the user-to-root and remote-to-local attacks [119]. Of course this happens with the particular KDD CUP 1999 dataset [72], yet it might likely happen in many other datasets. This phenomenon is according to No-Free-Lunch Theorem [142, 143] that says there are no universal classifiers for every kind of data.

In the adversarial network environments, the situation is even worse as attackers might launch and change the types of attacks every time, thus the streams of the data are unpredictable. Since we don't know which types of attacks are coming next, the primary difficulty lays on selecting of the best IDS at a certain time. In our scenario, we assume that each IDS has its own favorite types of attacks to detect.

One might think to run several IDSs at the same time and then either select the IDS that works well on average, or select the one that provides the best performance in the previous data that is believed to be representative. However, this manual selection costs a lot of efforts and usually are incorrect since the stream of the data in the heterogeneous and adversarial network environments are changed all the time.

One might also think to automatically combine the outputs of the IDSs. A simplest way to combine the IDSs' outputs is the majority voting. At any time, the final decision will be the prediction of the most frequent output. This approach is very efficient, but incorrect in many cases since the majority can be wrong. Several efficient fusion techniques for ensemble IDSs (see, for example, [39]) were proposed. However, there are no theoretical guarantees about the ability of the ensemble IDSs to follow the best IDS or to follow the sequence of the best IDSs in different segments of the data. This ability is understood as

the adaptability of the IDSs. Another combination method named Hedge/Boosting [51], which is a very efficient online learning framework. It has also the theoretical guarantee that says the combination of IDSs will have the performance close to the performance of the best IDS for the whole period of time T. However, this approach is sensitive in the adversarial environments, in which attackers intend to destroy the superiority of the best IDS by changing the attacks all the time. After T times, all the baseline IDSs will perform badly the same, thus the combination by means of the Hedge/Boosting might not be useful. This adversarial activity targeted at the ensemble IDSs can be included into the Exploratory category [20] for online machine learning.

In the future work, we propose to apply another efficient online learning framework, which is the mixing algorithm introduced by Bousquet&Warmuth (2002) [29], to combine different outputs of the baseline IDSs. We call the new proposed method an Adaptive Intrusion Detection System (A-IDS). The mixing algorithm consists of two steps: the *Loss Update* and the *Mixing Update*. The *Loss Update* is essentially the Hedge algorithm [51]. The beauty of the *Mixing Update* is that it efficiently remembers the IDSs that perform well in the past through the weight values and quickly recover them to deal with their favorite data at the current moment. Therefore, the A-IDS quickly adapts to the changing environments, such as heterogeneous and adversarial network environments. More details on how and why the algorithm works are given in [29].

5.2.2 Robust Intrusion Detection Systems

In an adversarial network environment where a machine learing-based intrusion detection system (ML-IDS) is deployed, attackers can adaptively manipulate the data to mislead the training process of the ML-IDS. This manipulation can be done by adding feature noise, such as unimportant patterns or features, to the flow of network data which will be captured for training ML-IDSs. For example, attackers can attach data noise to a network where a honeypot is running to collect attack samples. Moreover, in many cases attackers bypass the ML-IDS or the ML-IDS accidentally misclassifies some attack and normal samples. These introduce the label noise into the training datasets for the ML-IDS.

In the presence of training data noise (including feature and label noise), the ML-IDS might learn wrong patterns, thus providing unreliable classification results. Therefore, the robustness of ML-IDSs to ignore the data noise in the training phase is necessary. In the following, we describe two potential approaches for robust intrusion detection systems.

First, feature selection helps to look for important patterns while skipping the data noise. In the future work, we propose to apply and test our new reliable feature-selection algorithms (see Chapter 3) for robust intrusion detection systems in dealing with training data noise. Second, it has been recently shown that the convexity property of various machine learning algorithm, such as support vector machines and logistic regression algorithm, makes them not robust to the data noise [76]. The use of non-convex machine learning methods, such as the t-logistic regression [44], might solve the problem. In the future work, we study this approach for robust intrusion detection systems.

Appendix A

Anscombe's Quartet

Below is four datasets (I, II, III and IV) of the Anscombe's quartet. These datasets were generated by the statistician Francis Anscombe in 1973 [19] to show the importance of visualization of a dataset before statistically analysing it and the effect of outliers on statistical properties.

Anscombe's quartet

I		II		III		IV	
x	y	x	y	x	y	x	y
10.0	8.04	10.0	9.14	10.0	7.46	8.0	6.58
8.0	6.95	8.0	8.14	8.0	6.77	8.0	5.76
13.0	7.58	13.0	8.74	13.0	12.74	8.0	7.71
9.0	8.81	9.0	8.77	9.0	7.11	8.0	8.84
11.0	8.33	11.0	9.26	11.0	7.81	8.0	8.47
14.0	9.96	14.0	8.10	14.0	8.84	8.0	7.04
6.0	7.24	6.0	6.13	6.0	6.08	8.0	5.25
4.0	4.26	4.0	3.10	4.0	5.39	19.0	12.50
12.0	10.84	12.0	9.13	12.0	8.15	8.0	5.56
7.0	4.82	7.0	7.26	7.0	6.42	8.0	7.91
5.0	5.68	5.0	4.74	5.0	5.73	8.0	6.89

Nomenclature

IDS Intrusion Detection System

NFL No-Free-Lunch Theorems

SVM Support Vector Machine

CFS Correlation Feature Selection

mRMR Minimal Redundancy Maximal Relevance

GeFS Generic Feature Selection

EnFS Ensembel Feature Selection

GLp-SVM General Lp-norm Support Vector Machine

CO Convex Optimization

NCO Non-convex Optimization

M01LP Mixed 0-1 Linear Programming

PM01FP Polynomial Mixed 0-1 Fractional Programming

DCA Difference of Convex Functions Algorithm

WAF Web Application Firewall

NIDS Network-based Intrusion Detection System

HIDS Host-based Intrusion Detection System

GP Genetic Programming

GA Genetic Algorithm

BF Best First Search

C4.5 Decision Tree

CART Classification And Regression Tree

Malware Malicious Software

TN True Negative

FP False Positive

FN False Negative

TP True Positive

Bibliography

[1] Dmi classification software. http://www.cs.wisc.edu/dmi/. Last visited: 01.01.2012. 88

[2] Http dataset csic 2010. http://www.iec.csic.es/dataset/. Last visited: 01.06.2012. 59

[3] Knowles, d. w32.spybot.worm. http://www.symantec.com/security_response/writeup.jsp?docid=2003-053013-5943-99. Last visited:09.02.2011. 81

[4] O'connor, j. trojan.anserin. http://www.symantec.com/security_response/writeup.jsp?docid=2005-112315-0608-99. Last visited:09.02.2011. 81

[5] Offensive computing. http://www.offensivecomputing.net/. Last visited: 01.01.2011. 81

[6] Packet storm. http://packetstormsecurity.org/. Last visited: 01.01.2011. 81

[7] Pendriveapps. http://www.pendriveapps.com/. Last visited: 05.05.2011. 81

[8] Top 10 malware registry launchpoints. http://www.f-secure.com/weblog/archives/00001207.html. Last visited:05.05.2011. 83

[9] Unm (university of new mexico) audit data. http://www.cs.unm.edu/ immsec/systemcalls.htm. Last visited: 01.03.2012. 87, 88

[10] Vmware workstation. http://www.vmware.com/products/workstation/. Last visited:01.02.2011. 82

[11] Vx heavens. http://vx.netlux.org. Last visited: 01.01.2011. 81

[12] AGRAWAL, R., IMIELIŃSKI, T., AND SWAMI, A. Mining association rules between sets of items in large databases. *Proceedings of the 1993 ACM SIGMOD international conference on Management of Data 22*, 2 (June 1993), 207–216. 11

[13] AGRAWAL, R., AND SRIKANT, R. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases* (1994), VLDB '94, pp. 487–499. 11

[14] AMERICAN PSYCHOLOGICAL ASSOCIATION. *The Standards for Educational and Psychological Testing*. AERA Publications, 1999. 33

[15] AN, L. T. H., MINH, L. H., AND TAO, P. D. Optimization based dc programming and dca for hierarchical clustering. *European Journal of Operational Research 183*, 3 (2007), 1067–1085. 27, 28

[16] AN, L. T. H., AND TAO, P. D. The dc (difference of convex functions) programming and dca revisited with dc models of real world nonconvex optimization problems. *Annals of Operations Research 133*, 1-4 (2005), 23–46. 27, 28, 29, 54

[17] AN, L. T. H., TAO, P. D., AND MUU, L. D. A combined d.c. optimization - ellipsoidal branch-and-bound algorithm for solving nonconvex quadratic programming problems. *Journal of Combinatorial Optimization 2*, 1 (1998), 9–28. 27, 54

[18] AN, L. T. H., TAO, P. D., AND YEN, N. D. Properties of two dc algorithms in quadratic programming. *Journal of Global Optimization 49*, 3 (2011), 481–495. 27, 28

[19] ANSCOMBE, F. J. Graphs in Statistical Analysis. *The American Statistician 27*, 1 (Feb. 1973), 17–21. 40, 105

[20] BARRENO, M., NELSON, B., JOSEPH, A. D., AND TYGAR, J. D. The security of machine learning. *Machine Learning 81*, 2 (2010), 121–148. 104

[21] BAYER, U., HABIBI, I., BALZAROTTI, D., KIRDA, E., AND KRUEGEL, C. A view on current malware behaviors. In *In Proceedings of the 2nd USENIX conference on Largescale exploits and emergent threats: botnets, spyware, worms, and more, LEET'09, 8-8, Berkeley, CA, USA. USENIX Association.* (2009). 82

[22] BAYER, U., MOSER, A., KRÜGEL, C., AND KIRDA, E. Dynamic analysis of malicious code. *Journal in Computer Virology 2*, 1 (2006), 67–77. 81, 82

[23] BECHER, M., AND BECHER. *Web Application Firewalls*. VDM Publishing, 2007. 58

[24] BENNETT, K. P., BENNETT, P., AND PARRADO-HERNANDEZ, E. The interplay of optimization and machine learning research. *Journal of Machine Learning Research 7* (2006), 12651281. 21

[25] BENNETT, K. P., AND MANGASARIAN, O. L. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software 1*, 1 (1992), 23–34. 50

[26] BERG, P. E. Behavior-based Classification of Botnet Malware, Master thesis, Gjøvik Univeristy College, 2011. 7, 57, 81

[27] BERTSEKAS, D. P. *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press, 1982. 18

[28] BOTTOU, L. Stochastic learning. In *Advanced Lectures on Machine Learning*, O. Bousquet and U. von Luxburg, Eds., Lecture Notes in Artificial Intelligence, LNAI 3176. Springer Verlag, Berlin, 2004, pp. 146–168. 18

[29] BOUSQUET, O., AND WARMUTH, M. K. Tracking a small set of experts by mixing past posteriors. *Journal of Machine Learning Research 3* (Mar. 2003), 363–396. 104

[30] BRADLEY, P., AND MANGASARIAN, O. L. Feature selection via concave minimization and support vector machines. In *Machine Learning Proceedings of the Fifteenth International Conference(ICML 98* (1998), Morgan Kaufmann, pp. 82–90. 47, 48

[31] BRADLEY, P. S., AND FAYYAD, U. M. Refining initial points for k-means clustering. In *Proceedings of the Fifteenth International Conference on Machine Learning* (1998), ICML '98, pp. 91–99. 52

[32] CAVNAR, W., AND TRENKLE, J. N-gram-based text categorization. In *Proceedings of Symposium on Document Analysis and Information Retrieval (SDAIR)* (1994), pp. 161–175. 90

[33] CHANG, C.-T. An efficient linearization approach for mixed-integer problems. *European Journal of Operational Research 123*, 3 (2000), 652–659. 34, 37, 38, 40, 44, 45, 48, 50

[34] CHANG, C.-T. On the polynomial mixed 0-1 fractional programming problems. *European Journal of Operational Research 131*, 1 (2001), 224–227. 34, 37, 38, 40, 44, 45, 48, 50

[35] CHEBROLU, S., ABRAHAM, A., AND THOMAS, J. P. Feature deduction and ensemble design of intrusion detection systems. *Computers & Security 24*, 4 (June 2005), 295–307. xiii, 74, 75, 76

[36] CHEN, Y., LI, Y., CHENG, X., AND GUO, L. Survey and taxonomy of feature selection algorithms in intrusion detection system. In *Information Security and Cryptology (Inscrypt)* (2006), pp. 153–167. 69

[37] CLAUSEN, J. Branch and bound algorithms-principles and examples, 2003. xi, 26, 27

[38] CORCHADO, E., AND HERRERO, A. Neural visualization of network traffic data for intrusion detection. *Applied Soft Computing 11*, 2 (Mar. 2011), 2042–2056. 58

[39] CORONA, I., GIACINTO, G., AND ROLI, F. Intrusion detection in computer systems using multiple classifier systems. In *Supervised and Unsupervised Ensemble Methods and their Applications*, O. Okun and G. Valentini, Eds., vol. 126 of *Studies in Computational Intelligence*. Springer Berlin / Heidelberg, 2008, pp. 91–113. 10.1007/978-3-540-78981-9_5. 103

[40] CORTES, C., AND VAPNIK, V. Support-vector networks. *Machine Learning 20*, 3 (1995), 273–297. 18, 47, 48, 74

[41] CRESCENZO, G. D., GHOSH, A., AND TALPADE, R. Towards a theory of intrusion detection. In *10th European Symposium on Research in Computer Security ESORICS* (2005), pp. 267–286. 64

[42] DAI, H. A study on reliability in graph discovery. In *Proceedings of the Sixth IEEE International Conference in Data Mining-Workshops* (2006), pp. 775–779. 33

[43] DENNING, D. E. An intrusion-detection model. *IEEE Transactions on Software Engineering, Special issue on computer security and privacy 13*, 2 (Feb. 1987), 222–232. 1

[44] DING, N., AND VISHWANATHAN, S. V. N. t-logistic regression. In *Neural Information Processing Systems* (2010), pp. 514–522. 104

[45] DUDA, R. O., HART, P. E., AND STORK, D. G. *Pattern Classification (2nd Edition)*, 2 ed. Wiley-Interscience, Nov. 2001. 13, 19, 47, 52, 62, 75, 76, 79, 81, 85

[46] ESKIN, E., ARNOLD, A., PRERAU, M., PORTNOY, L., AND STOLFO, S. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. In *Applications of Data Mining in Computer Security* (2002), Kluwer. 1

[47] FAYYAD, U. M., AND IRANI, K. B. Multi-interval discretization of continuous-valued attributes for classification learning. In *International Joint Conferences on Artificial Intelligence* (1993), pp. 1022–1029. 14

[48] FIRDAUSI, I., LIM, C., ERWIN, A., AND NUGROHO, A. S. Analysis of machine learning techniques used in behavior-based malware detection. In *Proceedings of the 2010 Second International Conference on Advances in Computing, Control, and Telecommunication Technologies* (2010), ACT '10, pp. 201–203. 81

[49] FIRDAUSI, I., L. C. E. A. Analysis of machine learning techniques used in behavior-based malware detection. In *In Advances in Computing, Control and Telecommunication Technologies (ACT)* (2010). 82

[50] FODOR, I. K. A survey of dimension reduction techniques. Tech. rep., U.S. Department of Energy, June 2002. 19

[51] FREUND, Y., AND SCHAPIRE, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences 55*, 1 (Aug. 1997), 119–139. 104

[52] GHISELLI, E. *Theory of psychological measurement*. McGraw-Hill, 1964. 13

[53] GHOSH, A. K., AND SCHWARTZBARD, A. A study in using neural networks for anomaly and misuse detection. In *Proceedings of the 8th conference on USENIX Security Symposium - Volume 8* (1999), SSYM'99, pp. 12–12. 1

[54] GU, G., FOGLA, P., DAGON, D., LEE, W., AND SKORIĆ, B. Measuring intrusion detection capability: an information-theoretic approach. In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security* (New York, NY, USA, 2006), ASIACCS '06, ACM, pp. 90–101. 92

[55] GU, G., FOGLA, P., DAGON, D., LEE, W., AND SKORIC, B. Towards an information-theoretic framework for analyzing intrusion detection systems. In *Proceedings of the 11th European conference on Research in Computer Security* (2006), ESORICS'06, pp. 527–546. 64

[56] GUYON, I., GUNN, S., NIKRAVESH, M., AND ZADEH, L. A. *Feature Extraction: Foundations and Applications (Studies in Fuzziness and Soft Computing)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. 10, 12, 13, 32, 34, 43, 48, 58, 65, 75, 78

[57] GUYON, I., WESTON, J., BARNHILL, S., AND VAPNIK, V. Gene selection for cancer classification using support vector machines. *Machine Learning 46*, 1-3 (Mar. 2002), 389–422. 13, 47

[58] HALL, M. A. Correlation-based feature selection for machine learning. Tech. rep., University of Waikato, 1998. 13

[59] HALL, M. A. Correlation-based feature selection for discrete and numeric class machine learning. In *International Conference on Machine Learning* (2000), pp. 359–366. 4, 13, 32, 34, 35, 42, 43, 58, 85

[60] HOLMSTRM, K. The tomlab optimization environment in matlab, 1999. 71, 79, 88

[61] HUANG, C., AND MORAGA, C. A diffusion-neural-network for learning from small samples. *Int. J. Approx. Reasoning 35*, 2 (2004), 137–161. 47

[62] JAIN, A., DUIN, R., AND MAO, J. Statistical pattern recognition: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*, 1 (jan 2000), 4 –37. xi, 2

[63] JAIN, A. K. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters 31*, 8 (2010), 651–666. 4, 52, 54

[64] KANG, D.-K., FULLER, D., AND HONAVAR, V. Learning classifiers for misuse detection using a bag of system calls representation. In *Intelligence and Security Informatics, IEEE International Conference on Intelligence and Security Informatics, ISI* (2005), pp. 511–516. 87

[65] KANUNGO, T., MOUNT, D. M., NETANYAHU, N. S., PIATKO, C. D., SILVERMAN, R., AND WU, A. Y. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24* (2002), 881–892. 52

[66] KHAN, L., AWAD, M., AND THURAISINGHAM, B. A new intrusion detection system using support vector machines and hierarchical clustering. *The VLDB Journal 16*, 4 (Oct. 2007), 507–521. 17

[67] KOHAVI, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International Joint Conferences on Artificial Intelligence* (1995), pp. 1137–1145. 94

[68] KRUEGEL, C., VIGNA, G., AND ROBERTSON, W. A multi-model approach to the detection of web-based attacks. *Comput. Netw. 48*, 5 (Aug. 2005), 717–738. 58

[69] KUHN, H., AND TUCKER, A. Nonlinear programming. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability* (1951), J. Neyman, Ed., University of California Press, Berkeley, California, pp. 481–492. 18

[70] KUKAR, M., AND KONONENKO, I. Reliable classifications with machine learning. In *13th European Conference on Machine Learning* (2002), pp. 219–231. 3

[71] LEE, W., AND STOLFO, S. J. A framework for constructing features and models for intrusion detection systems. *ACM Transactions on Information and System Security (TISSEC) 3*, 4 (Nov. 2000), 227–261. xiii, 66, 67, 68, 69, 77

[72] LEE, W., STOLFO, S. J., AND MOK, K. W. A data mining framework for building intrusion detection models. *IEEE Symposium on Security and Privacy 0* (1999), 0120. 6, 66, 70, 77, 103

[73] LIPPMANN, R. P., GRAF, I., WYSCHOGROD, D., WEBSTER, S. E., WEBER, D. J., AND GORTON, S. The 1998 DARPA/AFRL Off-Line Intrusion Detection Evaluation. In *In First International Workshop on Recent Advances in Intrusion Detection (RAID)* (Louvain-la-Neuve, Belgium, 1998). 6, 65, 87, 88, 90, 91

[74] LIU, H., AND MOTODA, H. *Computational Methods of Feature Selection (Chapman & Hall/Crc Data Mining and Knowledge Discovery Series)*. Chapman & Hall/CRC, 2007. 12, 32, 48, 58

[75] LLOYD, S. P. Least squares quantization in PCM. *IEEE Transactions on Information Theory IT-28*, 2 (Mar. 1982), 129–137. 20

[76] LONG, P. M., AND SERVEDIO, R. A. Random classification noise defeats all convex potential boosters. *Machine Learning 78*, 3 (Mar. 2010), 287–304. 104

[77] LUENBERGER, D. G. *Introduction to Linear and Nonlinear Programming*. Addison-Wesley, 1973. 28

[78] LUNT, T. F., AND JAGANNATHAN, R. A prototype real-time intrusion-detection expert system. In *Proceedings of the 1988 IEEE conference on Security and privacy* (1988), SP'88, pp. 59–66. 1, 103

[79] MACQUEEN, J. B. Some methods for classification and analysis of multivariate observations. In *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability* (1967), L. M. L. Cam and J. Neyman, Eds., vol. 1, University of California Press, pp. 281–297. 20, 52

[80] MAHAJAN, M., NIMBHORKAR, P., AND VARADARAJAN, K. The planar k-means problem is np-hard. In *Proceedings of the 3rd International Workshop on Algorithms and Computation* (2009), WALCOM '09, pp. 274–285. 54

[81] MALIN, C. H., CASEY, E., AND AGUILINA, J. M. *Malware Forensics: Investigating and Analyzing Malicious Code*. Syngress, 2008. 81

[82] MALOOF, M. A. *Machine Learning and Data Mining for Computer Security: Methods and Applications (Advanced Information and Knowledge Processing)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005. 3

[83] MANGASARIAN, O. L. Exact 1-norm support vector machines via unconstrained convex differentiable minimization. *Journal of Machine Learning Research 7* (2006), 1517–1530. 47

[84] MANNILA, H., AND TOIVONEN, H. Discovering generalized episodes using minimal occurrences. In *In Proceedings of the 2nd International Conference on Knowledge Discovery in Databases and Data Mining* (1996), AAAI Press, pp. 146–151. 11

[85] MANNILA, H., TOIVONEN, H., AND VERKAMO, A. I. Discovering frequent episodes in sequences (extended abstract). In *In 1st Conference on Knowledge Discovery and Data Mining* (1995), pp. 210–215. 11

[86] MARCEAU, C. Characterizing the behavior of a program using multiple-length n-grams. In *Proceedings of New Security Paradigms Workshop* (2000), pp. 101–110. 90

[87] MCHUGH, J. Testing intrusion detection systems: A critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. In *Proceedings of the ACM Transactions on Information and System Security (TISSEC)* (2000), vol. 3, pp. 262–294. 78, 90

[88] MILLIGAN, G. W., AND ISAAC, P. D. The validation of four ultrametric clustering algorithms. *Pattern Recognition 12*, 2 (1980), 41–50. 52

[89] MIRKIN, B. *Clustering for Data Mining: A Data Recovery Approach (Chapman & Hall/CRC Computer Science & Data Analysis)*, 1 ed. Chapman and Hall/CRC, Apr. 2005. 52

[90] MITCHELL, T. M. *Machine learning*. McGraw Hill series in computer science. McGraw-Hill, 1997. 1

[91] NEUMANN, J., SCHNÖRR, C., AND STEIDL, G. Combined svm-based feature selection and classification. *Machine Learning 61*, 1-3 (Nov. 2005), 129–150. 47

[92] NGUYEN, H. T. An optimal k-means clustering and application to intrudion detection. Tech. rep., Norwegian Information Security Laboratory, Gjøvik University College, Norway, 2012. 7, 31

[93] NGUYEN, H. T., FRANKE, K., AND PETROVIC, S. Optimizing a class of feature selection measures. In *NIPS 2009 Workshop on Discrete Optimization in Machine Learning: Submodularity, Sparsity & Polyhedra (DISCML)* (2009). 5, 31

[94] NGUYEN, H. T., FRANKE, K., AND PETROVIC, S. Towards a generic feature-selection measure for intrusion detection. In *International Conference Pattern Recognition* (2010), pp. 1529–1532. 5, 7, 31, 57, 58, 78

[95] NGUYEN, H. T., FRANKE, K., AND PETROVIC, S. Improving effectiveness of intrusion detection by correlation feature selection. *International Journal of Mobile Computing and Multimedia Communications (IJMCMC) 3*, 1 (2011), 21–34. 7, 57, 58

[96] NGUYEN, H. T., FRANKE, K., AND PETROVIC, S. A new ensemble-feature-selection framework for intrusion detection. In *11th International Conference on Intelligent Systems Design and Applications (ISDA), 2011* (nov. 2011), pp. 213 –218. 6, 7, 31, 57

[97] NGUYEN, H. T., FRANKE, K., AND PETROVIC, S. On general definition of l1-norm support vector machines for feature selection. *International Journal of Machine Learning and Computing 1*, 3 (2011), 279–283. 6, 7, 31, 57

[98] NGUYEN, H. T., FRANKE, K., AND PETROVIC, S. Reliability in a feature-selection process for intrusion detection. In *Reliable Knowledge Discovery*, H. Dai, J. N. K. Liu, and E. Smirnov, Eds. Springer US, 2012, pp. 203–218. 5, 7, 31, 57, 78

[99] NGUYEN, H. T., PETROVIC, S., AND FRANKE, K. A comparison of feature-selection methods for intrusion detection. In *5th International Conference on Mathematical Methods, Models and Architectures for Computer Network Security, MMM-ACNS* (2010), pp. 242–255. 7, 57

[100] NGUYEN, H. T., PETROVIC, S., AND FRANKE, K. A general l1-norm support vector machine for feature selection. In *3rd International Conference on Machine Learning and Computing* (2011), pp. 591–595. 6, 7, 31, 57

[101] NGUYEN, H. T., TORRANO-GIMENEZ, C., ALVAREZ, G., FRANKE, K., AND PETROVIC, S. Enhancing the effectiveness of web application firewalls by generic feature selection. *Logic Journal of IGPL* (2012). 7

[102] NGUYEN, H. T., TORRANO-GIMENEZ, C., ÁLVAREZ, G., PETROVIC, S., AND FRANKE, K. Application of the generic feature selection measure in detection of web attacks. In *4th International Conference on Computational Intelligence in Security for Information Systems* (2011), pp. 25–32. 7, 57

[103] NIU, Y.-S., AND TAO, P. D. A dc programming approach for mixed-integer linear programs. In *Modelling, Computation and Optimization in Information Systems and Management Sciences* (2008), pp. 244–253. xix, 27, 28, 29

[104] NOCEDAL, J., AND WRIGHT, S. J. *Numerical Optimization*. Springer, Aug. 2000. 18, 23

[105] NORTHCUTT, S. *Network Intrusion Detection: An Analyst's Handbook*. New Riders Publishing, Thousand Oaks, CA, USA, 1999. 1

[106] PASTRANA, S., GIMENEZ, C. T., NGUYEN, H. T., AND ORFILA, A. A methodology to simplify the evasion of web application firewalls. Tech. rep., Norwegian Information Security Laboratory, Gjøvik University College, Norway, 2012. 7, 57

[107] PEARL, J. *Heuristics: intelligent search strategies for computer problem solving*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1984. 37

[108] PENG, H., LONG, F., AND DING, C. H. Q. Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence 27*, 8 (2005), 1226–1238. 13, 15, 16, 32, 34, 36, 43, 58, 62

[109] PFAHRINGER, B. Winning the kdd99 classification cup: bagged boosting. *SIGKDD Explor. Newsl. 1* (January 2000), 65–66. 91

[110] PHAM, D., DIMOV, S., AND NGUYEN, C. Selection of K in K -means clustering. *Journal of Mechanical Engineering Science, 219*, 1 (Jan. 2005), 103–119. 52

[111] QUINLAN, J. R. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993. 76, 79, 91

[112] RAISSI, C., BRISSAUD, J., DRAY, G., PONCELET, P., ROCHE, M., AND TEISSEIRE, M. Web analyzing traffic challenge: description and results. In *Proceedings of the ECML/PKDD 2007 Discovery Challenge* (2007), pp. 47–52. 59

[113] RAKOTOMAMONJY, A. Variable selection using svm based criteria. *Journal of Machine Learning Research 3* (Mar. 2003), 1357–1370. 47

[114] RIECK, K. *Machine learning for application-layer intrusion detection*. Ph.D. thesis, Berlin Institute of Technology, 2009. 12

[115] RIECK, K., HOLZ, T., WILLEMS, C., DÜSSEL, P., AND LASKOV, P. Learning and classification of malware behavior. In *Proceedings of the Conference on Detection of Intrusions and Malware & Vulnerability Assessment* (2008), pp. 108–125. 82

[116] ROESCH, M. Snort - lightweight intrusion detection for networks. In *Proceedings of the 13th USENIX conference on System administration* (1999), LISA '99, pp. 229–238. 1

[117] ROESCH, M. Snort: Lightweight intrusion detection for networks. In *LISA* (1999), USENIX, pp. 229–238. 103

[118] RYAN, J., LIN, M.-J., AND MIIKKULAINEN, R. Intrusion detection with neural networks. In *Advances in Neural Information Processing Systems 10* (1998), M. I. Jordan, M. J. Kearns, and S. A. Solla, Eds., Cambridge, MA: MIT Press, pp. 943–949. 58

[119] SABHNANI, M., AND SERPEN, G. Why machine learning algorithms fail in misuse detection on kdd intrusion detection data set. *Intelligent Data Analysis 8*, 4 (Sept. 2004), 403–415. 78, 103

[120] SAEYS, Y., ABEEL, T., AND PEER, Y. Robust feature selection using ensemble feature selection techniques. In *Proceedings of the European conference on Machine Learning and Knowledge Discovery in Databases - Part II* (2008), ECML PKDD '08, pp. 313–325. 43

[121] SAMI, A., YADEGARI, B., RAHIMI, H., PEIRAVIAN, N., HASHEMI, S., AND HAMZEH, A. Malware detection based on mining api calls. In *Proceedings of the 2010 ACM Symposium on Applied Computing* (2010), pp. 1020–1025. 81, 82

[122] SAMUEL, A. L. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3, 210–229. 1

[123] SCHILLER, C. A., BINKLEY, J., HARLEY, D., EVRON, G., BRADLAY, T., WILLEMS, C., AND CROSS, M. *Botnets–The Killer Web App*. Syngress, 2007. 84

[124] SKOUDIS, E., AND LISTON, T. *A Step-by-Step Guide to Computer Attacks and Effective Defenses (2nd Edition)*. Prentice Hall, 2006. 84

[125] SOMMER, R., AND PAXSON, V. Outside the closed world: On using machine learning for network intrusion detection. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy* (2010), SP '10, pp. 305–316. 3

[126] SON, T. A., AN, L. T. H., KHADRAOUI, D., AND TAO, P. D. Solving qos routing problems by dca. In *Intelligent Information and Database Systems* (2010), pp. 460–470. 27, 28

[127] STEINHAUS, H. Sur la division des corp materiels en parties. *Bulletin de l'Acadmie Polonaise des Sciences, 1* (1956), 801–804. 20

[128] STONE-GROSS, B., COVA, M., CAVALLARO, L., GILBERT, B., SZYDLOWSKI, M., KEMMERER, R. A., KRUEGEL, C., AND VIGNA, G. Your botnet is my botnet: analysis of a botnet takeover. In *ACM Conference on Computer and Communications Security* (2009), pp. 635–647. 81

[129] SUNG, A. H., AND MUKKAMALA, S. Identifying important features for intrusion detection using support vector machines and neural networks. In *Proceedings of the 2003 Symposium on Applications and the Internet* (2003), SAINT '03, pp. 209–. xiii, 58, 74, 75

[130] TAO, P. D., CANH, N. N., AND AN, L. T. H. An efficient combined dca and b&b using dc/sdp relaxation for globally solving binary quadratic programs. *Journal of Global Optimization 48*, 4 (2010), 595–632. 27, 28

[131] TIKHONOV, A. N., AND ARSENIN, V. Y. *Solutions of Ill-Posed Problems*. Winston, New York, 1977. 15

[132] TORRANO-GIMENEZ, C., PEREZ-VILLEGAS, A., AND ÁLVAREZ, G. A self-learning anomaly-based web application firewall. In *Computational Intelligence in Security for Information Systems-CISIS* (2009), pp. 85–92. 58

[133] VAPNIK, V. N. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995. 12, 16, 48

[134] VAPNIK, V. N. *Statistical Learning Theory*. Wiley-Interscience, 1998. 47

[135] VIGNA, G., AND KRUEGEL, C. *Handbook of Information Security*. Wiley, December 2005, ch. Host-based Intrusion Detection Systems. 87

[136] WANG, K., PAREKH, J. J., AND STOLFO, S. J. Anagram: A content anomaly detector resistant to mimicry attack. In *In Proceedings of the 9 th International Symposium on Recent Advances in Intrusion Detection (RAID* (2006), pp. 226–248. 12

[137] WANG, K., AND STOLFO, S. J. Anomalous payload-based network intrusion detection. In *Recent Advances in Intrusion Detection* (2004). 12

[138] WANG, L., AND SHEN, X. On l1-norm multiclass support vector machines: Methodology and theory. *Journal of the American Statistical Association 102* (2007), 583–594. 47

[139] WEKA MACHINE LEARNING PROJECT. Weka. URL http://www.cs.waikato.ac.nz/~ml/weka. 71, 79, 91

[140] WESTON, J., MUKHERJEE, S., CHAPELLE, O., PONTIL, M., POGGIO, T., AND VAPNIK, V. Feature selection for svms. In *Advances in Neural Information Processing Systems 13* (2000), MIT Press, pp. 668–674. 13, 47

[141] WILLEMS, C., HOLZ, T., AND FREILING, F. C. Toward automated dynamic malware analysis using cwsandbox. *IEEE Security & Privacy 5*, 2 (2007), 32–39. 82

[142] WOLPERT, D. H. The lack of a priori distinctions between learning algorithms. *Neural Comput. 8*, 7 (Oct. 1996), 1341–1390. 16, 103

[143] WOLPERT, D. H. The supervised learning no-free-lunch theorems. In *In Proc. 6th Online World Conference on Soft Computing in Industrial Applications* (2001), pp. 25–42. 16, 103

[144] WU, S. X., AND BANZHAF, W. Review: The use of computational intelligence in intrusion detection systems: A review. *Applied Soft Computing. 10*, 1 (Jan. 2010), 1–35. 1, 3, 58, 103

[145] XU, R., AND WUNSCH, I. Survey of clustering algorithms. *Neural Networks, IEEE Transactions on 16*, 3 (2005), 645–678. 52

[146] XYDAS, I., MIAOULIS, G., BONNEFOI, P.-F., PLEMENOS, D., AND GHAZANFARPOUR, D. Using an evolutionary neural network for web intrusion detection. In *Proceedings of the 26th IASTED International Conference on Artificial Intelligence and Applications* (2008), AIA '08, pp. 258–265. 58

[147] YE, Y., WANG, D., LI, T., YE, D., AND JIANG, Q. An intelligent pe-malware detection system based on association mining. *Journal in Computer Virology 4*, 4 (2008), 323–334. 81

[148] YE, Y., WANG, D., LI, T., YE, D., AND JIANG, Q. An intelligent pe-malware detection system based on association mining. *Journal in Computer Virology 4*, 4 (2008), 323–334. 82

[149] YUAN, J.-L., AND FINE, T. L. Neural-network design for small training sets of high dimension. *IEEE Transactions on Neural Networks 9*, 2 (1998), 266–280. 47

[150] ZAKI, M. J. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering 12*, 3 (May 2000), 372–390. 11

[151] ZHU, J., ROSSET, S., HASTIE, T., AND TIBSHIRANI, R. 1-norm support vector machines. In *Neural Information Processing Systems* (2003), MIT Press, p. 16. 47

About the Author

Curriculum Vitae

Hai Thanh Nguyen graduated from Lomonosov Moscow State University in applied mathematics and computer science in 2007. His master thesis on mathematical cryptanalysis of hash functions was nominated for the best thesis at the Faculty of Computational Mathematics and Cybernetics, Moscow State University. In October 2008, Nguyen joint the Norwegian Information Security Laboratory as a research fellow. He is doing research on machine learning for information security and digital forencics with the supervision of Prof. Franke and Prof. Petrović. Since then, he has published six international scientific journal articles, two book chapters and eleven peer-reviewed conference papers. From April-2012, he has been a visitting scholar in the machine learning group at University of California at Santa Cruz and working with Prof. Warmuth on online-learning for big data analysis and applications to information security and digital forensics. Nguyen obtained the Vietnamese Overseas Scholarship Program for studying abroad in Russia from 2001 to 2007, the Best Presentation Award at the PhD workshop organized by Norwegian Research School of Technology in 2011 and recently the Grant from the Leiv Eiriksson Mobility Programme 2012 of the Research Council of Norway for collaboration with University of California at Santa Cruz.

List of Publications

Journal Articles

1. **Nguyen, H.T.**, Torrano-Gimenez, C., Alvarez, G., Franke, K., Petrovic, S. (2012). "Enhancing the Effectiveness of Web Application Firewalls by Generic Feature Selection". In *Proceeding of the special issue in the Oxford Logic Journal of the IGLP*, Online ISSN 1368-9894 - Print ISSN 1367-0751, doi:10.1093/jigpal/jzs033.

2. Torrano-Gimenez, C., **Nguyen, H.T.**, Franke, K.. Alvarez, G. (2012). "Combining Expert Knowledge with Automatic Feature Extraction for Reliable Web Attack Detection". In *Proceeding of the Security and Communication Networks Journal*, John Wiley & Sons, Ltd-DOI: 10.1002/sec.603.

3. Hartung, D., Olsen, M. A., Xu, H., **Nguyen, H.T.**, and Busch, C. (2012). Comprehensive analysis of spectral minutiae for vein pattern recognition. In *Proceeding of the IET Biometrics Journal*, Vol. 1, Issue 1, pp 25-36.

4. Pastrana, S., Gimenez, C. T., **Nguyen, H. T.**, and Orfila, A. (2012). A Methodology to Simplify the Evasion of Web Application Firewalls. *Computers & Security Journal*, Elsevier. (submitted).

5. **Nguyen, H.T.**, Franke, K., Petrovic, S. (2011). "On General Definition of L1-norm Support-Vector Machine for Feature Selection". In *Proceedings of the International Journal of Machine Learning and Computing*, IJMLC 2011 Vol.1(3): 279-283 ISSN: 2010-3700.

6. **Nguyen, H.T.**, Franke, K., Petrovic, S. (2011). "Improving Effectiveness of Intrusion Detection by Correlation-Feature Selection". In *International Journal of Mobile Computing and Multimedia Communication (IJMCMC)* 3(1), 21-34.

Chapters in Books

1. **Nguyen, H.T.**, Franke, K., Petrovic, S. (2012). "Reliability in A Feature-Selection Process for Intrusion Detection". In (Eds.) Honghua Dai, James Liu and Evgueni Smirnov: "*Reliable Knowledge Discovery*", Springer 2012, ISBN 978-1-4614-1902-0. p. 203-218.

2. **Nguyen, H.T.**, Franke, K., Petrovic, S. (2012). "Feature-Extraction Methods for Intrusion Detection". In (Eds.) M. Gupta, J. Walp and R. Sharman, "*Threats, Countermeasures and Advances in Applied Information Security*". IGI Global, pages 23-52, DOI: 10.4018/978-1-4666-0978-5.ch002, ISBN13: 9781466609785.

Peer-reviewed Conferences Papers

1. **Nguyen, H.T.**, and Franke, K. (2012). "Adaptive Intrusion Detection System via Online Machine Learning". In *12th International Conference on Hybrid Intelligent Systems (HIS 2012)*, Pune, India. (accepted paper)

2. Berg, P., Franke, K., and **Nguyen, H.T.** (2012). "Feature Selection for Botnet Malware Detection". In *12th International Conference on Intelligent Systems Design and Applications (ISDA 2012)*, Kochi, Indian. (accepted paper)

3. **Nguyen, H.T.**, and Franke, K. (2012). " A General Lp-norm Support Vector Machine via Mixed 0-1 Programming Problem". In *8th International Conference on Machine Learning and Data Mining*, Berlin, pp. 40-49.

4. **Nguyen, H.T.**, Franke, K., Petrovic, S. (2011). "A New Ensemble-Feature-Selection Framework for Intrusion Detection". In *11th International Conference on Intelligent Systems Design and Applications (ISDA)*, November 22-24, 2011 - Crdoba, Spain, pp. 213-218.

5. **Nguyen, H.T.**, Torrano-Gimenez, C., Alvarez, G., Franke, K., Petrovic, S. (2011) Application of the Generic Feature Selection Measure in Detection of Web Attacks. In *Proc. of International Conference on Computational Intelligence in Security for Information Systems (CISIS)*, Lecture Notes in Computer Science, 2011, Volume 6694/2011, 25-32.

6. **Nguyen, H.T.**, Franke, K., Petrovic, S. (2011). "A General L1-norm Support-Vector Machine for Feature Selection". In *Proceedings of the International Conference on Machine Learning and Computing (ICMLC 2011)*, s.V1-591-V1-595, IEEE Press, Singapore, 26-28 February.

7. Torrano-Gimenez, C., **Nguyen, H.T.**, Alvarez, G., Franke, K., Petrovic, S. (2011), "Applying feature selection to payload-based web application firewalls", *In 3rd International Workshop on Security and Communication Networks (IWSCN)*, May 2011.

8. **Nguyen, H.T.**, Petrovic, S., Franke, K.(2010). "A Comparison of Feature-Selection Methods for Intrusion Detection". In *Proceedings of Fifth International Conference on Mathematical Methods, Models, and Architectures for Computer Networks Security (MMM-ACNS)*, St.Petersburg, Russia, September 8-11, 2010, pp.242-255.

9. **Nguyen, H.T.**, Franke, K., Petrovic, S. (2010). "Towards a Generic Feature-Selection Measure for Intrusion Detection", In *Proc. International Conference on Pattern Recognition (ICPR 2010)*, Istanbul, Turkey, pp 1529-1532.

10. **Nguyen, H.T.**, Franke, K., Petrovic, S. (2010). "Improving Effectiveness of Intrusion Detection by Correlation Feature Selection". In *Proceedings of the International Conference on Availability, Reliability and Security (ARES)*, Krakow, Poland, pp. 17-24.

11. **Nguyen, H.T.**, Franke,K., Petrovic, S. (2009). "Optimizing a class of feature selection measures", In *Proceedings of the NIPS 2009 Workshop on Discrete Optimization in Machine Learning: Submodularity, Sparsity & Polyhedra (DISCML)*, Vancouver, Canada.

12. Karpunin, G.A., **Nguyen, H.T.** (2008). "On Optimal Selection of XOR Function in A Differential Cryptanalysis Model of the MDx Hash Functions". In *Proceedings of 7th Russian Conference on Mathematics and Information Security*, October 30-31, Moscow (2008), Part 2, pp.65-70. (in russian).

13. **Nguyen, H.T.**, Karpunin, G.A (2007). "On Comparising XOR with MAJ from the View of Cryptography". *Proceedings of the IX International Workshop on Discrete Mathematics and Its Applications* dedicated to 75-anniversary from the birthday of Academician O. B. Lupanov, 18-22/June/2007, Faculty of Mechanics and Mathematics, Lomonosov Moscow State University, Russia.

14. **Nguyen, H.T.** (2007). "Cryptanalysis of Hash Functions of MDx Family". *Proceedings of the XIV International Scientific Conference for Undergraduate, Postgraduate Students and Young Scientists "LOMONOSOV"*, April/2007, Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University, Russia.

Talks & Posters

1. **Nguyen, H.T.**, Franke, K. (2012). "Feature Selection for Information Security and Digital Forensics Problems: Methods and Challenges" in *the Workshop on Machine Learning with Real-time and Streaming Applications*, May 7-11 2012, University of California, Berkeley. (poster presentation)

2. **Nguyen, H.T.**, Franke, K. (2012). "An Unified Framework of Machine Learning via Mixed 0-1 Linear Programming" in *the Workshop on Mixed Integer Programming*, July 16-19 2012, University of California, Davis. (poster presentation)

3. **Nguyen, H.T.**, Franke, K., Petrovic, S. (2011). "Feature Selection via Mixed 0-1 Programming Problem". *The 18th Machine Learning Summer School is co-organized by INRIA and PASCAL*, Bordeaux, France, 4-17 September, 2011. (poster paper).

4. Berg, P.E., Franke, K., and **Nguyen, H.T.** (2011). Feature Selection for Botnet Malware Detection. In *8th Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA 2011)*, July 7-8th, 2011, Amsterdam, The Netherlands. (poster paper).