BAŞKENT UNIVERSITY
INSTITUTE OF SCIENCE AND ENGINEERING

OPTIMIZED WEIGHTED ENSEMBLE CLASSIFIER FOR INTRUSION DETECTION APPLICATION

ATİLLA ÖZGÜR

PhD THESIS
2017

OPTIMIZED WEIGHTED ENSEMBLE CLASSIFIER FOR INTRUSION DETECTION APPLICATION

ENİYİLENMİŞ AĞIRLIKLI SINIFLANDIRICI TOPLULUĞU İLE SALDIRI TESPİT UYGULAMASI

Atilla ÖZGÜR

Thesis Submitted
in Partial Fulfillment of the Requirements
For the Degree of Doctor of Philosophy
in Department of Electrical and Electronics Engineering
at Başkent University

2017

This thesis, titled: ”OPTIMIZED WEIGHTED ENSEMBLE CLASSIFIER FOR INTRUSION DETECTION APPLICATION”, has been approved in partial fulfillment of the requirements for the degree of DOCTOR OF PHILOSOPHY IN DEPARTMENT ELECTRICAL AND ELECTRONICS ENGINEERING, by our jury, on 11/05/2017

Chairman:	
	Associated Professor Hasan Şakir BİLGE
Member:	
	Associated Professor Mustafa DOĞAN
Member (Supervisor):	
	Associated Professor Hamit ERDEM
Member:	
	Assistant Professor Fatih NAR
Member:	
	Assistant Professor Mustafa SERT

APPROVAL

../../2017

Prof. Dr. Emin AKATA

Institute of Science and Engineering

BAŞKENT ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ
YÜKSEK LİSANS / DOKTORA TEZ ÇALIŞMASI ORİJİNALLİK RAPORU

Tarih:30 /05/ 2017

Öğrencinin Adı, Soyadı : Atilla Özgür

Öğrencinin Numarası : 20820015

Anabilim Dalı : Elektrik Elektronik Mühendisliği

Programı : Doktora

Danışmanın Unvanı/Adı, Soyadı : Doç. Dr. Hamit Erdem

Tez Başlığı : Eniyilenmiş Ağırlıklı Sınıflandırıcı Topluluğu ile Saldırı Tespit Uygulaması

Yukarıda başlığı belirtilen Doktora tez çalışmamın; Giriş, Ana Bölümler ve Sonuç Bölümünden oluşan, toplam 94 sayfalık kısmına ilişkin, 30 / 05/ 2017 tarihinde şahsım tarafından turnitin adlı intihal tespit programından aşağıda belirtilen filtrelemeler uygulanarak alınmış olan orijinallik raporuna göre, tezimin benzerlik oranı %4'tür.

Uygulanan filtrelemeler:

1. Kaynakça hariç
2. Alıntılar hariç
3. Beş (5) kelimeden daha az örtüşme içeren metin kısımları hariç
4. Tez çalışmasından çıkarılmış, yazarları (Atilla Özgür ve Hamit Erdem) olan preprint (https://peerj.com/preprints/1954/) hariç

"Başkent Üniversitesi Enstitüleri Tez Çalışması Orijinallik Raporu Alınması ve Kullanılması Usul ve Esaslarını" inceledim ve bu uygulama esaslarında belirtilen azami benzerlik oranlarına tez çalışmamın herhangi bir intihal içermediğini; aksinin tespit edileceği muhtemel durumda doğabilecek her türlü hukuki sorumluluğu kabul ettiğimi ve yukarıda vermiş olduğum bilgilerin doğru olduğunu beyan ederim.

Öğrenci İmzası:………………….

Onay

30 /05 / 2017

Öğrenci Danışmanı Doç. Dr. Hamit Erdem

ACKNOWLEDGMENTS

I would like to dedicate this thesis to my late grandmother Reyhane Dereli and my grandfather İbrahim Dereli. They taught me nothing is impossible if you put enough effort.

My heartfelt appreciation goes to my family: My wife Funda and my daughter Duru.

I would like to express my deepest gratitude to my thesis advisor Hamit Erdem for his guidance, advice, criticism, encouragement, and insight throughout the thesis.

Atilla ÖZGÜR

ABSTRACT

OPTIMIZED WEIGHTED ENSEMBLE CLASSIFIER FOR INTRUSION DETECTION APPLICATION

Atilla ÖZGÜR

Başkent University

The Department of Electrical and Electronics Engineering

Computer and communication systems become the foundations of modern life. With the advances in the Internet, usage of these systems increases and intrusions against these systems increases too. Therefore, finding and preventing intrusions against these systems becomes more and more important. To protect these systems, Intrusion Detection Systems (IDS) are implemented. In recent years, machine learning and optimization techniques are increasingly used in IDS. New methods are implemented using KDD99 and its derivative NSL-KDD datasets based on intelligent IDS systems in this thesis study. First, a detailed review is made on studies that uses above mentioned datasets, and according to this review, detailed statistics are derived on usage of these datasets. Next, two different methods are proposed for IDS. These methods are based on principles of classifier ensemble and hybrid IDS. In the first method, genetic algorithms (GA) are used for feature selection (an important part for classification) and ensemble weight finding. The proposed method is named as Genetic Algorithms based Feature Selection and Weights Finding (GA-FS-WF). In the second method, hybrid ensemble classifier subject re-visited again. In this method, convex optimization techniques are used for finding weights for ensemble classifiers. Proposed method models weights finding in ensemble as a mathematical objective function and solves it as an optimization problem. In both proposed methods, full dataset NSL-KDD is used. Success of proposed methods are measured with classifier performance metrics and compared with similar methods in the literature.

KEYWORDS: Feature Selection, Classifier Fusion,Genetic Algorithms, Intrusion Detection Systems, Machine Learning, Convex Optimization, Convex Relaxation.

Advisor: Assoc Prof. Hamit ERDEM, Başkent University, Department of Electrical and Electronics Engineering.

ÖZ

ENİYİLENMİŞ AĞIRLIKLI SINIFLANDIRICI TOPLULUĞU İLE SALDIRI TESPİT UYGULAMASI

Atilla ÖZGÜR

Başkent Üniversitesi

Elektrik Elektronik Mühendisliği Anabilim Dalı

Bilgisayar ve iletişim sistemleri modern hayatın temellerini oluşturmaktadır. Internet ağının gelişmesiyle birlikte bu sistemlerin kullanımında büyük artışlar olmakta, ancak bu sistemlere yönelik saldırılar da aynı oranda artmaktadır. Bu yüzden, söz konusu sistemlerin saldırılara karşı korunması ve gelen saldırıların tespiti giderek önem kazanmış ve bu amaçla Saldırı Tespit Sistemleri (STS) geliştirilmiştir. Son yıllarda STS'lerde makine öğrenmesi ve eniyileme tekniklerinin kullanımı giderek artmaktadır. Bu tez çalışmasında özgün STS yöntemleri önerilmiş ve önerilen bu yöntemler KDD99 ve türevi NSL-KDD veri setleri kullanılarak doğrulanmıştır. STS çalışmaları üzerinde yapılan detaylı literatür taraması sonucuna göre, bu veri setlerinin makine öğrenmesi alanında kullanımına yönelik detaylı istatistikler çıkarılmıştır. Çalışmanın devamında, STS için iki farklı sınıflandırıcı füzyon yöntemi geliştirilmiştir. Genetik Algoritma tabanlı Nitelik Seçme ve Ağırlık Bulma (GA-NS-AB) olarak adlandırılan ilk yöntemde, sınıflandırıcı çalışmaları için önemli bir aşama olan nitelik çıkarma ve sınıflandırıcı füzyonu ağırlık bulma işlemleri, Genetik Algoritmalar (GA) kullanılarak gerçekleştirilmiştir. İkinci yöntemde ise sınıflandırıcı ağırlıklarını bulma problemi, eniyileme problemi olarak modellenmiş ve yeni bir maliyet fonksiyonu tanımlanmıştır. Bu fonksiyonun çözümü için dışbükey gevşetme ve dışbükey eniyileme yöntemleri kullanılmıştır. Geliştirilen her iki yöntemde de NSL-KDD veri setinin tamamı kullanılırken, yöntemlerin başarısı benzer yöntemlerle karşılaştırılmıştır.

ANAHTAR SÖZCÜKLER: Nitelik Seçme, Sınıflandırıcı Füzyonu, Genetik Algoritma, Saldırı Tespit Sistemleri, Makine Öğrenmesi, Dışbükey Eniyileme, Dışbükey Gevşeme.

Danışman: Doç. Dr. Hamit ERDEM, Başkent Üniversitesi, Elektrik Elektronik Mühendisliği Bölümü.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS and ABBREVIATIONS

ANOVA	Analysis of variance
CPU	Central Processing Unit
CRF	Conditional Random Field
DARPA	Defense Advanced Research Projects Agency
DOS	Denial of Service
ELM	Extreme Learning Machines
FN	False Negative
FP	False Positive
FPGA	Field-programmable gate array
GA-FS-WF	Genetic Algorithms based Feature Selection and Weights Finding
GA	Genetic Algorithms
IDS	Intrusion Detection Systems
IEEE	Institute of Electrical and Electronics Engineers
ISCX	An IDS dataset.
KDD99	An IDS dataset derived from DARPA dataset.
KDD	Knowledge Discovery in Databases
KNN	K-nearest Neighbor
ML	Machine Learning
MLP	Multi Layer Perceptron
MLR	Machine Learning Research
MOA	Massive Only Analysis, a data stream mining library
NI	No Information
NP	Non-deterministic Polynomial-time
NSL-KDD	An IDS dataset derived from KDD99 dataset.
QFWEC	Weighted Classifier Ensemble based on Quadratic Form
ROC	Receiver operating characteristic
SCI	Science Citation Index
SDWEC	Sparsity Driven Weighted Ensemble Classifier
SMO	Sequential Minimal Optimization
SVM	Support Vector Machines
TCP	Transmission Control Protocol
TN	True Negative
TP	True Positive
UCI	University of California Irvine
WMV	Weighted Majority Voting

1. INTRODUCTION

1.1. History of cyber attacks

Theoretical background of computer viruses was introduced by Neumann in his seminal paper [1], ”Theory of self-reproducing automata” in 1966. First practical viruses are written in between 1970 to 1980. First viruses were mostly harmless programs, written on a whim of programmers. Their developers were trying to investigate the conceptual limits of computers. All of them were easily cleaned [2]. First known computer virus is written by Bob Thomas. This virus copied itself between nodes of Arpanet– forerunner of Internet. When this program was written, the computer virus concept did not exist; nevertheless, it is generally accepted as the first computer virus [3].

Between 1980 and 1990, first wave of computer viruses started. In 1983, Cohen wrote the first article about computer viruses, giving both source code and theoretical background [4]. Cohen performed a number of experiments in different operating systems, and proved that virus concept is independent of programming language and operating system. Given suitable conditions, a virus can propagate very easily. Although first computer viruses and worms were written for mainframe and Unix systems, they found their fertile grounds in MS-DOS and windows systems. Most of these viruses were boot-sector viruses [3].

After 1990 to 2000, second wave of computer viruses started. Although *viruses* was dominant in first wave, with the advent of Internet and email, *worms* become more dominant [3].

After 2000 to today, attacks of viruses and worms become a common phenomena, and malware infestations become very common. All types of malware use vulnerabilities of applications and operating systems. Since writing programs and applications become more and more complex, vulnerabilities in programs will not decline. Recent numbers from Common Vulnerabilities and Exposures (CVE) in Figure 1.1 support this.

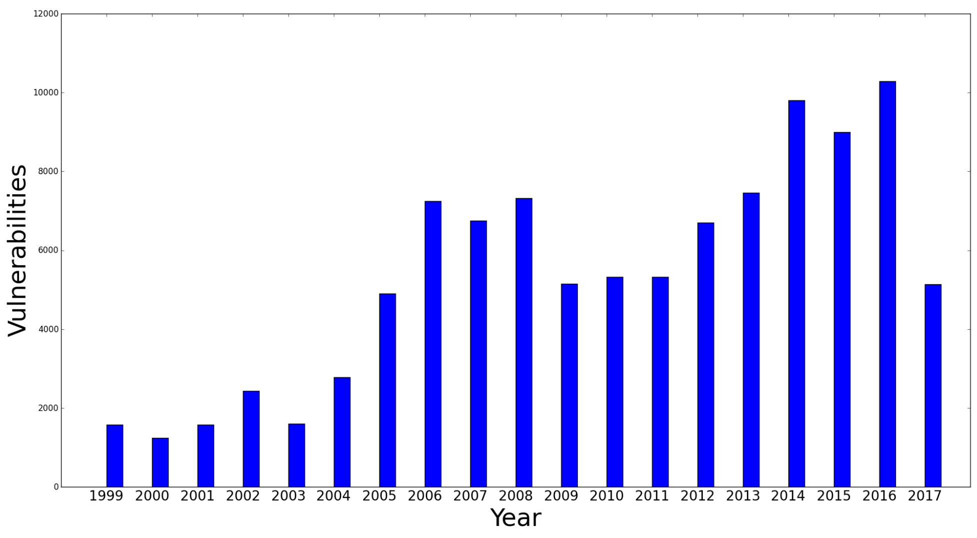


Figure 1.1 CVE common vulnerabilities and exposures by year [5]. (Year 2017 values includes first 4 months only.)

1.2. Why cyber attacks increase

The increase in number of attacks and number of incidents are due to two reasons. The first reason is that Technology *favor* attackers. Anderson showed that an attacker needs to find only one bug to gain a more favorable position.

Even a very moderately resourced attacker can break anything that's at all large and complex. There is nothing that can be done to stop this, so long as there are enough different security vulnerabilities to do statistics. Anderson[6]

The second reason is that hacking and cracking became *profitable businesses*. That is most computer crimes are *professionalized*. There are web sites [7] that offer Distributed Denial of Services (DDOS) attacks. In some forums, you can buy DDOS attacks as cheap as $5 per day [8] compared to $100 per day in 2009 [9].

1.3. Intrusion detection systems

As can be seen from above discussion, cyber attacks will not decline in foreseeable future therefore security tools will gain more and more importance. Different security tools can be seen in Table 1.1.

Table 1.1 Survey of security tools used [10]

Tool Used	Percentage in Participants
Anti-virus software	86.0%
Firewalls	80.7%
Data encryption (data in transit)	64.0%
Audit logs of each access records	60.0%
Data encryption (data at rest)	58.7%
Patch and vulnerability management	57.3%
Intrusion detection systems (IDS)	54.0%
Network monitoring tools	52.7%
Mobile device management (MDM)	52.0%
User access controls	50.7%
Intrusion detection and prevention systems	48.0%
Access control lists	47.3%
Single sign on	47.3%
Web security gateway	41.3%
Multi-factor authentication	39.3%
Messaging security gateway	37.3%
Data loss prevention (DLP application)	36.0%

Intrusion Detection Systems (IDS) are increasingly used in enterprises. Most common IDS structure in literature is given in Figure 1.2. According to this structure, most used datasets in the IDS research is KDD99 and its derivative NSL-KDD.

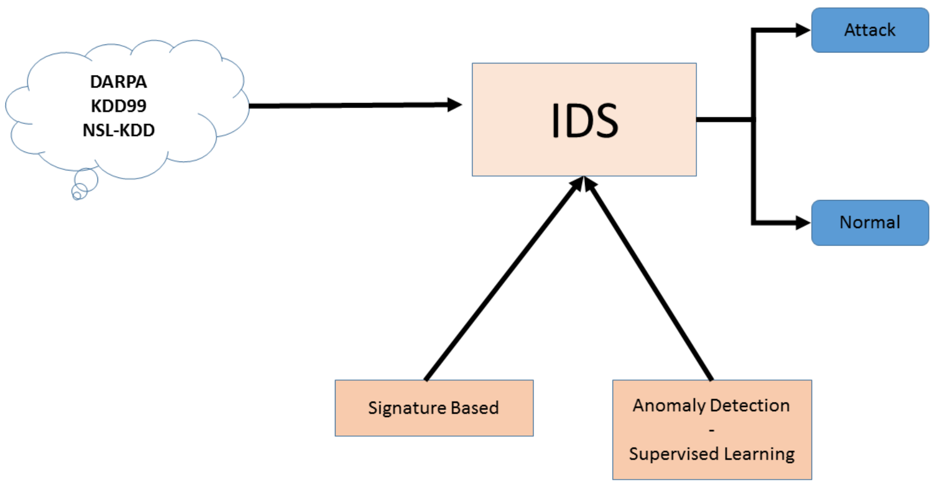


Figure 1.2 Most common IDS structure in literature

1.4. Contributions of this thesis

As mention before, the most used IDS dataset in the literature is KDD99 and its derivative NSL-KDD. Usage of KDD99 can be seen in Figure 1.3 and usage of

NSL-KDD can be seen in Figure 1.4. We review these datasets and derive detailed statistics in their usage in intrusion detection and machine learning in Chapter 2. The following descriptive statistics about the reviewed studies are given in Chapter 2: main contribution of these articles, applied algorithms, compared classification algorithms, software toolbox usage, the size and type of the used dataset for training and testing, and classification output classes (binary, multi-class).

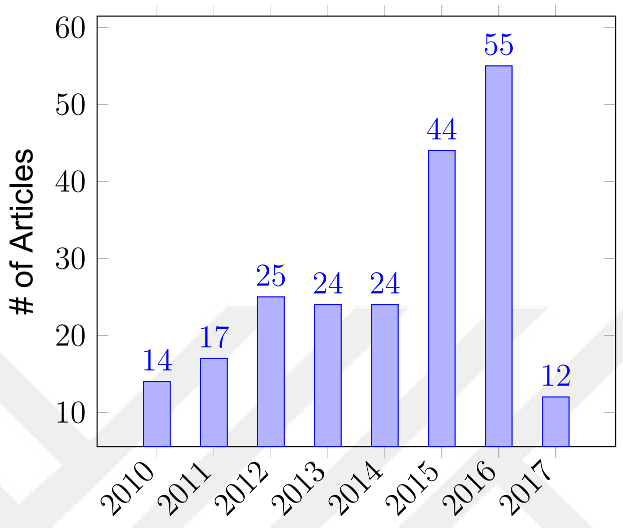


Figure 1.3 **KDD99 dataset usage by years.** (Year 2017 values includes first 4 months only.)

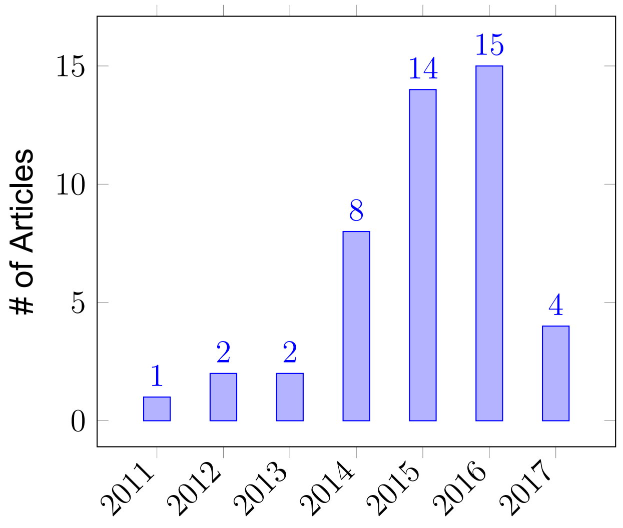


Figure 1.4 **NSL-KDD dataset usage by years.** (Year 2017 values includes first 4 months only.)

The most used methods in IDS articles are hybrid methods, see Table 2.3. In the following chapters, we propose two different hybrid methods for IDS. In the chapter 3, we propose a method that combines classifier ensemble and feature selection using genetic algorithms named ”GA-FS-CE: Genetic Algorithms based Feature Selection and Classifier Ensemble”. In the chapter 4, another weighted classifier ensemble that uses convex optimization methods for finding weights of classifier ensemble are proposed. Problem of weights finding is formulated using objective function that consists of data term, sparsity term and non-negativity constraint on the weights. Both of these methods are tested on NSL-KDD dataset.

2. RELATED WORKS

Although KDD99 dataset is more than 17 years old, it is still widely used in Machine Learning Research (MLR) and Intrusion Detection Systems (IDS). This study reviews 215 research articles from 90 journals indexed in Science Citation Index Expanded between 2010 and 2016. If papers presented in other indexes and conferences are included, number of studies would be much higher. The number of published studies shows that KDD99 is the most used dataset in IDS and machine learning areas, and it is the de facto dataset for these research areas. To show recent usage of KDD99 and the related sub-dataset (NSL-KDD) in IDS and MLR, the following descriptive statistics about the reviewed studies are given: main contribution of articles, the applied algorithms, compared classification algorithms, software toolbox usage, the size and type of the used dataset for training and testing, and classification output classes (binary, multi-class). All of the results are easily reproducible since they are created from provided bibliography file using python scripts. These python scripts and bibliography file are published in an open source repository. In addition to these statistics, a checklist for future researchers has been provided.

2.1. Chapter introduction

Internet, mobile, e-commerce, communication, and information systems have become essential parts of our daily life. Wide usage of these systems makes communication easier, increases data transfer and information sharing, and improves life quality. Although these systems are used in many fields, they suffer from the various attacks such as viruses, worms, or Trojan horses. Due to importance of these systems, these attacks must be identified and stopped as soon as possible. Research about finding attacks and removing their effects have been defined as Intrusion Detection Systems (IDS) [11].

IDS studies can be considered as classification tasks that separate normal behavior of networks from attacks. After the first paper about IDS [12], thousands of studies have been published in this domain. Machine learning and data mining algorithms are widely used in IDS. Most of the machine learning and data mining algorithms are based on the assumption that problem space does not change very fast. But in IDS domain, attackers continuously change and improve their capabilities [13]. Due to this reason, even though machine learning and data mining algorithms are very successful in other domains, their performance degrades in IDS. Thus, IDS is an unsolved problem since this domain is evolving continuously [13].

Similar to other classification and clustering problems, IDS algorithms need training dataset. Although standard datasets are available for other fields, there is no up-to-date standard dataset for IDS. Lack of an up-to-date standard dataset for IDS research has been mentioned by numerous studies [14, 13, 15]. Recent reviews [16, 17, 18, 19] also identify this problem as a research gap. Nonetheless, KDD99 is the most used dataset in IDS domain [20, 21, 22]. According to survey paper of [19], more than 50% of reviewed papers used DARPA/KDD99 in their studies.

Created in 1999 [23], KDD99 has been widely used in many studies for the past 17 years and cited in many studies —Reference article for KDD99 preparation [23] has been cited 943 times according to Google Scholar (May 2017). Moreover, 215 research articles that used KDD99 were published in Science Citation Index Expanded journals from 2010 to present, Table 2.11. Interestingly, KDD99 usage seems to increase in recent years, most probably due to increase in popularity of IDS, Figure 2.1.

According to the results, KDD99 dataset is primarily used in IDS and machine learning research. Additionally, this dataset also has been used for other purposes, such as feature selection and data streams. Based on the 215 published studies, 171 of them has been applied to IDS and 44 of them has not been applied to IDS. From these 215 published studies, 197 of them are machine learning studies. Of these 197 studies, 161 of them are both machine learning and IDS, while 36 are machine learning studies that do not claim to work on IDS. These numbers shows that KDD99 dataset is the main intersection of machine learning research, IDS, and information security.

Although KDD99 has been used in many IDS and machine learning studies, no review study exists that evaluate and analyze the published research and answer the following questions:

- Which machine learning algorithms and IDS methods are used mostly?
- What is the training and testing dataset usage in the published studies?
- What are the sizes of training and testing dataset in proposed studies?
- How many classes have been considered in IDS classification?
- Which performance metrics have been used to measure the results of the classification?

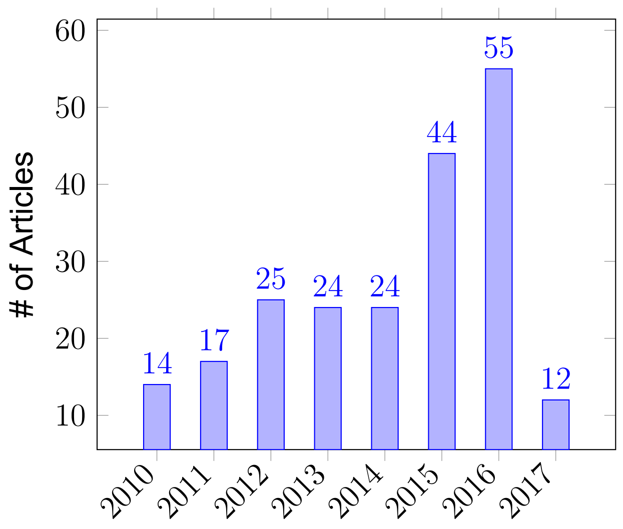


Figure 2.1 **KDD99 dataset usage by years.** (Year 2017 values includes first 4 months only.)

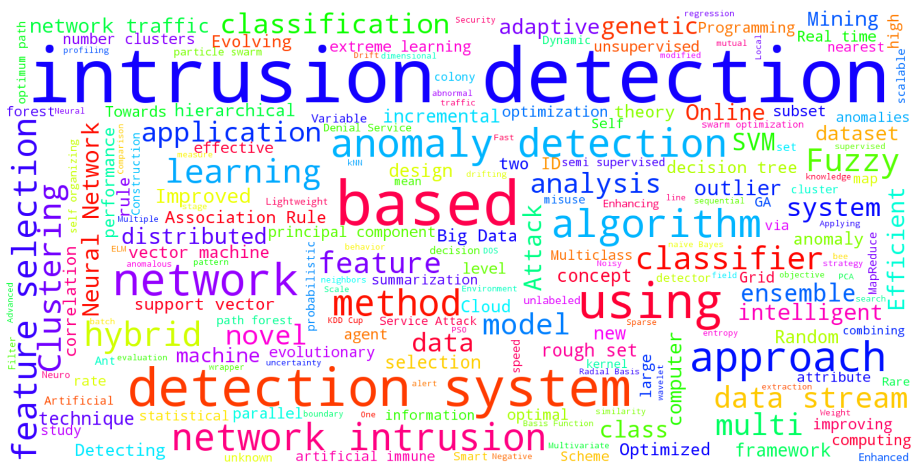


Figure 2.2 Word cloud generated from titles of reviewed 215 research articles.

- Which software tools have been used for implementation and comparison?

To answer these questions, the proposed study reviews 215 studies from 2010 to present focusing on KDD99 usage statistics. We think that the results of the proposed study will be useful for the other researchers who plan to use this dataset in IDS or machine learning studies. All of the results are derived from a bibliography file using python scripts that are published in open source github repository (`https://github.com/ati-ozgur/KDD99ReviewArticle`).

This review differs from the previous review articles considering following aspects: First, most of the reviews in this domain try to include critical papers and explain major approaches. In contrast, our study tries to be comprehensive. Second, only articles indexed in Science Citation Index are included, Table 2.11. No conference articles or articles from other indexes are included in this study. If a journal is removed from Science Citation Index in a year, only articles that are indexed in Science Citation Index are included in this study. To show this aspect, provided bibliography file contains Web of Science number (Accession Number or WOS-number) for all of the articles. We believe that our study includes most of the SCI-indexed studies that used KDD99. Third, comprehensive descriptive statistics about KDD99, machine learning and IDS are given. Some of these statistics are as follows:

1. KDD99 has been analyzed considering number of output classes, training and testing datasets in reviewed studies, Table 2.4 and Table 2.5.
2. Main contribution that concerns on the applied method using KDD99. The applied methods may be clustering, classification, feature selection/reduction algorithms. All the applied methods in the focused period has been evaluated and presented in Table 2.3.
3. The usage frequency of machine learning and IDS algorithms has been presented in Table 2.6 and discussed in detail.
4. Proposed algorithms are implemented and compared with standard algorithms using variety of software packages (Table 2.8).
5. Training and testing dataset sizes and classification types (binary, multi-class).
6. Most of the reviewed articles compare their proposed method with other classifiers. These classifiers have been shown in Table 2.7 and discussed.
7. Although, KDD99 and derived sub-set NSL-KDD dataset have been used in recent studies, some studies compared their results with other datasets. These other datasets and their usage statistics are given in Table 2.9
8. Categorizing the main theme of the published article in three main groups as Machine Learning, Anomaly Detection or Alert Correlation has been presented in Figure 2.8.

Fourth, considering collected statistics, strengths and weaknesses of reviewed articles, a checklist is provided. Fifth, comprehensive bibliography file about reviewed articles are provided. This file can also be used other purposes. For example: (a) Figure 2.2 word cloud from titles are generated from this bibliography file, (b) which datasets are used in data stream studies in addition to KDD99, (c) which algorithms are commonly used in artificial immune systems in KDD99. Finally, the findings of this study would be useful for researchers who may want to use KDD99 or a similar big dataset in their research since KDD99 is one of the biggest datasets in University of California Irvine (UCI) machine learning repository.

The remainder of the chapter is organized as follows: Section 2.2 considers similar related reviews. Section 2.4 gives definitions and history of DARPA, KDD99 and NSL-KDD datasets. Section 2.5 gives a general machine learning model that most reviewed articles use with KDD99 and evaluates contribution of reviewed articles considering the structure of the presented model. Section 2.6 gives descriptive statistics about general KDD99 usage along with figures and tables. Section 2.7 suggests a checklist considering common mistakes and strengths of the reviewed articles for further studies to improve the quality of similar studies. Finally, section 2.8 discusses the results of this chapter.

2.2. Related reviews about KDD99 and IDS

Most of the IDS reviews try to find prominent papers about the subject and summarize them. This approach provides fast learning opportunity for the reader. In contrast to previous review studies, this study follows a different approach. This study provides descriptive statistics that would be useful to researchers who may want to use KDD99 in their research.

One of the most similar review to ours has been presented by Tsai et al.. Their study evaluated 55 articles between 2000 and 2007 that focused on intersection of IDS and machine learning. First, they give definition of the most used single classifiers in machine learning for IDS containing k-nearest neighbor, support vector machines, artificial neural networks, self-organizing maps, decision trees, naive bayes, genetic algorithms, fuzzy logic, hybrid classifiers, and ensemble classifiers. Second, they provide yearly statistics for these categories. Third, they investigate the used dataset in the proposed period. According the study, KDD99 has been used nearly 60% of the published studies. To expand the published review that included 55 articles, our study reviews 215 articles (Section 2.6), and includes more statistics.

Kolias et al. [21] reviewed usage of swarm intelligence techniques in IDS. From these methods, ant colony optimization, ant colony clustering and particle swarm optimization have been compared in their review. Only descriptive statistic included in their study was performance comparison of swarm intelligence techniques in IDS.

Liao et al. [25] categorized IDS systems according to detection approaches (statistics, pattern, rule,state and heuristic based), detection methodology (anomaly detection, signature based and stateful protocol analysis.) They also gave information about whether IDS are time series based and used data source (network or host based). Their taxonomy also included other characteristics of IDS in their study. Instead of giving summary statistics about IDS characteristics, they summarized all information about 30 IDS articles in a table. According to Liao et al., swarm intelligence, immune systems, data mining, genetic algorithms and other machine learning techniques were widely used. Different from Liao et al., we give precise statistics about used algorithms in IDS studies.

Ganapathy et al. [20] reviewed intersection of feature selection and intelligent algorithms in Intrusion Detection. For feature selection, gradually feature removal method, modified mutual information-based feature selection algorithm, CRF-based feature selection, and wrapper based genetic feature selection methods have been compared. Regarding to classification techniques, Ganapathy et al. compared neural networks, genetic algorithms, fuzzy sets, rough sets, Neuro-Fuzzy, fuzzy-genetic algorithms and particle swarm optimization. They did not give any statistics about reviewed studies.

Modi et al. [26] reviewed cloud based IDS systems. According to their survey, intelligent techniques (artificial neural networks, fuzzy logic, association rules, support vector machines, genetic algorithms, hybrid techniques) were widely used in cloud based IDS. Our study shows similar results as can be seen in Table 2.6.

Yang et al. [27] had surveyed artificial immune systems in IDS. They reviewed the concepts antibody/antigen encoding, generation algorithm, evolution algorithm but did not provide any statistics about the reviewed articles.

Elsayed et al. [28] had surveyed evolutionary and swarm intelligence algorithms in network intrusion detection using DARPA and KDD99. They investigated usage of genetic algorithms, genetic programming, ant colony optimization and swarm optimization for different stages of IDS. They presented a few descriptive statistics for evaluating the reviewed articles. First statistics is commonly used fitness functions, second statistics is articles' dataset usage, third statistics is the applied algorithm, and the last statistics is detection rate of the applied algorithm.

Ahmed et al. [19] have surveyed network anomaly detection techniques. They categorize techniques as classification, statistical based, information theory, and clustering. Most of the reviewed papers, more than 50%, uses DARPA/KDD99. Lack of recent dataset problem and problems of DARPA/KDD99 are given special attention. Descriptions for other available datasets for network anomaly detection domain are provided.

Folino and Sabatino [29] have reviewed ensemble based techniques in intrusion detection. They provided following metrics: datasets used, main algorithms used, ensemble method, and performance metrics. Most of the reviewed articles use KDD99 in their studies.

The above mentioned studies show that intersection of IDS and Machine Learning is actively researched and KDD99 is the most used dataset in this domain. However, they do not provide enough statistics about this dataset. This study tries to present more comprehensive study to find satisfactory answers to the mentioned questions by giving more statistics and checklist for guidance.

2.3. How review articles are selected?

KDD99 is an old dataset and used in very different domains, in addition to Intrusion Detection. To find relevant articles via primary search engines, the following key words are used:

1. Intrusion Detection
2. KDD 99, KDD99
3. Anomaly Detection
4. Data Stream Learning

The names of the search engines used are the following:

1. Thomson Reuters (Web of Knowledge) `http://apps.`webofknowledge.com
2. Elsevier Science Direct `http://www.sciencedirect.com/`

3. Springer Link (Springer) http://link.springer.com/advanced-search
4. IEEE http://ieeexplore.ieee.org
5. Wiley http://onlinelibrary.wiley.com/advanced/search
6. Taylor and Francis Online http://www.tandfonline.com

The articles that use DARPA dataset, instead of KDD99 are ignored in this study. We did not include any articles from conferences, book chapters or lecture notes, only research articles. Every article's journal is searched in Sci Expanded Database. We know that some of the articles may be missed in this way, but we believe that 95% of published articles are included in this review chapter. This shows that even though KDD99 is very old, it is still very widely used. We have included some articles that used KDD99 from different domains in this way. We have found total of 215 articles in our study from 2010–2017.

2.4. Datasets: DARPA, KDD99, and NSL-KDD

Figure 2.3 and Table 2.1 give overall summary for related datasets (DARPA, KDD99, and NSL-KDD) in this study. DARPA is a base raw dataset. KDD99 is the feature extracted version of DARPA dataset. NSL-KDD is the duplicates removed and size reduced version of KDD99 dataset. Dataset statistics extracted from reviewed articles are given in Section 2.6.

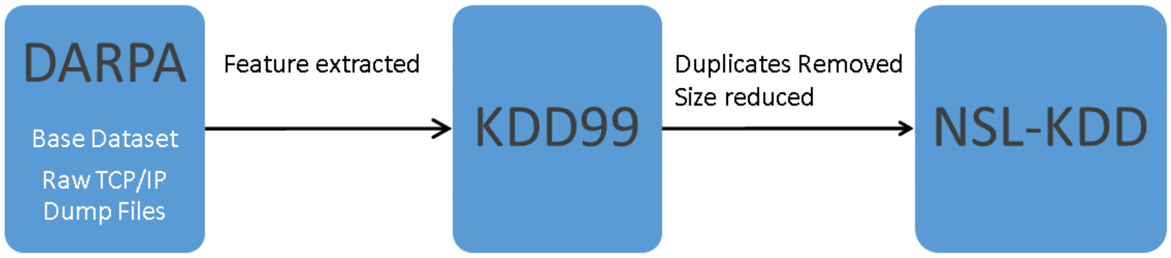


Figure 2.3 **The relation between main and extracted datasets**. KDD99 is created from DARPA, NSL-KDD is created from KDD99.

Table 2.1 **DARPA, KDD99, and NSL-KDD datasets information**

Name	Training Size	Testing Size	Note
DARPA 99	6.591.458 kb (6.2gb)	3.853.522 kb (3.67gb)	Base Dataset. Raw TCP/IP Dump files
KDD99	4898431	311029	Features extracted and preprocessed for machine learning
NSL-KDD	125973	22544	Duplicates removed, size reduced

2.4.1. DARPA dataset

MIT Lincoln LAB hosted DARPA-sponsored IDS-event in 1998 [30]. In this DARPA event, an attack scenario to an Air-Force base was simulated. One year later, in 1999, this event was repeated [31] with improvements suggested by computer security community [32]. DARPA dataset consists of host and network dataset files. Host dataset, IDS bag, is small dataset that contains system calls and is less used than its network counterpart. Network dataset consists of seven weeks of raw TCP/IP dump files. Since DARPA dataset consists of raw files, researchers need to extract features from these files to use them in machine learning algorithms. First two weeks were attack free; therefore, it is suitable for training anomaly detection algorithms. In the remaining five weeks, various attacks were used against simulated Air-Force base, [33]. KDD99 dataset was created from DARPA network dataset files by Lee and Stolfo [23] for this DARPA sponsored event.

2.4.2. KDD99 dataset

Lee and Stolfo [23], one of the participating teams of the DARPA event, gave their feature extracted and preprocessed data to Knowledge Discovery and Data Mining (KDD) yearly competition [34]. Pfahringer [35] won KDD 99 competition using mixture of bagging and boosting. Most articles compare their results with his result. KDD99 can be easily used in machine learning studies; therefore, it is much more used in IDS and MLR than DARPA dataset.

KDD99 has the following characteristics:

1. KDD99 has two week's of attacks-free instances and five week's of attack instances, making it suitable for anomaly detection.
2. Output classes are divided into 5 main categories: DOS (Denial of Service), Probe, R2L (Root 2 Local), U2R (User 2 Root), and Normal.
3. KDD99 Dataset contains 24 attack types in training and 14 more attack types in testing for total of 38 attacks. These 14 new attacks theoretically test IDS capability to generalize to unknown attacks. At the same time, it is hard for machine learning based IDS to detect these 14 new attacks [36].
4. KDD99 is heavily imbalanced dataset to attack instances. Approximately 80% percent of flow is attack traffic (3925650 attack instances in total 4898430

instances). Normally, typical network contains approximately 99.99% percent of normal instances. KDD99 violates this principle. Most articles needs to re-sample dataset to conform to typical network normality assumption, particularly anomaly detection articles.

5. U2R and R2L attacks are very rare in KDD99 (Table 2.2).

6. Duplicate records in both training and testing datasets bias results for frequent DOS attacks and normal instances.

7. KDD99 is a large dataset; therefore, most studies use a small percentage of it. Following studies among reviewed articles [37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49] claim that KDD99 is a very large dataset for machine learning algorithms; correspondingly, they use very small percentage of KDD99.

Table 2.2 **KDD99 attack distribution**

	Training Size	(%)	Test Size	(%)
Normal	972781	19.85	60593	19.48
DOS	3883390	79.27	231455	74.41
Probe	41102	00.83	4166	01.33
U2R	52	00.001	245	00.07
R2L	1106	00.02	14570	04.68
Total	4898431	100	311029	100

Numerous shortcomings of KDD99 with respect to IDS are well documented in literature, [13, 50, 51, 52, 15].

2.4.3. NSL-KDD dataset

To reduce deficiencies of KDD99 dataset for machine learning algorithms, NSL-KDD dataset is introduced by [22]. NSL-KDD has been generated by removing redundant and duplicate instances; thus decreasing size of dataset. Since NSL-KDD is a re-sampled version of KDD99, IDS deficiencies remain. NSL-KDD is a very popular dataset in intersection of machine learning and IDS literature due to its small size compared to full KDD99 dataset. Usage of NSL-KDD is increasing in recent years, see Figure 2.4.

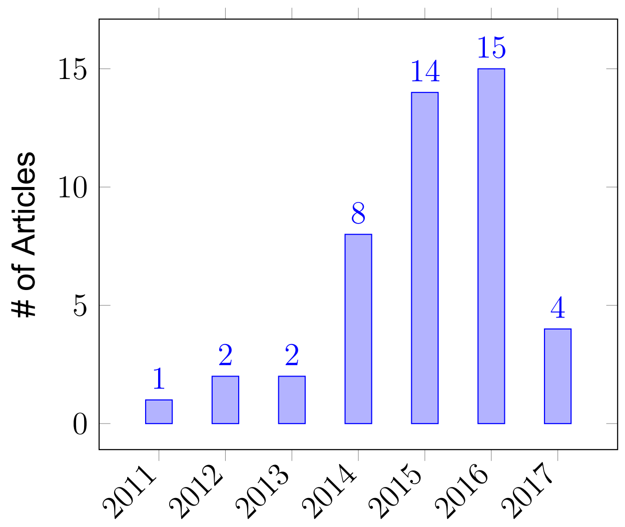


Figure 2.4 **NSL-KDD dataset usage by years.** (Year 2017 values includes first 4 months only.)

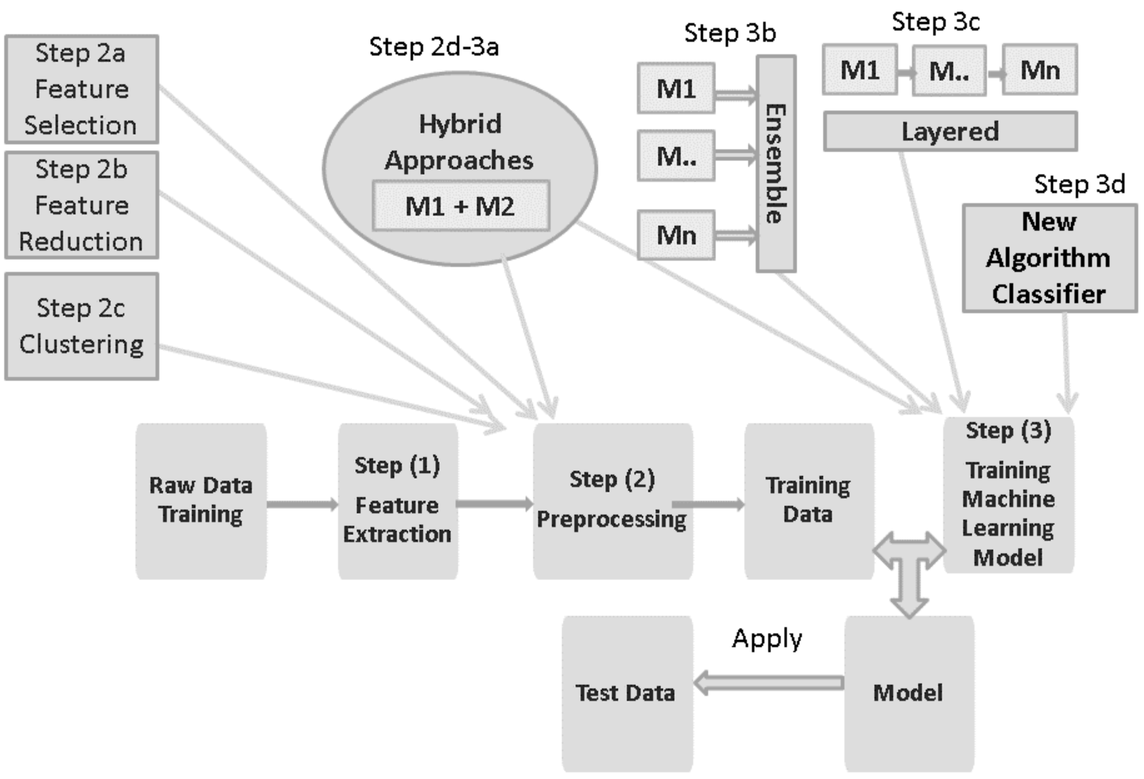


Figure 2.5 **General machine learning flow chart.** Almost all of the reviewed articles make their contribution in steps 2a-2d and 3a-3d. Table 2.3 shows article counts for these contributions.

2.5. General machine learning work flow using KDD99

Figure 2.5 shows general machine learning work flow using any dataset. This work flow contains 3 main steps relevant to our discussion. These are step (1) feature extraction, step (2) preprocessing, and (step 3) training machine learning model. Normally, feature extraction step (1) is the most important step in machine learning [53]. As KDD99 is a feature extracted dataset, this step is unnecessary.

Most reviewed studies made their contributions to preprocessing step (2) or training machine learning model step (3). For Step (2) preprocessing, reviewed articles used 4 different techniques: (2a) Feature Selection, (2b) Feature Reduction or Feature Transformation, (2c) Clustering, and (2d) Hybrid Approaches. Feature selection (2a) is using various algorithms to reduce number of existing 41 features. Feature Transform/Reduction (2b) is to change feature space of dataset to another space, then optionally reduce number of transformed features. For example, principal component analysis is a popular choice among reviewed studies (Table 2.6). Clustering (2c) is reduce features or instances using a clustering algorithm, for example k-means clustering. Hybrid Approaches (2d-3a) is using combination of two different algorithms for preprocessing or training machine learning model step. Most of the time, a feature selection/reduction/machine learning algorithm is hybridized with an optimization algorithm (for example: particle swarm optimization).

For Step (3) training machine learning model, reviewed articles can be categorized using 4 different type of techniques: (3a) Hybrid Approaches, (3b) Ensemble, (3c) Layered, and (3d) New Algorithm Classifier. An example of Step (3a) Hybrid Approaches is training a neural networks with genetic algorithms instead of back propagation. Ensemble approach, Step (3b), is a parallel combination of different machine learning algorithms. Layering, Step (3c), is a serial combination of different machine learning algorithms. New Algorithm Classifier, Step (3d), means the applied algorithm may be entirely new or used the first time in IDS.

According to given work flow, contributions of the most reviewed articles may be more than two. For example, using a new optimization algorithm for feature selection and classification is counted as both feature selection and hybrid in this review. Also, using principal component analysis for feature reduction and using optimization algorithm to train a classifier is counted as both hybrid and feature reduction. Table 2.3 shows categorization of articles according to the work flow.

Table 2.3 **Evaluating the reviewed articles regarding to machine learning model Figure 2.5**

Contribution(Novelty)	Article Count	Figure 3
Hybrid	71	(2d,3a)
New Algorithm Classifier	66	(3d)
Feature Selection	55	(2a)
New Algorithm Anomaly Detection	54	(3d)
Layered(Cascaded)	35	(3c)
Feature Reduction	35	(2b)
New Algorithm Clustering	35	(2c)
New Optimization Algorithm	28	(2d,3a)
Ensemble	22	(3b)
Data Streams	20	
Agent Based	16	
Feature Transformation	11	(2b)

2.6. KDD99 descriptive statistics

Different from previous review studies, we present more descriptive statistics to evaluate published studies in focused period (2010–2016). Other extracted statistics are presented before such as Figure 2.1, Figure 2.4 and Table 2.3. In addition, the following statistics have been extracted from the reviewed 215 studies:

1. Classification output classes
2. Training and Testing Dataset Usage
3. Use of Cross Validation
4. Dataset sizes used in training and testing machine learning algorithms
5. Applied algorithms in proposed method
6. Classifiers used for comparison
7. Software Toolbox Usage
8. Other Datasets used in Reviewed Studies
9. Performance Metrics used in Experiments
10. IDS vs Not IDS
11. Main IDS Type according to study

These descriptive statistics are presented using figures and tables and have been discussed in detail.

2.6.1. Classification output Classes in the reviewed studies

The output classes can be binary or multi classes when machine learning algorithms are applied to the KDD99. Table 2.4 shows output classes in reviewed articles. Multi class 5 are DOS, Probe, Normal, U2R and R2L as explained in Section 2.4.2. Multi Class X selects subsets of 23 classes of KDD99, for example an output that consists of 7 attacks and normal gives results for 8 output classes. Multi Class X studies are not comparable to other studies.

Table 2.4 **Comparison of the published studies based on classification output classes**. Experiments in articles may have more than one output class such as Binary and Multiclass 5; therefore total article count in this table is more than 215

Classification Output	Article Count
Binary (Attack/Normal)	197
Multiclass 5 (DOS/Probe/U2R/R2L/Normal)	73
Multiclass 23 (22 attacks/Normal)	4
No Binary: Gives other result	8
Multi Class X (Subset of 23)	21

2.6.2. Training and testing dataset usage

Table 2.5 **Confusion matrix for training and test set usage**. Normally, only diagonal of matrix should have values, but most of the reviewed studies use KDD99 training dataset for both testing and training purposes.

		KDD99	
		Training	Test
Reviewed Study	Training	212	8
	Test	161	58

Normally, in machine learning studies, datasets should be divided into training and testing datasets. Machine learning algorithms should be trained on training dataset and be tested on test dataset that is entirely separate from training datasets. Considering this usage, DARPA, KDD99 and NSL-KDD datasets contains two parts, training and testing. As mentioned before these two parts have different attacks and different probability distributions. Training a machine learning algorithm in a subset of KDD99 training dataset; then, testing trained model in another subset gives optimistic results. Generally, machine learning algorithms should be trained on KDD99 training dataset and tested on KDD99 testing dataset.

Table 2.5 shows training and testing dataset usage in the reviewed articles. Most reviewed articles (212) used KDD99 training dataset for training; but, 8 articles behaved differently. These 8 articles either merged training and testing dataset then re-sampled or used training dataset for testing purposes. The main reason for this type of dataset usage is to reduce difference between training and testing dataset. Only about 26% of reviewed articles (58) used real testing dataset. Others (161) used KDD99 training dataset as both testing and training purposes. This type of dataset usage provides optimistic, thus non-realistic results for these studies. Table 2.5 shows that most of the literature have used re-sampled version of KDD99 training dataset for both training and testing.

2.6.3. Cross validation

K-fold cross validation is one of the suggested techniques in training machine learning models. Among the reviewed 215 studies, only 45 (%20) studies applied cross validation, while 170 (%80) studies did not apply cross validation.

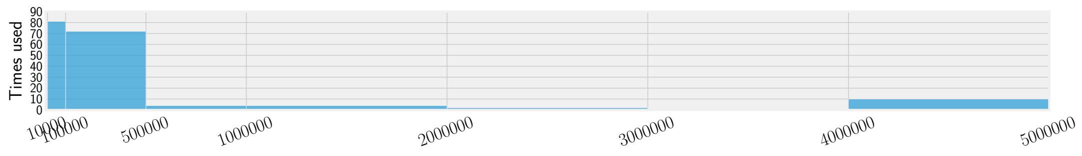
2.6.4. Dataset sizes used in training and testing ML algorithms

In this review, 12 articles claimed KDD99 is a large dataset for machine learning research and used smaller subset of full dataset. Figure 2.6 and Figure 2.7 shows that training and testing dataset usage is skewed to small sizes. That is, most articles worked with small dataset sizes in reviewed studies. The smallest 20 training datasets contain 100 to 1000 instances, while smallest 20 testing datasets contain 100 and 1112 instances. These numbers are small compared to full size (4.9 Million instances) KDD99 dataset. Using very small dataset sizes may be unacceptable from the viewpoint of statistical analysis.

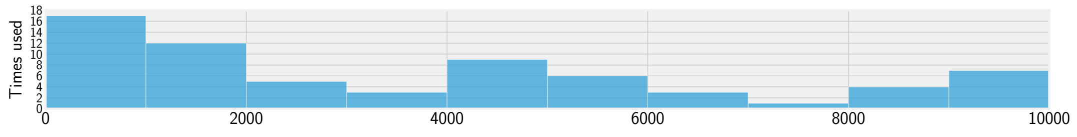
Figure 2.7 shows testing dataset sizes used in reviewed articles. Figure 2.7 only shows usages of testing dataset that are less than full size (311029 instances).

2.6.5. Applied algorithms in proposed methods

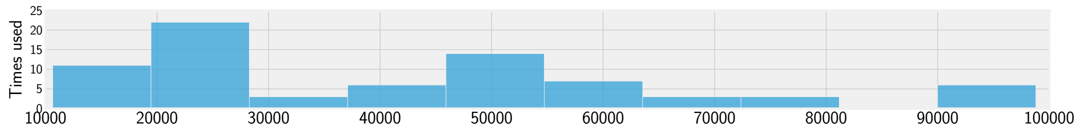
Various algorithms have been used in KDD99 based IDS research. Table 2.6 shows algorithms that are used by the reviewed studies. If a classifier, for example support vector machines, is used for comparison purposes, it is included in both Table 2.6 and Compared Classifiers Table 2.7.



KDD99 Training Dataset is used 235 times for all sizes



KDD99 Training Dataset is used 67 times for sizes less than 10.000



KDD99 Training Dataset is used 75 times for sizes between 10.000 and 100.000

Figure 2.6 **KDD99 training usage sizes.** Most of the usage is with low sizes.

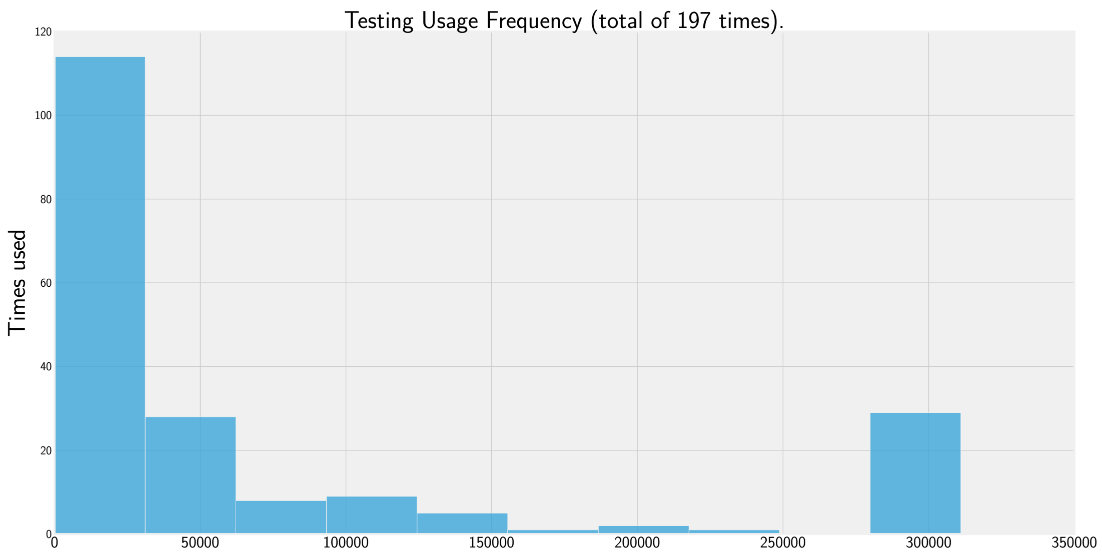


Figure 2.7 **KDD99 testing usage sizes.** Most of the low size usages comes from resampling of KDD99 Training dataset.

2.6.6. Classifiers used for comparison

Generally, most studies compare their results with other methods in their experiments. In KDD99 based machine learning research, most comparisons are made against other classifiers. Table 2.7 shows classifiers used for comparison in

Table 2.6 **Most used algorithms in the literature**. Algorithms used less than 3 are not shown.

Name	**Article Count**
support vector machines	32
decision tree	26
genetic algorithm	17
k-means clustering	16
k-nearest neighbors	14
naive bayes	14
principal component analysis	14
neural networks (multilayerperceptron)	11
particle swarm optimization	10
rough sets	8
artificial immune system	7
genetic programming	6
neural networks (selforganizingmap)	5
mutual information	5
random forest	5
bayesian network	5
optimum-path forest	4
extreme learning machine	4
fuzzy rules mining	4
rule based learner	3
fuzzy clustering	3
adaboost	3
clustering	3
multivariate correlation analysis	3

the reviewed studies. Some rare articles in this review were not machine learning articles, even though they use KDD99. These rare articles did not compare their suggested approaches with other classifiers. Also, some articles that proposed new methods for IDS, have not compared their proposed method with other classifiers. For these two types of articles, Table 2.7 includes *None*. For *Literature* entries, some articles did not compare their methods and their datasets using software toolboxes but only reported literature results. In our opinion, all machine learning IDS articles should use software toolboxes (Table 2.8) to compare their methods with common methods instead of only reporting literature results. Main reason for this advice is science reproducibility since every article is a bit different (sampling strategy, randomize seed, and different sizes for datasets).

Table 2.7 **Classifiers used for comparison in the experiments of reviewed studies**. Classifiers used less than 4 are not shown.

Classifier	**Article Count**
support vector machines	48
naive bayes	44
literature(no experimental comparison)	37
decision tree(j48)	30
k-nearest neighbors	27
neural networks (multilayerperceptron)	26
none(Not compared with other methods)	24
decision tree	17
random forest	16
bayesian network	15
adaboost	9
k-means	9
neural networks (selforganizingmap)	7
neural networks (radialbasisfunction)	7
local outlier factor	6
decision tree(cart)	5
part	5
rule based learner(jripper)	4
naive bayes tree	4
random tree	4
one class support vector machines	4
decision tree(c4.5)	4

2.6.7. Software used in reviewed studies

Many software toolboxes has been used in IDS studies. Table 2.8 summarizes software used to implement or compare algorithms in the articles. Most articles (78) did not give any information about applied software. This restricts reproducibility of applied method. Based on Table 2.8, Weka is widely used for classifier comparison even if it is not used for implementation. Matlab and LibSVM are also used for comparison. Most of the proposed methods are implemented using general purpose programming languages. As a remarkable note, although Python (2) and R (1) are thought as the languages of data science and machine learning [54], they were among the least used tools.

2.6.8. Different datasets used in reviewed studies

In addition to KDD99, different datasets were also used in the reviewed articles, Table 2.9. Non IDS datasets in this review show that KDD99 is used as just another dataset in some studies. NSL-KDD is re-sampled version of KDD99 as explained in Section 2.4.3. Some studies used both NSL-KDD and KDD99, while others used only NSL-KDD dataset. Other IDS datasets are used rarely. For example, ISCX and

Table 2.8 **Software used in the reviewed articles**. Weka, Matlab, and LibSVM are mostly used for comparison purposes. General purpose programming languages are used for implementation. Software that are used less than 2 are not shown.

Software Tool/Package	Article Count
NoInformation	101
Weka	47
Matlab	42
Java	16
LibSVM	14
C++	7
Hadoop	5
MOA	6
R	4
Pascal	3
CSharp	3
Liblinear	2
Python	2

Kyoto have been used only 6 times. These numbers show lack of recent IDS dataset in the literature.

Table 2.9 **Most used datasets**. * denotes IDS datasets. Datasets that are used less than 3 are not shown.

Dataset Name	Article Count
KDD99*	183
NSL-KDD*	46
Iris	11
DARPA*	10
Glass	9
Shuttle	7
Kyoto*	6
ISCX*	6
Breast Cancer	6
Ionosphere	5
Poker Hand	5
Pima	5
Lymphography	5
Wine	5
Synthetic Data	5
Abalone	5
Synthetic	4

Table 2.10 **Performance metrics used**. Usage of performance metrics are highly irregular. Some articles does not give any metrics (*). Metrics used less than 3 are not shown.

Performance Metric	Article Count
Detection Rate	186
False Positive (FalseAlarm)	95
Training Time	62
Testing Time	50
False Negative	30
Confusion Matrix Multi 5 General	29
Error Rate	28
ROC-Curve	28
True Positive	28
Precision	20
F-Measure	18
Number Of elected Features	16
True Positive (Sensitivity-Recall)	15
ROC-Area Under Curve	15
Recall	13
Cost Per Example	12
True Negative	12
Correlation Coefficient	9
Confusion Matrix Binary	9
Computational Complexity Time	7
Memory Usage	7
Root Mean Square Error	7
Specificity	6
Sensitivity	6
None*	6
Geometric Mean	5
Computational Complexity	5
CPU Usage	4
True Negative (Specificity)	4

2.6.9. Performance metrics used in reviewed studies

Various performance metrics can be used to evaluate to machine learning algorithms. Table 2.10 summarizes which metrics are provided in the 215 articles reviewed. Detection rate is most consistent metric provided; although some articles fail to provide this metric. For example, some articles gave figures for their detection rate but did not give an exact number; therefore, reader has to guess about its value. Other articles gave 5-class detection rates but did not give overall detection rate for comparison. If a researcher would like to compare the given results with other articles, it is often impractical since dataset sizes differ greatly from article to article. Some articles gave detection rate by class but failed to provide number of class instances therefore it is impossible to get single detection rate for attack versus normal.

Some articles did not present information about used testing dataset. Machine learning algorithms get different results in KDD99 train and testing dataset as mentioned in Section 2.4.2. Therefore; it is important for articles that use KDD99 to indicate that if they used training or testing dataset of KDD99.

Other performance metrics differ widely in our reviewed articles. Computational Complexity metrics were not given in most articles. Also training time was given 63 (29%) times, and testing time was given 51 (23%) times. Considering importance of these two metrics, their usage is not high.

Generally, authors suggest that following metrics should be given in the KDD99 research articles. (1) Detection Rate, (2) Confusion matrix, (3) Training Time, (4) Testing Time and, (5) Computational Complexity for newly proposed methods.

2.6.10. Main IDS type according to reviewed Studies

Figure 2.8 shows which IDS methodologies are used in the collected articles. Total count is more than 215, since most articles use more than one methodology. KDD99 is a popular choice for both machine learning and anomaly detection studies.

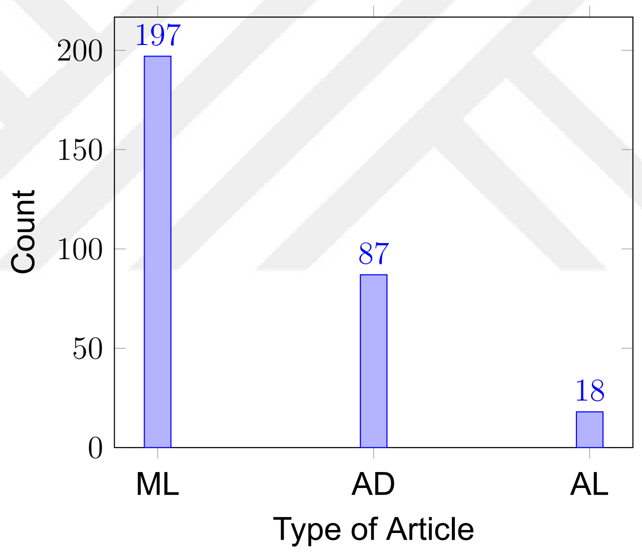


Figure 2.8 **Article counts by methodology** (ML:Machine Learning, AD:Anomaly Detection, AC:Alert Correlation)

2.6.11. IDS vs not IDS studies

Figure 2.9 shows how many articles claim that they are IDS studies among the reviewed articles. Even though, IDS articles form the majority, number of Non IDS articles shows that KDD99 is also widely used in other domains.

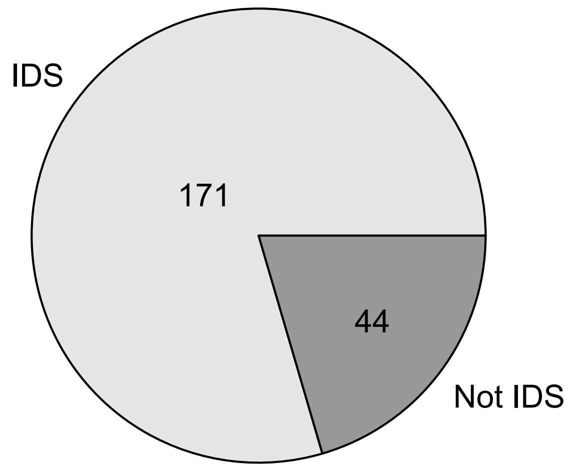


Figure 2.9 **IDS and Not IDS articles in 2010–2017 for total of 215 articles**. KDD99 is mostly used in IDS studies; but, some studies use it also, especially feature selection and data streams studies.

2.7. Suggested checklist for avoiding common mistakes

After evaluating 215 studies from the SCI-indexed 90 journals, we suggest a checklist for researchers who want to apply a machine learning or IDS method using KDD99 or other large datasets. The suggested checklist has been provided considering common mistakes and strengths points of the reviewed studies. This checklist could be useful for future studies in a similar areas. The instructions in checklist are as follows:

- Point out training and test dataset clearly.
- If the target dataset is KDD99, identify if full dataset or a portion of dataset is used.
- Specify train, test and validation dataset sizes in a table.
- Increase reproducibility of the study by giving software package, re-sampling strategy, and random seeds.
- To evaluate the classification result, provide confusion matrix, detection rate, training time and testing time.
- Compare the result of the proposed approach with other most used methods.
- Determine the number of output classes. For KDD99 using 5 or 23 classes will be preferred.

2.8. Chapter discussion

In this chapter, 215 recent studies that use KDD99 dataset between 2010 and 2016 have been reviewed. A different review process is followed from previous reviews in the same area. Instead of finding the major contributions to the area, descriptive statistics are extracted. Review results show the following findings: (1) Even though KDD99 is an 17-years-old dataset, it is still widely used in IDS and machine learning research. (2) Decision tree derivatives and support vector machines are the most applied algorithms. (3) Weka and Matlab are the most used software toolboxes, even though most studies did not give any information about software usage. (4) Detection rate is the most used performance metric to show classification quality. Additionally, considering common errors and strengths of the reviewed works, a checklist has been suggested to improve the research quality in similar areas.

Table 2.11 **Journals and article counts**

Journal Name	Article Count	Journal Name	Article Count
Expert Systems with Applications	27	Concurrency and Computation: Practice and Experience	1
Security and Communication Networks	14	IEEE Transactions on Parallel and Distributed Systems	1
Information Sciences	8	Communications in Statistics - Simulation and Computation	1
Knowledge-Based Systems	8	Cluster Computing	1
Applied Soft Computing	7	China Communications	1
Neurocomputing	7	Artificial Intelligence Review	1
Neural Computing and Applications	6	Artificial Intelligence	1
IEEE Transactions on Knowledge and Data Engineering	5	Arabian Journal for Science and Engineering	1
Applied Intelligence	5	Applied Mathematics and Information Sciences	1
The International Arab Journal of Information Technology	5	Applied Artificial Intelligence	1
Pattern Recognition	5	Acta Polytechnica Hungarica	1
International Journal of Computational Intelligence Systems	4	IEEE Transactions on Neural Networks and Learning Systems	1
Computer Communications	4	International Journal of Communication Systems	1
The Journal of Supercomputing	4	IEEE Transactions on Smart Grid	1
Engineering Applications of Artificial Intelligence	4	IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews	1
Journal of Intelligent and Fuzzy Systems	3	Telecommunication Systems	1
Machine Learning	3	Soft Computing	1
Journal of Network and Computer Applications	3	Signal, Image and Video Processing	1
International Journal of Computer Science and Network Security	3	Sensors	1
IETE Journal of Research	2	Programming and Computer Software	1
IEEJ Transactions on Electrical and Electronic Engineering	2	Pattern Recognition Letters	1
International Journal of Innovative Computing, Information and Control	2	PLoS ONE	1
Tsinghua Science and Technology	2	Neural Processing Letters	1
The Scientific World Journal	2	Mobile Networks and Applications	1
EURASIP Journal on Wireless Communications and Networking	2	Mobile Information Systems	1
The Computer Journal	2	Knowledge and Information Systems	1
Defence Science Journal	2	Journal of the Faculty of Engineering and Architecture of Gazi University	1
Data Mining and Knowledge Discovery	2	Journal of Visual Languages and Computing	1
Journal of Information Science and Engineering	2	Journal of Parallel and Distributed Computing	1
Journal of Intelligent Information Systems	2	Journal of Network and Systems Management	1
Simulation Modelling Practice and Theory	2	Journal of Advanced Research	1
Genetic Programming and Evolvable Machines	2	Iranian journal of fuzzy systems	1
IEEE Latin America Transactions	2	Iranian Journal of Science and Technology-Transactions of Electrical Engineering	1
Computational Intelligence and Neuroscience	2	International Journal on Artificial Intelligence Tools	1
Intelligent Automation and Soft Computing	2	International Journal of Systems Science	1
IEEE Transactions on Cybernetics	2	International Journal of Pattern Recognition and Artificial Intelligence	1
IEEE Transactions on Computers	2	International Journal of Fuzzy Systems	1
Mathematical Problems in Engineering	2	International Journal of Computers Communications and Control	1
IEEE Systems Journal	1	Turkish Journal of Electrical Engineering and Computer Sciences	1
IEEE Transactions on Dependable and Secure Computing	1	Intelligent Data Analysis	1
Future Generation Computer Systems	1	IETE Technical Review	1
Wireless Personal Communications	1	IET Information Security	1
Computing and Informatics	1	IEICE Transactions on Information and Systems	1
Frontiers of Computer Science	1	IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences	1
Discrete Dynamics in Nature and Society	1	ACM Transactions on Autonomous and Adaptive Systems	1

3. GENETIC ALGORITHMS BASED FEATURE SELECTION AND CLASSIFIER ENSEMBLE

Chapter abstract

As information systems develop, intrusion detection systems (IDS) become more and more important. IDS can be considered as a classification problem. Nowadays, to improve accuracy of classifiers, it is recommended to use classifier ensembles instead of single classifiers. One of the important steps of classification applications is feature selection step. The performance of classification depends on selected features and applied classification approach. This study proposes to use genetic algorithms for both feature selection and weight selection for weighted classifier ensemble in IDS. Number of classifiers used in ensemble changes between 3 and 7 with weighted combination method. Following classifiers have been used: Adaboost, Decision Tree, Logistic Regression, Naive Bayes, Random Forests, Gradient Boosting, and K-Nearest Neighbor. Well-known intrusion detection dataset, NSL-KDD, has been used in experiments. The results of the proposed method have been compared with weighted vote, simple vote, and probability vote ensemble methods, and also with single classifiers.

3.1. Chapter introduction

Computer-based systems are part of our daily life and are a fundamental part of production systems and services. Billions of users connect to Internet using computer based systems. This complex network makes it easy to get information but it is vulnerable to attacks. Day by day, complexity and numbers of attacks is increasing. Intrusion detection systems (IDS) are used to protect information systems against those attacks[11].

Intrusion detection systems are categorized into two types according to detection methodology[11]. Signature-based IDS keeps a signature database for every attack. Every incoming request to system is compared with this database. According to this comparison, request is labeled as either an attack or normal request. Anomaly-based IDS starts with the definition of what normal is. This definition normally is based on statistical or machine learning principles. Every incoming request to system is compared with this normal definition. According to conformity of the request to this definition, request is labeled as either an attack or normal[11].

General IDS and machine learning work has been studied well in related review works [20, 21, 55]. According to Özgür and Erdem [55], 142 studies were published in SCI indexed journals between 2010–2015 that were intrusion detection related. Among these 142 studies, 118 of them use IDS and machine learning techniques together, Table 3.1 and Figure 3.1.

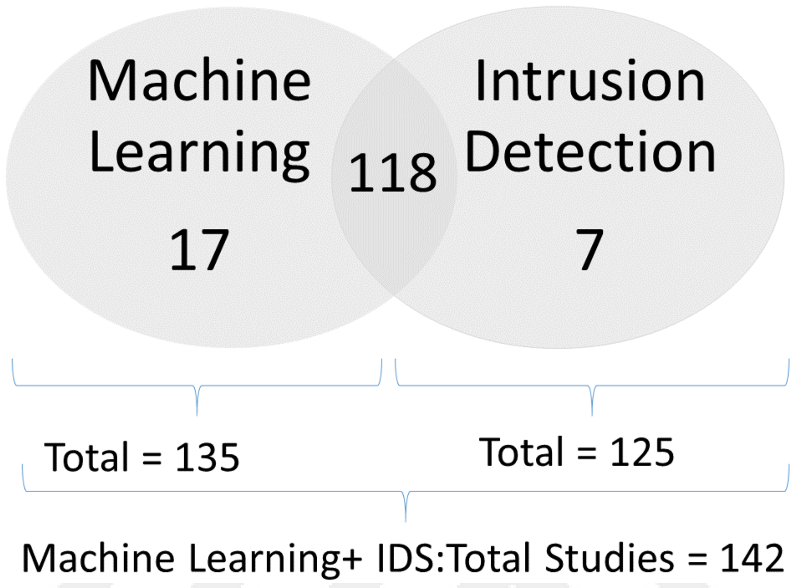


Figure 3.1 IDS and ML articles counts published in SCI indexed journals

Feature selection step is a very important step in classification of high dimensional datasets. Since feature selection is an NP-hard problem, different techniques have been applied [53]. Yıldız et al. [56] used genetic algorithms for feature selection in breast cancer classification problem. Similarly, evolutionary algorithms are used for feature selection and finding feature weights[57]. Since datasets for IDS are comparably large in feature and instance size, training and testing of machine learning algorithms takes long time. To decrease training and testing times, the feature selection step has been widely used in IDS applications [20, 55].

Single classifier machine learning algorithms are widely used in IDS studies. As an example, Neural networks has been applied on KDD99 dataset [58] [59]. In other studies [60] [61] [62], multiple machine learning algorithms have been fused. According to Özgür and Erdem [55], 14 studies used classifier ensemble among 142 IDS and machine learning studies. When deciding ensemble outcome, single classifier may use single vote or probabilistic vote. In addition to these two voting methods, votes may be changed using weights. Normally, weights for ensemble can be found empirically or using domain knowledge. Other studies have used heuristic optimization techniques for the same purpose [63, 64, 65].

Table 3.1 **Methods used in IDS and machine learning [55]**

Method	**Article Count****
IDS	125
Machine Learning	135
Hybrid	50
Feature Selection	34
Ensemble	14
Genetic Algorithms	16
Total	**142**

**Note that if an article uses two techniques, it is counted twice.

In IDS and machine learning systems, hybrid algorithms started to become more important. In recent 142 studies of IDS and machine learning, 50 studies applied hybrid methods[55]. This number makes hybrid algorithms by far most popular technique in machine learning and IDS, see Table 3.1. Hybrid methods combine different algorithms in the same system. Usually, heuristic optimization techniques have been used together with other machine learning methods. GA are among the most used heuristic optimization techniques. GA is a global optimization algorithm and applied to wide range of different NP-hard problems. Dener et al. [66] used genetic algorithms on the well-known problem of shortest distance. Similarly, GA is used in other engineering problems: Üstündağ et al. [67] used GA on de-noising of radar signals, and Gürsü [68] used GA on estimation of high voltage problems. Additionally, Yıldız et al. [56] used GA on breast cancer domain for feature selection and data fusion.

Sylvester and Chawla [64] proposed GA-based method named EVEN, that find weights of classifiers. Maghsoudi et al. [65] used GA to find classifier weights in hyper spectral imaging. Besides, GA has been used for feature selection and classifier ensemble weights finding in the literature. But, according to best of authors' knowledge, there is no study that applies GA for selecting features and finding weights of combined classifiers simultaneously.

In proposed method, feature selection and weights finding problem solved together in one objective function. It is called Genetic Algorithms based Feature Selection and Weights Finding, in short form GA-FS-WF. Effects of these two steps are given in the previous studies. Feature selection reduces training and testing time of classifiers, which are critical for IDS[69]. At the same time, single classifiers are combined via different weights. Classifier combination improves accuracy of overall system. By applying GA to these two steps together, error propagation from first step to other steps is prevented. In short, GA-FS-WF improves accuracy of overall system while reducing training and testing time.

To show effect of classifier size in the proposed classifier ensemble, 3 through 7 classifiers are combined in the experiments. Considering previous studies, following single classifiers have been selected: Adaboost, Decision Tree, Logistic Regression, Naive Bayes, Random Forests, Gradient Boosting and K-nearest neighbor. Statistical tests of ANOVA and t-test are applied to show that proposed system's classification accuracy is statistically significant. In summary, GA-FS-WF method has following contributions:

1. Feature selection and weights finding problem solved simultaneously.
2. Since NSL-KDD dataset is very large and problem is NP-Hard, GA that is a heuristic optimization method is used.
3. Training and testing time reduced since both of these steps are solved simultaneously.
4. Effect of classifier size in multiple classifier ensemble is investigated. (This study combined 3–7 classifiers.)
5. Multiple classifier combination methods – simple vote, probability vote and weighted vote– have been compared.
6. Classification accuracy of proposed method is shown using statistical tests of ANOVA and t-test.

3.2. Materials and methods

3.2.1. NSL-KDD and KDD99 datasets

Relatively old datasets – DARPA,KDD99 and NSL-KDD – are still used in IDS research [16, 17, 21, 55]. DARPA IDS dataset was created by MIT Lincoln Labs in 1998 [30]. Feature extracted version of DARPA dataset [23] was used in KDD99 competition. To improve machine learning algorithms performance on KDD99, NSL-KDD was created [22]. NSL-KDD is a feature extracted and duplicate records deleted version of KDD99. Even though deficiencies of these datasets are well-known [13, 15], they are still the most used datasets in the IDS machine learning research [21, 55, 16, 17].

Table 3.2 **Dataset information**

Name	**Training Size**	**Testing Size**	**Note**
DARPA 99	6.2GB	3.67GB	Base dataset. Raw TCP/IP Files
KDD99	4898431 Instances	311029 Instances	Feature extracted and preprocessed
NSL-KDD	125973 Instances	22544 Instances	Duplicates removed and size reduced

Size and general attributes of these dataset are given in Table 3.2. In this study, NSL-KDD dataset has been used for training, validation and test purposes, Table 3.3. Training part of NSL-KDD has been used for training. Testing part of NSL-KDD has been divided to two parts. First 10% has been used for validation, while remaining 90% part has been used for testing.

Table 3.3 **Dataset information used in experiments**

	Training	Validation	Testing
%	100	10	90
Instance	125973	2254	20290

3.2.2. Classifier performance

Classifier performance can be measured with different metrics. Normally, IDS output consists of two classes (normal and attack). Classifiers may make mistakes when classifying instances. Sometimes normal instances can be classified as attack and vice versa. All possible cases for this situation:

1. Attack instance is predicted as attack: True Positive (TP)
2. Attack instance is predicted as normal: False Negative (FN);
3. Normal instance is predicted as attack: False Positive (FP)
4. Normal instance is predicted as normal: True Negative (TN);

Table 3.4 Confusion matrix

		Actual Class	
		Attack	Normal
Predicted Class	Attack	True Positive	False Positive
	Normal	False Negative	True Negative

$$ClassificationAccuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

According to this four case Table 3.4 is created. The most used metric for classifier performance, accuracy is defined in Equation 3.1.

3.2.3. Feature selection

In classification of large dataset, an important step is the feature selection [53], since increasing number of features in a dataset will increase training and testing time of machine learning algorithms. Additionally, some machine learning algorithms are sensitive to redundant features and may loose accuracy while working with them[53]. Therefore, feature selection is a widely used pre-processing step [20, 53]. Feature selection step aims to decrease number of features while preserving overall performance of the classifier.

KDD99 and NSL-KDD dataset has 41 features and one class. Since these datasets has large number of instances, training and testing times are longer. In these datasets, feature selection step is widely used [55]. According to Table 3.1, 34 out of 142 study has applied feature selection.

3.2.4. Classifier ensembles

To improve classification accuracy, more than one machine learning algorithm may be combined. This process is known with different names in different domains. Some of these names are: Classifier fusion, classifier ensemble, classifier combination, mixture of experts, committees of neural networks, voting pool of classifiers, and others [61]. Generally three methods are used to combine classifiers[70]. These are: simple or majority voting, probability voting, and weighted voting. In simple voting, each classifiers uses one vote and most voted class will be the output result. In probability voting, each classifier uses its confidence in output as probability vote. In weighted voting, probability output of each classifier will be multiplied with weights,Figure 3.2.

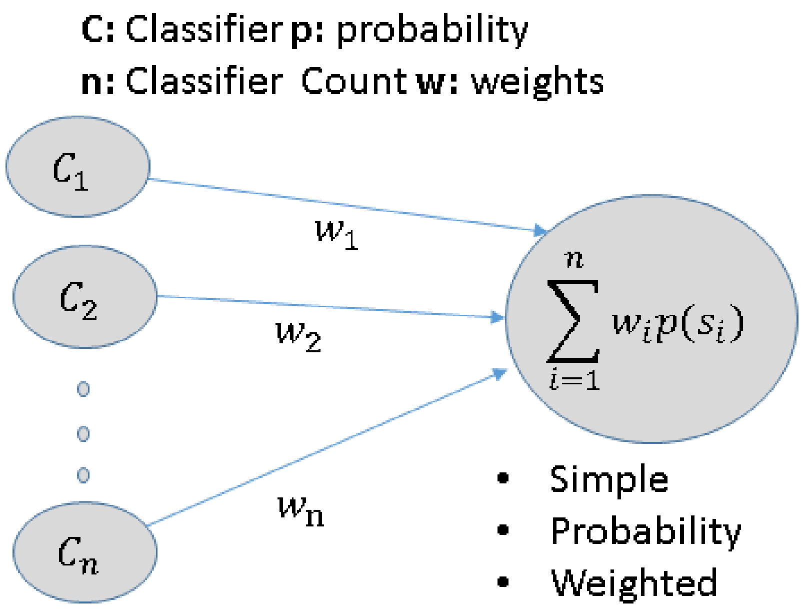

C: Classifier p: probability
n: Classifier Count w: weights
C_1
C_2
C_n
w_1
w_2
w_n
$\sum_{i=1}^{n} w_i p(s_i)$
Simple
Probability
Weighted

Figure 3.2 Multi classifier ensemble

3.2.5. Genetic algorithms

Inspired by evolution theory, genetic algorithms (GA) codes the optimization problem using a genome. Binary genome, as the first proposed genome, is widely used. All genomes that are used in problem called population. While genomes change from generation to generation, different areas in solution space are searched. Genomes are evolved using crossover and mutation operations. Using crossover operation two genome in population transfer part of their genomes to next generation. For selection of genomes in crossover operation, different methods are used. In this study, roulette wheel method is used since it increases chances for successful genomes. Mutation operation is change of small part of genome with a probability. While crossover allows that successful genomes will live in next generation, mutation operation enable search of different areas in solution space; thus, GA will not converge to a local minimum [71]. If only crossover operation is used, most successful genomes in population will be lost; therefore, elitism principle is used. Elitism transfers some of the most successful genomes to next generation without change.

3.2.6. Proposed method

Proposed method is called as GA-FS-WF (Genetic Algorithms based Feature Selection and Weights Finding). GA-FS-WF's general flow chart is given in Figure 3.3. To select features and to find weights, a two part genome is defined

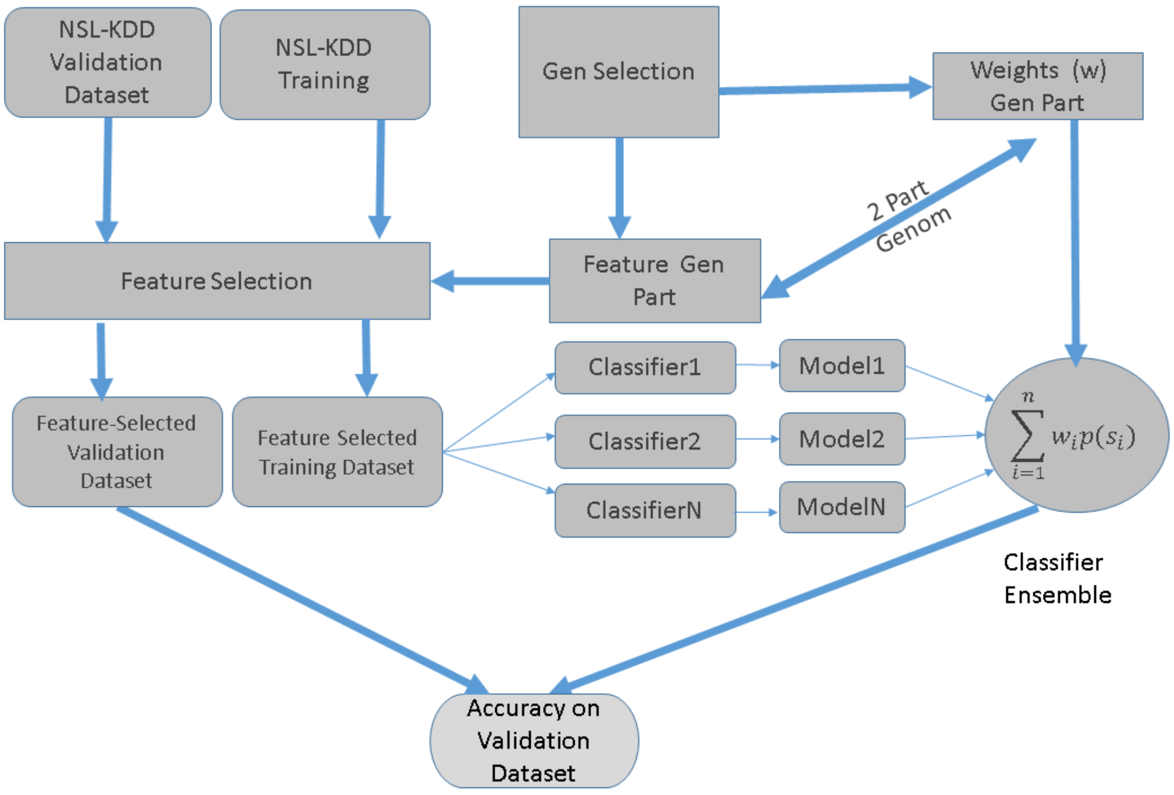


Figure 3.3 Proposed method flowchart

in Figure 3.3.

Most important part of genetic algorithms is coding of problem as a genome. In this study, binary two part genome (FEATURES + WEIGHTS) is used, see Figure 3.4.

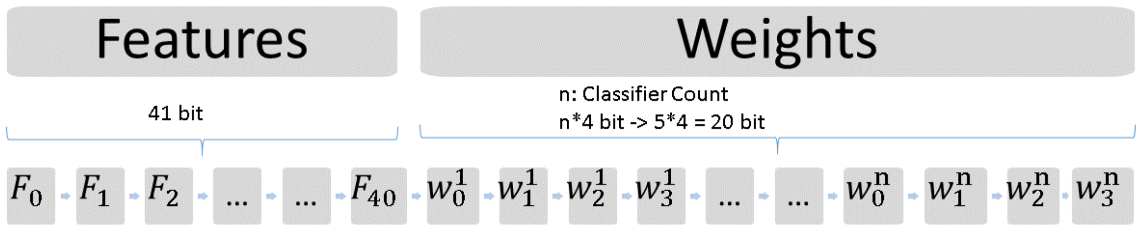

Features
Weights
41 bit
n: Classifier Count
n*4 bit -> 5*4 = 20 bit
F_0 F_1 F_2 ... ... F_{40} w_0^1 w_1^1 w_2^1 w_3^1 ... ... w_0^n w_1^n w_2^n w_3^n

Figure 3.4 Binary genome

Features genome part consists of 41 bit that comes from NSL-KDD data set's 41 features. If a bit is 0 in feature genome part, corresponding feature is not selected; but, if that bit is 1, then that feature is selected. For example genome string 1100.., in this 41 bit length genome string, 0. feature (duration) and 1. (protocol_type) are selected.

Weights genome part size depends on classifier ensemble size. Weight of each classifier is coded as four bits. This 4-bit value affects decision of ensemble. For example 1010 bit string will multiply probability vote of classifier with 10. If 5 classifier is combined, then weights genome part will be $5x4 = 20$ bit. Then, in 5 classifier ensemble, genome size is 61 bit total. For features, 41 bit are used. For weights 20 bit, 5 classifiers x 4 bit, are used.

Table 3.5 **Genetic algorithms hyper parameters**

Name	**Value**
Genome Type	Binary
Genome Length	53-69 bit
Population Size	80
Crossover rate	0.9
Mutation Rate	0.02
Generation Number	100
Elitism	1
Selection	Roulette Wheel

Proposed method is GA-based optimization problem. Ensemble classifier's accuracy value on validation dataset, Equation 3.1, has been chosen as fitness function. Genome values were given as input to fitness functions. These genome values were divided into two parts; features and weights. Using feature genome part, features were selected from training and testing dataset. After this step, using training dataset single classifiers were trained. Trained classifiers affect output result according to the weights genome part. At last, ensemble classifier applied to validation dataset. Accuracy value (Equation 3.1) of ensemble classifier on validation dataset has been returned as fitness score. This flow has been shown in Figure 3.3. In addition, pseudo code for full system has been given in listing 3.1. Following programs have been used in the experiments: python, scikit-learn, matplotlib, and pyevolve. Other GA hyper parameters has been given in Table 3.5.

Pseudo Code 3.1 Proposed Method (GA-FS-WF) Pseudocode

```
THRESHOLD_VALUE <– 0.90
MAX_ITERATION <– 100

population <– create population randomly
training_ds_all_features <– read training dataset features
training_class <– read training dataset class
validation_ds_all_features <– read validation dataset features
validation_class <– read validation dataset class
classifiers <– create classifier list
best_gen <– null

for i=1 to MAX_ITERATION
   for j=0 to length(population)
     gen = population(j)
     ( gen_features, gen_weight) = partition(gen)
     training_ds = feature_selection(gen_features
                  ,training_ds_all_features)
     validation_ds = feature_selection(gen_features
                  ,validation_ds_all_features)
     classifiers = train_classifiers(training_ds)
     combined = Combination(classifiers ,gen_weight)
     accuracy_validation = combined.score(validation_ds ,validation_class)
     gen.skor = accuracy_validation
     if (accuracy_validation > THRESHOLD_VALUE)
       best_gen = gen
       goto end
     end if
   end for
   sorted_population <– Sort Population according to score.
   new_population <– select gen with elitizm (sorted_population)
   best_gen <– find_best_gen(sorted_population)
   new_population <– crossover and mutation (sorted_population)
   population <– new_population
end for
end:
   return best_gen
```

3.3. Simulation study and results

GA-FS-WF was tested on the NSL-KDD dataset. Following combination methods were tested: Simple voting, probability voting, and weighted voting. Ensemble sizes of 3–7 classifiers were used. For every ensemble size, 16 experiments, totally 80 experiments were conducted. Change of fitness function from generation to generation has been given in Figure 3.5. As can be seen in this figure, fitness value increases regularly from generation to generation.

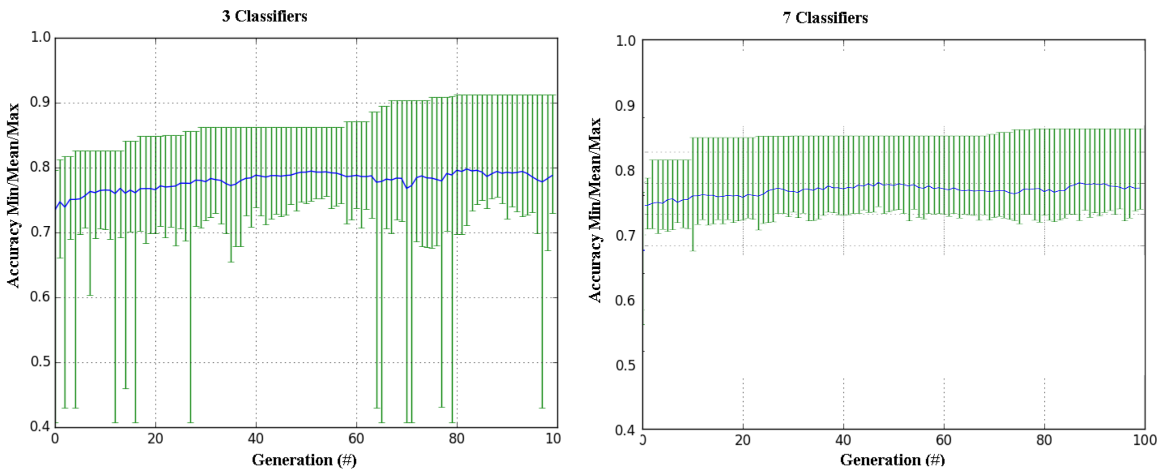


Figure 3.5 Fitness functions of example GA runs

Results of the experiments has been given Figure 3.6. Simple voting, probability voting, and weighted voting combination method's performance has been shown in Figure 3.6. Classification accuracy decreases with the increase of ensemble size in Figure 3.6. This result is independent of combination method used in ensemble. According to results, 3-4 classifier ensemble has best classification accuracy. Weighted combination method has better results compared to other combination methods (Simple voting and probability voting) when classifier ensemble size is low (3-4 classifiers). Best result among 3-4 classifiers is 3 classifier ensemble.

Generally, with the increase of classifier size in ensemble, accuracy results decrease. The reason of this phenomena is that diverse classifier should be used in classifier ensembles [61, 63, 72]. Classifier diversity has no accepted definition [72], nonetheless it affects ensemble accuracy. When ensemble size increases, classification diversity reduces; thus, classification accuracy decreases [70, 72].

Table 3.6 **Classifier ensemble performance values**

	Classifier Count														
	Maximum, Mean, Standard Deviation and Minimum Values														
Ensemble		**3**			**4**			**5**			**6**			**7**	
	max	mean(std)	min	max	mean (std)	min	max	mean(std)	min	max	mean(std)	min	max	mean(std)	min
Single Vote	0.90	0.84 (0.034)	0.77	0.85	0.81 (0.015)	0.78	0.83	**0.80 (0.018)**	0.76	0.81	0.78 (0.022)	0.74	0.83	**0.79 (0.021)**	0.76
Probability Vote	0.91	0.82 (0.038)	0.77	0.90	0.86 (0.025)	0.82	0.84	0.78 (0.021)	0.76	0.86	0.79 (0.034)	0.75	0.83	0.78 (0.020)	0.75
Weighted Vote	0.90	**0.89 (0.011)**	0.87	0.91	**0.88 (0.027)**	0.80	0.91	0.76 (0.065)	0.71	0.90	**0.80 (0.118)**	0.45	0.81	0.73 (0.053)	0.57

Maximum, mean, standard deviation, and minimum classification accuracy values has been given in Table 3.6. As can be seen in Table 3.6, 3-4 classifier ensembles have best accuracy values. To show effects of proposed GA-based feature selection,

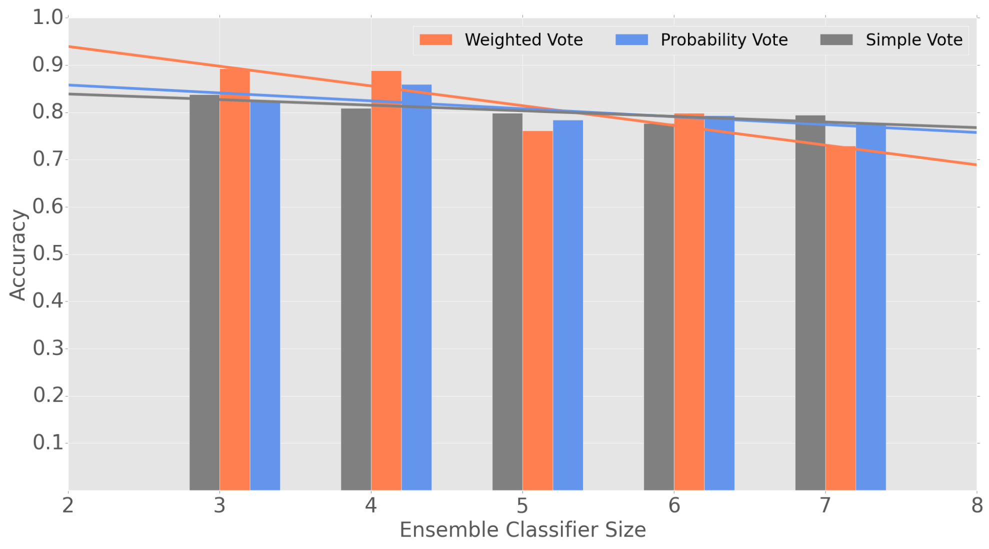


Figure 3.6 Accuracy of ensembles according to classifier size and regression lines. As ensemble classifier size increases accuracy values decreases. Fastest to decrease is weighted ensemble

classifier ensemble with all features and classifier ensemble with selected features have been compared. These results are given in Table 3.7.

Table 3.7 **Effect of feature selection to classification accuracy**

Ensemble Type	41 Features	Feature Selection with GA
Simple Vote	0.76	0.89
Probability Vote	0.79	0.91
Weighted Vote	0.79	0.91

In the feature selection step, 19 average features, 14 minimum, and 28 maximum features have been selected among 41 features in 80 experiments. Feature **service** has been selected in all experiments (80/80); thus, it is the most informative feature in this dataset. After that, features (num_compromised, num_root) have been selected in 59/80 experiments. Selection count for other features can be seen in Table 3.12. According to the results, GA-based feature selection improves classification accuracy compared to full feature classification.

Testing dataset accuracy of single classifiers has been shown in Table 3.8. Feature selection step improves accuracy of single and ensemble classifiers. The most successful classifiers are decision tree, gradient boosting and k-nearest neighbor (5 neighbors are chosen empirically). Classification ensemble has better accuracy (0.9) compared to single classifiers, Table 3.8 and Table 3.7.

Table 3.8 **Single classifiers accuracy values: effect of GA feature selection**. All features vs GA feature selection

	Single Classifiers Mean (Standard Deviation)						
Classifiers Feature Count	AdaBoost	Decision Tree	Gradient Boosting	KNN	Logistic Regression	Naive Bayes	Random Forests
All (41)	0.77(0)	0.79(0.02)	**0.83(0)**	0.77(0)	0.69(0)	0.45(0)	0.77(0.01)
GA (19)	0.79(0.02)	**0.83(0.04)**	**0.83(0.03)**	**0.83(0.02)**	0.64(0.08)	0.46(0.05)	0.79(0.02)

GA-based feature selection reduces training and testing time, Table 3.9. Training time reduction was about 51% and testing time reduction was about 33-39%.

Table 3.9 **Effect of feature selection to training and testing time**

	Training Time (Sec)			Testing Time (Sec)		
	All Features	Feature Selected	Decrease %	All Features	Feature Selected	Decrease %
Simple Vote	30.67	14.88	51.47	2.36	1.58	33.15
Probability Vote	30.82	14.83	51.87	2.00	1.23	38.35
Weighted	30.94	14.82	**52.09**	2.00	1.22	**39.18**

To show statistical significance of results, classifier ensemble results have been tested using statistical tests: ANOVA and t-test, Table 3.10. ANOVA statistical test shows statistically significant differences between groups. Small p-values (* < 0.05) show that differences between group members are statistically significant. Smaller p-values (** < 0.001) show that statistical significance is more stronger. ANOVA test results in Table 3.10 show that differences of accuracy values for 3,4 and 7 classifier ensemble are statistically significant. In other classifier ensemble sizes (5 and 6), this statistical difference becomes nil or very low for both ANOVA and t-test results. T-test results in Table 3.10 show that weighted combination accuracy values are statistically different from simple voting and probabilistic voting. Then, for 3-4 classifier ensemble, high accuracy values are statistically significant. For 7 classifier ensemble, weighted voting has low accuracy values, and this result is statistically significant. In summary, 3-4 classifier ensemble with weighted voting should be used.

3.3.1. Comparison with literature

Results of the GA-FS-WF is compared with 10 different study from the literature, Table 3.11 Most of the studies that use NSL-KDD use training dataset as test dataset in their results since NSL-KDD training dataset and test dataset are statistically different. Using similar dataset for both training and testing purposes gives optimistic

Table 3.10 ANOVA and t-test statistical tests results

	p-value * <0.05 significant ** <0.001 more significant			
Classifier Size	ANOVA	t-test Weighted Simple Vote	t-test Weighted Probability Vote	t-test Simple Probability Vote
3	**	**	**	0.2015
4	**	**	**	**
5	0.042*	0.036*	0.16	0.07
6	0.626	0.42	0.76	0.15
7	**	**	0.002*	0.011*

Table 3.11 **Comparison of proposed method with the literature**. All of the results are on the NSL-KDD dataset.

Study	Year	Feature Selection	Test Database	Detection Rate
Rastegari et al. [44]	2015	Yes	Training	0.7800
Kang and Kim [73]	2016	Yes	Training	0.9693
Pereira et al. [74]	2012	Yes	Training	0.9661
Seresht and Azmi [75]	2014	No	Training	0.8831
Farid et al. [76]	2014	No	Training	0.8344
Singh et al. [77]	2015	Yes	Training	0.9867
Bhattacharya and Selvakumar [78]	2015	Yes	Testing	0.8314
Mohammadi et al. [79]	2012	Yes	Testing	0.8014
Liu et al. [80]	2016	Yes	Testing	0.7460
la Hoz et al. [81]	2015	Yes	Testing	0.8800
GA-FS-WF (3 Classifier Ensemble)	2017	Yes	Testing	0.9088

results. Therefore, in Table 3.11 those results that use NSL-KDD Training dataset as test dataset have better results. For example, results of Kang and Kim [73] (0.9693) and Pereira et al. [74] (0.9661) are on the training dataset. GA-FS-WF has better results compared with other results that use NSL-KDD test dataset.

3.4. Results and chapter discussion

In this chapter, feature selection and weights finding in classifier ensemble problem have been investigated using genetic algorithms in intrusion detection and classification applications. Proposed method has been tested on the NSL-KDD dataset using three different ensemble combination methods. The results can be summarized as follows:

1. Multiple classifier ensemble is more successful compared to single classifiers.

Table 3.12 Features and how many times they are selected in experiments

Feature Name	Selection Count	Feature Name	Selection Count
service	80	num_shells	38
num_compromised	59	num_file_creations	38
num_root	59	srv_rerror_rate	37
src_bytes	53	is_guest_login	37
duration	52	srv_serror_rate	35
dst_host_rerror_rate	51	dst_host_same_src_port_rate	33
num_access_files	50	diff_srv_rate	32
is_host_login	50	num_failed_logins	31
dst_host_srv_rerror_rate	50	dst_host_srv_diff_host_rate	30
su_attempted	48	dst_host_serror_rate	29
rerror_rate	46	serror_rate	26
urgent	45	dst_host_srv_serror_rate	25
num_outbound_cmds	45	same_srv_rate	23
dst_bytes	44	dst_host_same_srv_rate	22
srv_count	43	wrong_fragment	22
hot	43	dst_host_diff_srv_rate	22
srv_diff_host_rate	43	flag	20
dst_host_count	43	logged_in	18
root_shell	40	count	6
land	40	dst_host_srv_count	0
protocol_type	39		

2. When effect of classifier size in multiple classifier ensemble has been investigated, it is found that 3-4 classifier ensemble is more successful on the NSL-KDD dataset.

3. GA-based simultaneous feature selection and ensemble weights finding improve classification accuracy and reduce training and testing time.

4. Generally, weighted classifier combination is better than simple voting and probability voting, and these results are statistically significant.

Proposed method, GA-FS-WF, can be applied to other large datasets. Also using GA or another optimization algorithm, GA-FS-WF should also be applicable to other domains.

4. SPARSITY-DRIVEN WEIGHTED ENSEMBLE CLASSIFIER

In this chapter, a novel sparsity-driven weighted ensemble classifier (SDWEC) that improves classification accuracy and minimizes the number of classifiers is proposed. Using pre-trained classifiers, an ensemble in which base classifiers votes according to assigned weights is formed. These assigned weights directly affect classifier accuracy. In the proposed method, ensemble weights finding problem is modeled as a cost function with the following terms: (a) a data fidelity term aiming to decrease misclassification rate, (b) a sparsity term aiming to decrease the number of classifiers, and (c) a non-negativity constraint on the weights of the classifiers. As the proposed cost function is non-convex thus hard to solve, convex relaxation techniques and novel approximations are employed to obtain a numerically efficient solution. Sparsity term of cost function allows trade-off between accuracy and testing time when needed. The efficiency of SDWEC was tested on 11 datasets and compared with the state-of-the art classifier ensemble methods. The results show that SDWEC provides better or similar accuracy levels using fewer classifiers and reduces testing time for ensemble.

4.1. Chapter Introduction

The accuracy of classification can be improved by using more than one classifier. This process is known by different names in different domains such as classifier fusion, classifier ensemble, classifier combination, mixture of experts, committees of neural networks, or voting pool of classifiers, and others [61].

Ensembles can be categorized as weak or strong depending on the used classifier type [82]. The weak classifiers use machine learning algorithms with fast training times and lower classification accuracy. Due to fast training times, weak classifier ensembles contain high number of classifiers, such as 50–200 classifiers. On the other hand, strong classifiers have slow training times and high generalization accuracy individually. Due to slow training times, strong classifier ensembles contain as low as 3–7 classifiers.

Although using more classifiers increases generalization performance of ensemble classifier, this degrades after a while. To put it in another way, similar classifiers do not contribute to overall accuracy very much. This deficiency can be removed by increasing the classifier diversity [61, 63, 72]. Therefore, finding new diversity measurements [83] and improving existing ones [72] are an ongoing research effort in ensemble studies.

Research in the ensembles can be categorized into two groups according to their construction methods: (a) Combining pre-trained classifiers. (b) Constructing classifiers and ensemble together.

Methods in the first group (a) are the easiest to understand and the mainly used methods to create ensembles. The classifiers are trained using training set and combined in an ensemble. The simplest method to ensemble classifiers is majority (plurality) voting. In the majority voting method, every classifier in an ensemble gets a single vote for result. The output is the most voted result. A well-known approach that uses majority voting in its decision stage is Bootstrap aggregating algorithm (Bagging) [84]. Bagging trains weak classifiers from same dataset using uniform sampling with replacement, then classifiers are combined using simple majority voting [70]. Instead of using a single vote for every classifier, weighted voting might be used [70]. Standard Weighted majority voting (WMV) algorithm [70] uses accuracy of individual classifiers for finding weights. Classifiers that have better accuracies in training step get higher weights for their votes, and become more effective in voting. Kuncheva and Rodriguez [85] proposed a probabilistic framework for classifier ensembles. This framework shows relationships between four combiners: majority voting, weighted voting, recall voting, and naive bayes voting. According to the experiments of Kuncheva and Rodríguez [85] on 73 benchmark datasets, there is no definite best combiner among those four. These results conform to “no free lunch theorem” [86, 87]. No universal classifier exists that is suitable for every problem. Numerous other methods has been proposed for finding weights to combine pre-trained classifiers, Table 4.1. Methods in Table 4.1 are also summarized in Section 4.1.1. Similar to approaches in Table 4.1, main focus of this study is to present a new approach for finding weights in an ensemble that uses pre-trained classifiers using convex optimization techniques.

In the second categorization (b), ensemble construction and classifier construction affect each other. Adaboost [88] is a well known example for this categorization that trains weak classifiers iteratively and adds them to ensemble. Different from bagging, subset creation is not randomized in boosting. At each iteration, subsets are obtained using results of previous iterations. That is miss classified data in previous subsets are more likely included. In ensemble, standard weighted majority voting is used.

Gurram and Kwon [89] used similar approach to classify remote sensing images. Randomly selected features were used to train weak SVM classifiers. Optimization process of training and combination of these classifiers were done together. Lee et al. [90] combined neural network weak classifiers in their ensemble. Genetic algorithms were used for finding weights for neural network neurons and increase diversity among neural networks. Then, these diverse neural networks were combined using negative correlation rule. Neural networks were trained and combined in one step. Tian and Fend [91] proposed an approach that combines feature sub-selection and ensemble optimization. They proposed three-term cost function: a classification accuracy term, a diversity term and a feature size term. They solved this ensemble cost function using population based heuristics optimization. Zhang et al. [92] used Kernel sparse representation based classifiers for ensemble in face recognition domain. Features were projected to higher dimensions using kernels, then sparse representation of these features were found using optimization techniques. Similarly, Kim et al. [93] proposed ensemble approach for biological data. Their approach were similar to boosting but they selected sparse features in their weak classifiers.

Table 4.1 Ensemble weights finding studies that use pre-trained classifiers

Study	Year	Classifiers	Method	Size	Sparse	Cost Function	Regularizer	Notes
Sylvester and Chawla [64]	2006	12 Different Classifiers	Genetic Algorithms	120	No	No Information	No Information	
Li and Zhou [94]	2009	Decision Tree	Quadratic Programming	100	Yes	Hinge Loss	L_1	
Kim et al. [95]	2011	Decision Tree	Matrix Decomposition	64	No	Indicator Loss	No Regularization	
Mao et al. [96]	2011	Decision Tree,SVM	Matching Pursuit	100	Yes	Sign Loss	No Regularization	
Zhang and Zhou [97]	2011	K-Nearest Neighbor	Linear Programming	100	Yes	Hinge Loss	L_1	
Goldberg and Eckstein [98]	2012	NI	Linear Programming	NI	Yes	Indicator Loss	L_0	[a]
Santos et al. [99]	2012	SVM,MLP	Genetic Algorithms	6	No	No Cost Function	No Regularization	
Yin et al. [100]	2012	Neural Networks	Genetic Algorithms	100	Yes	Square Loss	L_1	[b]
Meng and Kwok [101]	2013	Decision Tree,SVM,KNN	Domain Heuristic	3	No	No Cost Function	No Regularization	
Tinoco et al. [102]	2013	SVM,MLP	Linear Programming	6	Yes	Hinge Loss	L_1	[d]
Hautamaki et al. [103]	2013	Logistic Regression	Nelder–Mead	12	Yes	cross-entropy [104]	$L_1, L_2, L_1 + L_2$	[c]
Şen and Erdoğan [105]	2013	13 Different Classifiers	Convex Opt.	130	Yes	Hinge Loss	L_1 , Group Sparsity	
Mao et al. [106]	2013	Decision Tree	Singular Value Decomposition	10	No	Absolute Loss	No Regularization	
Yin et al. [107]	2014	Neural Networks	Genetic Algorithms	100	Yes	Square Loss	L_1	[e]
Yin et al.[108]	2014	Neural Networks	Quadratic Programming	100	Yes	Square Loss	L_1	[f]
Zhang et al. [63]	2014	5 Different Classifiers	Differential Evolution	5	No	No Cost Function	No Regularization	
Mao et al. [109]	2015	Decision Tree	Quadratic Form	200	No	Square Loss	L_1	

[a] No experimental results.
[b] Diversity Term Yule's Q Statistic is used
[c] Improved version of [99]
[d] 3 regularizers are compared
[e] Journal version of [100]
[f] Convex Quadratic model of [107] and [100]

4.1.1. Related works: ensembles that combine pre-trained classifiers

Focus of this study is to combine pre-trained classifiers so that combined accuracy of the ensemble is better than individual classifiers. This study aims to accomplish this objective in a sparse manner so that not all of the classifiers are used in ensemble. Although some of the other sparse approaches [89, 92, 93, 110] are mentioned before, in this section, only ensemble classifiers that proposes methods to find weights for base classifiers are investigated.

Sylvester and Chawla [64] proposed differential evolution to find suitable weights for ensemble base classifiers. Similar to most heuristic solution techniques, they did not explicitly define cost function, but use classification accuracy for fitness function. ID3 decision trees, J48 decision trees (C4.5), JRIP rule learner (Ripper), Naive Bayes, NBTree (Naive Bayes trees), One Rule, logistic model trees, logistic regression, decision stumps, multi-layer perceptron (MLP), SMO (support vector machine), and 1BK (k-nearest neighbor) classifiers from Weka toolbox [111] were used in the experiments.

Li and Zhou [94] modeled ensemble weights finding problem using cost function that consists of hinge loss and L_1 regularization. This cost function were minimized using Quadratic programming. Decision tree weak classifiers and UCI datasets were used for experiments. A semi-supervised version was also suggested.

Zhang and Zhou [97] formulated weights finding problem using three different cost functions: LP1, LP2, LP3. LP1 uses a cost function that consists of Hinge loss only. LP2 uses a cost function that consists of Hinge loss and L_1 regularization. LP3 allows weights to be negative. These cost functions were minimized using linear programming. They used K-Nearest neighbor algorithm as base classifiers and UCI datasets in their experiments.

Kim et al. [95] proposed an approach similar to boosting. They considered two weight vectors, one for classifier and one for instances. Hard to classify instances get more weight and correspondingly they affect weight vector more. Different from boosting, their approach works with pre-trained classifiers. Weights for ensemble was found using matrix decomposition and an iterative algorithm. Decision weak classifiers and UCI datasets were used for experiments.

Mao et al. [96] proposed matching pursuit algorithm to find weights for ensemble base classifiers. Since matching pursuit is a sparse approximation algorithm [112], their approach include sparsity. Decision Tree and SVM weak classifiers and UCI datasets were used for experiments.

Goldberg and Eckstein[98] modeled weights finding problem with indicator loss function and L_0 regularization function. According to Goldberg and Eckstein[98], this problem is NP-hard in special cases. They gave different relaxation strategies to solve this problem and gave their relaxation bounds. Different from other studies, this study was purely theoretical.

Santos et al. [99] combined multi-layer perceptrons (MLP) and SVM algorithms to classify remote sensing images. They did not give any explicit cost function but used genetic algorithms for finding weights. An improved version of their studies [102] modelled weights finding problem using hinge loss and L_1 regularization. This cost function were minimized using linear programming. In both versions, remote sensing images were classified using ensemble of MLP and SVM classifiers.

Mong and Kwok [101] combined Decision Tree(J48), K-Nearest Neighbor and SVM classifiers. They suggest using following domain heuristic for weights of classifiers: "...weighted ranking (precision of false alarm $>$ recall of false alarm $>$ classification accuracy) is an appropriate and correct way to decide the weight values with high confidence in ensemble selection..."[101]

Hautamaki et al. [103] investigated using sparse ensemble in speaker verification domain. Ensemble weights finding problem were modeled using cross-entropy loss function and three different regularization functions, L_1,L_2, and $L_1 + L_2$. These cost functions were minimized using Nelder–Mead method. Logistic regression classifiers were used in experiments.

Yin et al. [100] modeled ensemble weights finding problem with a cost function that consists of the terms a square loss, L_1 regularization and a diversity based-on Yule's Q statistics. They used neural network classifiers on 6 UCI datasets in their experiments. In their first study [100], the proposed cost function were minimized using genetic algorithms. In their second study [107], the Pascal 2008 webspam dataset were added to their experiments. Finally, convex optimization techniques [108] were used to minimize the same cost function.

Sen and Erdogan [105] modeled ensemble weights finding problem using a cost function that consists Hinge loss and two different regularization functions, L_1 and group sparsity. This cost function were minimized using convex optimization techniques. In their experiments, 13 different classifiers were compared on 12 UCI datasets and 3 other datasets using CVX Toolbox [113, 114].

Zhang et al. [63] proposed Differential Evolution for finding suitable weights for ensemble base classifiers. Similar to most heuristic solution techniques, they did not explicitly define cost function, but use classification accuracy for fitness function. Decision Tree (J4.8), Naive Bayes, Bayes Net, k-Nearest Neighbor, and ZeroR classifiers from Weka toolbox [111] were used in the experiments.

Mao et al. [106] modeled ensemble weights finding problem using a cost function that consists of absolute loss only. Proposed cost function was minimized using 0–1 matrix decomposition. In a later study [109], Mao et al. proposed a cost function that consists of square loss and L_1 regularization function. This cost function was minimized using quadratic form approximation. Both studies used decision tree weak classifiers and UCI datasets in experiments.

As can be seen from Table 4.1, numerous approaches exist for finding weights in ensemble classification. Inspired from studies of [97, 106, 109, 115], sparsity-driven weighted ensemble classifier (SDWEC) has been proposed. Proposed cost function consists of the following terms: (1) a data fidelity term with sign function aiming to decrease misclassification rate, (2) L_1-norm sparsity term aiming to decrease the number of classifiers, and (3) a non-negativity constraint on the weights of the classifiers. Cost function proposed in SDWEC is hard to solve since it is non-convex and non-differentiable; thus, (a) the sign operation is convex relaxed using a novel approximation, (b) the non-differentiable L_1-norm sparsity term and the non-negativity constraint are approximated using *log-sum-exp* and Taylor series. Contributions of SDWEC can be summarized as follows:

1. A new cost function is proposed for ensemble weights finding problem.
2. This cost function is minimized using novel convex relaxation and approximation techniques for sign function and absolute value function.
3. SDWEC improves classification accuracy, while minimizing the number of classifiers used in the ensemble.
4. According to sparsity level of SDWEC, number of classifiers used in the ensemble decreases; thus, the testing time for whole ensemble decreases.
5. The sparsity level of SDWEC allows trade-off between testing accuracy and testing time when needed.
6. Computational Complexity of SDWEC is analyzed theoretically and experimentally, which is linear in number of data rows, number of classifiers and number of algorithm iterations.

4.2. Sparsity-driven weighted ensemble classifier

An ensemble consists of l number of classifiers. Classifiers are trained using training dataset. We aim to increase ensemble accuracy on test dataset by finding suitable weights for classifiers using validation dataset. Ensemble weights finding problem is modeled with the following matrix equation.

$$sgn(\underbrace{\begin{bmatrix} -1 & -1 & \dots & +1 \\ +1 & -1 & \dots & -1 \\ \vdots & \vdots & \vdots & \vdots \\ -1 & \dots & \dots & +1 \\ +1 & \dots & \dots & -1 \end{bmatrix}}_{H_{mxl}} \underbrace{\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{l-1} \\ w_l \end{bmatrix}}_{w_{lx1}}) \approx \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{m-1} \\ y_m \end{bmatrix}}_{y_{mx1}}$$

In this matrix equation, classifiers predictions are weighted so that obtained prediction for each data row becomes approximately equal to expected results. Matrix H consists of l classifier predictions for m data rows that are drawn from validation dataset. Our aim is to find suitable weights for w in a sparse manner while preserving condition of $sgn(Hw) \approx y$ (sign function). For this model, the following cost function is proposed:

$$J(w) = \frac{\lambda}{m}\sum_{s=1}^{m}(sgn(\mathbf{H_s}w) - y_s)^2 + \frac{1}{l}||w||_1^1 \quad \text{subject to } w \geq 0 \tag{4.1}$$

m	number of samples
w	classifier weights
H	classifiers results $\{-1, 1\}^{mxl}$
l	number of individual classifiers
λ	data fidelity coefficient
y	true labels $\{-1, 1\}^{mx1}$
H_s	s_{th} row vector of matrix H

In equation 4.1, the first term acts as a data fidelity term and minimizes the difference between true labels and ensemble predictions. Base classifiers of ensemble give binary predictions (-1 or 1) and these predictions are multiplied with weights through sign function which leads to $\{-1, 0, 1\}$ as an ensemble result. To make this term independent from data size, it is divided to m (number of data rows). The second term is sparsity term [116] that forces weights to be sparse [115]; therefore, minimum number of classifiers are utilized. In sparsity term, any L_p-norm ($0 \leq p \leq 1$) can be used. Weights become more sparse as p gets closer to 0. However, when ($0 \leq$

$p < 1$), sparsity term becomes non-convex and thus the problem becomes harder to solve. When p is 0 (L_0-norm) then problem becomes NP-hard [117]. Here, L_1-norm is used as a convex relaxation of L_p-norm [116, 118]. Similar to the data fidelity term, this term is also normalized with division by l (number of individual classifiers). The third term is used as a non-negativity constraint. Since base binary classifiers use values in interval $(-1, 1)$ for class labels, negative weights change sign of prediction; thus they change class label of prediction. To avoid this problem, the constraint term is added to force weights to be non-negative. Using the method of Lagrange multipliers and the definition of $|x| = max(-x, x)$, cost function is transformed into equation 4.2.

$$\begin{aligned} J(w) = &\frac{\lambda}{m}\sum_{s=1}^{m}(sgn(\mathbf{H_s}w) - y_s)^2 \\ &+ \frac{1}{l}\sum_{r=1}^{l} max(-w_r, w_r) \\ &+ \frac{\beta}{l}\sum_{r=1}^{l} max(-w_r, 0) \end{aligned} \tag{4.2}$$

In equation 4.2, the constraint $w \geq 0$ is better satisfied as β becomes larger. Equation 4.2 is non-convex function, since *sgn* function creates jumps on cost function surface. In addition, *max* function is non-differentiable. Functions *max* and *sgn* in Equation 4.2 are hard to minimize. Therefore, we propose a novel convex relaxation for *sgn* as given in equation 4.3. Figure 4.1 shows approximation of sign function using Equation 4.3.

$$sgn(\mathbf{H_s}w) \approx \frac{\mathbf{H_s}w}{|\mathbf{H_s}\hat{w}| + \epsilon} = S_s\mathbf{H_s}w \tag{4.3}$$

where

$$S_s = (|\mathbf{H_s}\hat{w}| + \epsilon)^{-1} \tag{4.4}$$

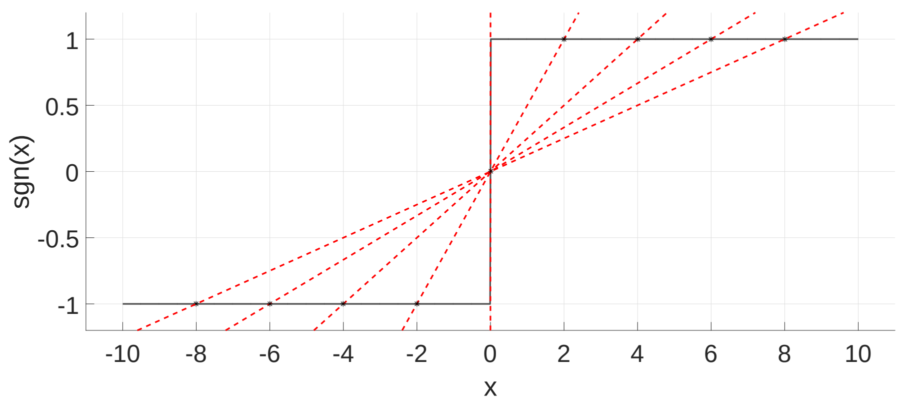


Figure 4.1 Sign function approximation using equation 4.3. Dotted Lines are approximations using Equation 4.3 at various points.

In this equation, ϵ is a small positive constant. We also introduce a new constant $\hat{w}$ as a proxy for w. Therefore, $S_s = (|\mathbf{H_s}\hat{w}| + \epsilon)^{-1}$ is also a constant. However, this *sgn* approximation is only accurate around introduced *constant* $\hat{w}$. Therefore, the approximated cost function needs to be solved iteratively. Additionally, *max* function is approximated with *log-sum-exp* [119] as follows:

$$max(-w_r, w_r) \approx \frac{1}{\gamma} log(e^{-\gamma w_r} + e^{\gamma w_r}) \tag{4.5}$$

Accuracy of *log-sum-exp* approximation becomes better as γ, a positive constant, increases. In double-precision floating-point format [120], values up to 10^{308} in magnitude can be represented. This means that $\gamma|w_r|$ should be less than 710 where $exp(709) \approx 10^{308}$, otherwise exponential function will produce infinity (∞). At $w_r = 0$, there is no danger of numerical overflow in exponential terms of a *log-sum-exp* approximation; thus, large γ values can be used. But as $|w_r|$ gets larger, there is a danger of numerical overflow in exponential terms of *log-sum-exp* approximation, since $e^{\gamma|w_r|}$ may be out of double precision floating point upper limit.

To remedy this numerical overflow problem, a novel adaptive γ approximation is proposed, where γ_r is adaptive form of γ and defined as $\gamma_r = \gamma(|\hat{w}_r| + \epsilon)^{-1}$. The accuracy of approximation can be improved by decreasing ϵ or increasing γ. Figure 4.2 shows proposed adaptive γ and resulting approximations for two different set of values $(\gamma = 10, \epsilon = 0.1)$ and $(\gamma = 10, \epsilon = 1)$.)

Validity of the approximation can be checked by taking the limits at $-\infty, 0$, and $+\infty$ with respect to w_r. These limits are $-x, \frac{\epsilon\sqrt{2}}{\lambda_r}$, and x when w_r goes to $-\infty, 0$, and $+\infty$. As $|x|$ gets larger, dependency to γ decreases; thus, proposed adaptive γ approximation is less prone to numerical overflow compared to standard *log-sum-exp* approximation.

Application of adaptive γ approximation leads to the following equations:

$$max(-w_r, w_r) \approx \frac{1}{\gamma_r} log(e^{-\gamma_r w_r} + e^{\gamma_r w_r}) \tag{4.6}$$

$$\beta max(-w_r, 0) \approx \frac{\beta}{\gamma_r} log(e^{-\gamma_r w_r} + 1) = P(w_r) \tag{4.7}$$

This approximation leads to the cost function shown in equation 4.8, where n is the number of iterations.

$$\begin{aligned} J^{(n)}(w) = & \frac{\lambda}{m} \sum_{s=1}^{m} (S_s \mathbf{H_s} w - y_s)^2 \\ & + \frac{1}{l} \sum_{r=1}^{l} \frac{1}{\gamma_r} log(e^{-\gamma_r w_r} + e^{\gamma_r w_r}) \\ & + \frac{1}{l} \sum_{r=1}^{l} \frac{\beta}{\gamma_r} log(e^{-\gamma_r w_r} + 1) \end{aligned} \tag{4.8}$$

In order to achieve a second-order accuracy and to obtain a linear solution, after taking the derivative of the cost function, equation 4.8 is expanded as a second-order Taylor series centered on $\hat{w}_r$, leading to equation 4.9.

$$\begin{aligned} J^{(n)}(w) = & \frac{\lambda}{m} \sum_{s=1}^{m} (S_s \mathbf{H_s} w - y_s)^2 \\ & + \frac{1}{l} \sum_{r=1}^{l} (A_r + B_r w_r + C_r w_r^2) \end{aligned} \tag{4.9}$$

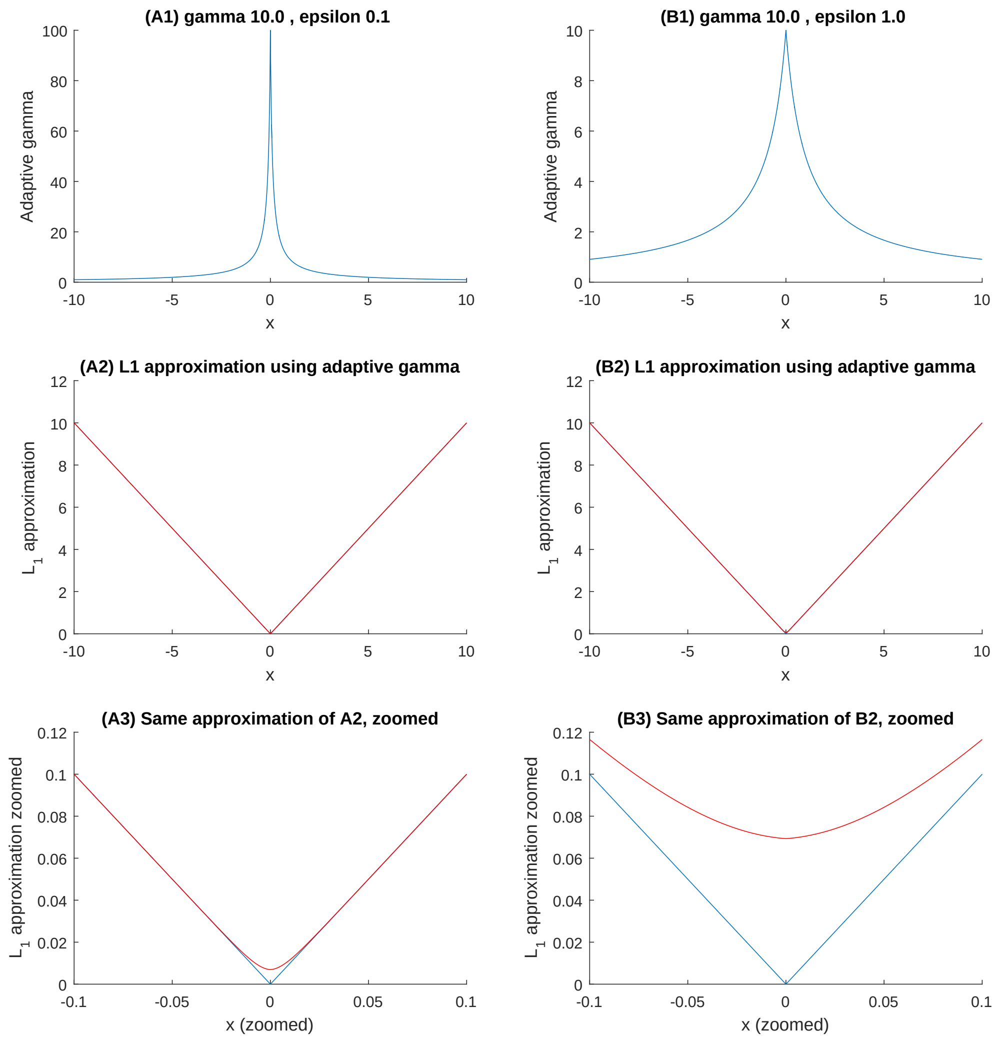


Figure 4.2 Adaptive gamma (γ_1) L_1 Approximation with different ϵ values

In equation 4.9, A_r represents constants terms while B_r and C_r are the coefficients of the terms w_r and w_r^2, respectively. This approximation is accurate around $\hat{w}_r$ where Taylor series is centered. Since $\hat{w}_r$ values change at each iteration, if w_r values differ significantly from constant point, $\hat{w}_r$, approximation diverges from true cost function. To ensure that w_r changes slowly, a new regularization term, $(w_r - \hat{w}_r)^2$, is added into the cost function. Refined cost function is given in Equation 4.10.

$$\begin{aligned} J^{(n)}(w) &= \frac{\lambda}{m}\sum_{s=1}^{m}(S_s\mathbf{H_s}w - y_s)^2 \\ &+ \frac{1}{l}\sum_{r=1}^{l}(A_r + B_r w_r + C_r w_r^2) \\ &+ \frac{1}{l}\sum_{r=1}^{l}(w_r - \hat{w}_r)^2 \end{aligned} \tag{4.10}$$

Equation 4.10 can be written in a matrix-vector form as follows:

$$\begin{aligned} J^{(n)}(w) &= \frac{\lambda}{m}(S\mathbf{H}w - y)^T(S\mathbf{H}w - y) \\ &+ \frac{1}{l}(v_A^T\vec{1} + v_B^T w + w^T C w) \\ &+ \frac{1}{l}(w - \hat{w})^T(w - \hat{w}) \end{aligned} \tag{4.11}$$

S	matrix form of S_s
v_A	vector form of A_r
$\vec{1}$	vector of ones
v_B	vector form of B_r
C	diagonal matrix form of C_r

Equation 4.11 is strictly convex and positive definite thus it has a unique global minimum. Therefore, to minimize $J^{(n)}(w)$ in Equation 4.11, the derivative with respect to w is taken and is equalized to zero.

This leads to system of linear equations:

$$
\begin{aligned}
&Mw = b \\
&where \\
&M = \frac{2\lambda}{m}(S\mathbf{H})^T(S\mathbf{H}) + \frac{2C+2}{l} \\
&b = \frac{2\lambda}{m}(S\mathbf{H})^T y + \frac{2\hat{w} - B_r}{l}
\end{aligned}
\tag{4.12}
$$

Final model is solved using algorithm 1 iteratively.

Due to the employed numerical approximations, small negative weights may occur around zero. Since our feasible set is $w \geq 0$, back projection to this set is performed after solving linear system at each iteration in algorithm 1. This kind of back-projection to feasible domain is commonly used [121]. Additionally, small weights in ensemble do not contribute to overall accuracy; therefore, these small weights are thresholded after iterations are completed.

Algorithm 1 SDWEC Pseudo code

```
1:  $H, y, \lambda, \beta, \gamma, \epsilon$ are initialized
2:  $w \leftarrow \vec{1}$
3:  $m, l \leftarrow size\, H_{mxl}$
4:  $k \leftarrow 25$                                        ▷ Maximum Iteration
5:  for n = 1 to k do
6:      $\hat{w} \leftarrow w$
7:      $\gamma_r \leftarrow \frac{\gamma}{|\hat{w}|+\epsilon}$
8:      construct S of diagonal form of $S_s$
9:      construct $v_B$ and $C$
10:     $M \leftarrow \frac{2\lambda}{m}(SH)^T(SH) + \frac{2C+2}{l}$
11:     $b \leftarrow \frac{2\lambda}{m}(SH)^T y + \frac{2\hat{w}-V_B}{l}$
12:     solve $Mw = b$
13:     $w = max(w, 0)$                      ▷ Back projection to $w \geq 0$ domain
14: end for
15: $w_{threshold} = argmin_{w_r}(P(w_r) - 10^{-3})^2$
16: $w = \begin{cases} w & if\ w > w_{threshold} \\ 0 & otherwise \end{cases}$
```

An example run of Algorithm 1 can be seen in Figure 4.3, where cost values for equations 4.2 and 4.11 decrease steadily. As seen in Figure 4.3, the difference

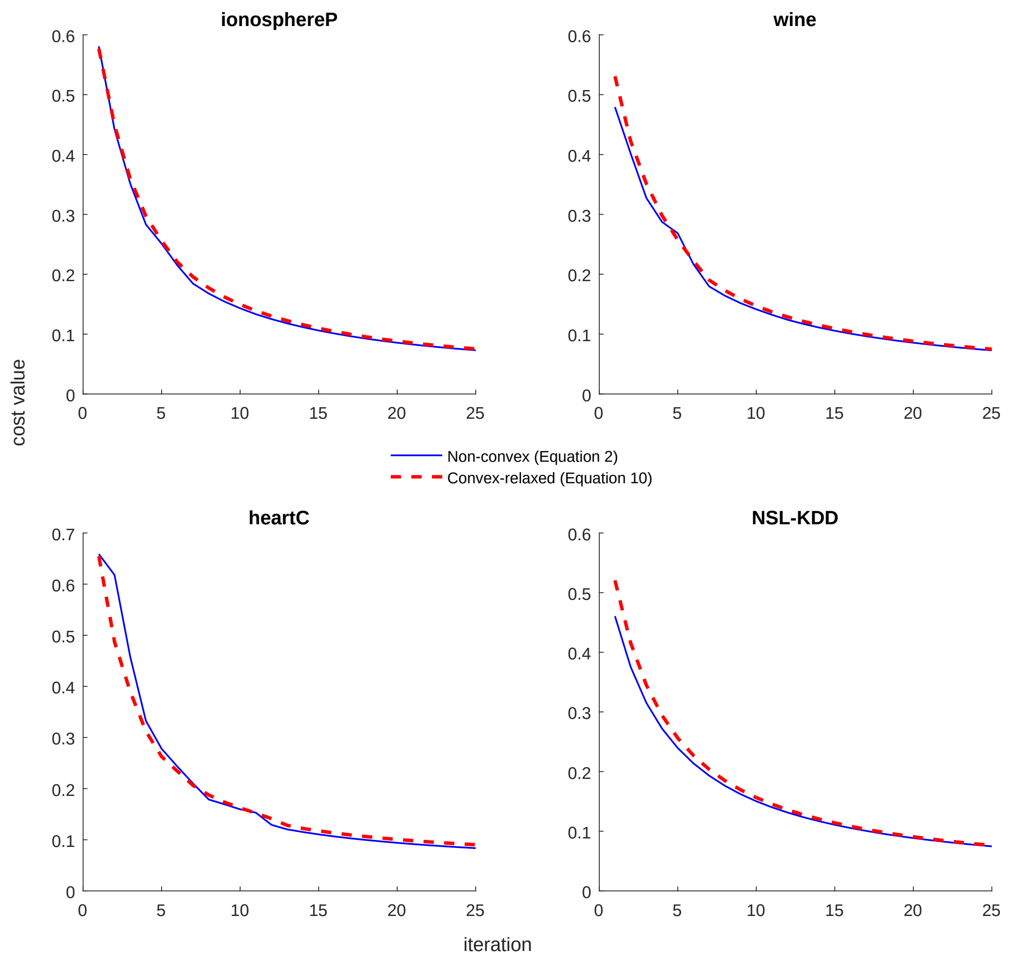


Figure 4.3 Minimization of the cost function for 4 datasets (Non-convex equation 4.2 vs convex relaxed equation 4.11).

between non-convex cost function and its convex relaxation is minimal especially in the final iterations. This shows that two functions converge to very similar values. Since non-convex Equation 4.2 and convex Equation 4.11 are converged to similar points, this converged points are within close proximity of the global minimum.

Non-convex Equation 4.2 and convex-relaxed Equation 4.11 are close to each other due to the term $(w - \hat{w})^2$ and employed iterative approach for minimization. These results show success of the proposed approximations.

4.3. Experimental results

The performance of SDWEC has been tested on 11 datasets; 10 UCI datasets and NSL-KDD [22]. NSL-KDD is a popular database for intrusion detection [122, 123]. SDWEC has been compared with the following algorithms : Single tree classifier (C4.5), bagging [84], WMV [70], and state-of-the-art ensemble QFWEC [109]. In all ensemble methods, 200 base classifiers (C4.5) are used. Each dataset is divided to training (80%), validation (10%), and testing (10%) datasets. This process has been repeated 10 times for cross validation. Mean values have been used in Table 4.2. The accuracy values for QFWEC in Table 4.2 are higher than original publication [109] since weights are found using validation dataset instead of training dataset, which provides better generalization.

4.3.1. Experimental results: sparsity

The principle of parsimony (sparsity) states that simple explanation should be preferred to complicated ones [116]. Sparsity mostly used for feature selection in machine learning. In our study, principle of sparsity is used for selecting among weak classifiers. According to dataset and hyper-parameters used, SDWEC achieves different sparsity levels. When SDWEC applied to 11 different datasets, sparsity levels between 0.80 and 0.88 has been achieved (Figure 4.4). This means that among 200 weak classifiers, 24 (sparsity level of 0.88) to 40 (sparsity level of 0.80) classifiers were used in ensemble.

Two different results with different sparsity values (A and B), chosen from Figure 4.5 have been provided in Table 4.2. SDWEC-A has no sparsity, all 200 base classifiers have been used in ensemble; thus, it has superior performance at the cost of testing time. SDWEC-A has best accuracy values in 4 out of 10 datasets and it is very close to top performing ones in others. SDWEC-B has 0.90 sparsity, 20/200 base

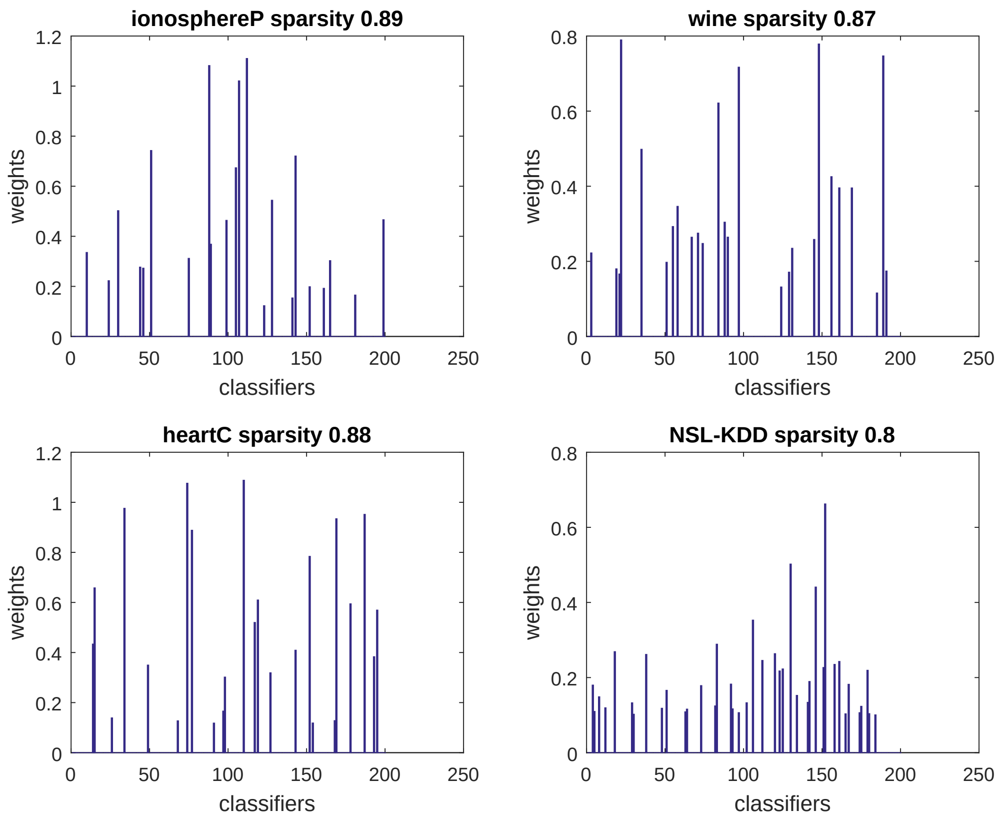


Figure 4.4 4 Datasets and their sparsity levels ($\lambda = 1, \beta = 10, \gamma = 20, \epsilon = 0.1$)

classifiers have been used in ensemble; nonetheless, it has best accuracy values in 2 out of 10 datasets. Besides, its accuracy values are marginally lower (about 2%) but its testing time is significantly better (90%) than the other approaches.

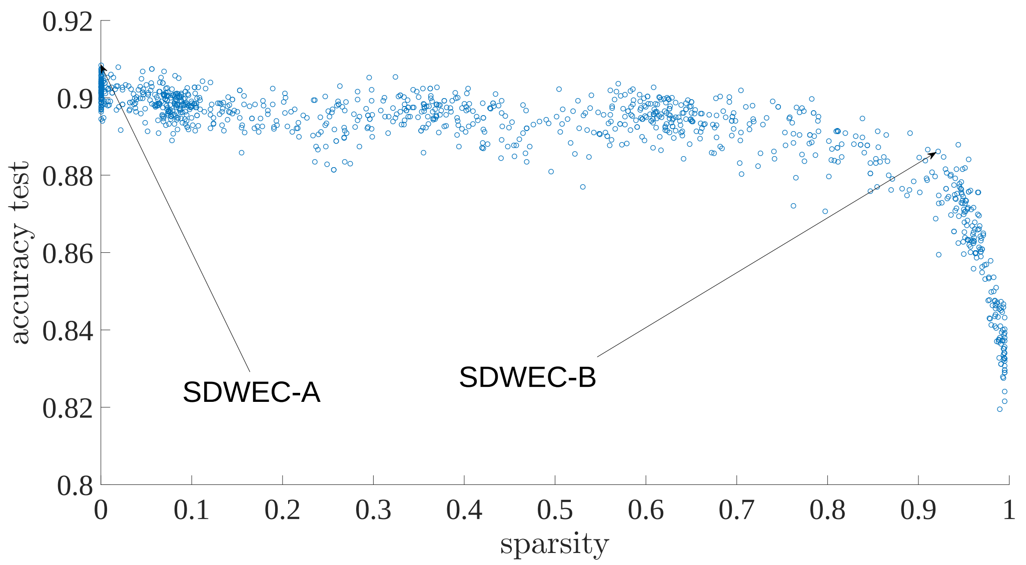


Figure 4.5 **Sparsity vs accuracy of SDWEC.** The sparsity and accuracy values come from the mean of 11 datasets. Corresponding values can be seen in Table 4.2

Table 4.2 **Comparison of accuracies (sparsity values are given in parentheses)**

Datasets	QFWEC	SDWEC-A	SDWEC-B	WMV	bagging	singleC
breast	0.9736	**0.9737** (0)	0.9532 (0.90)	0.9355	0.9722	0.9400
heartC	0.8085	0.8186 (0)	**0.8279** (0.90)	0.8118	0.8118	0.7268
ionosphere	0.9344	0.9371 (0)	**0.9427** (0.92)	0.9371	0.9342	0.8799
sonarP	0.8088	**0.8136** (0)	0.8126 (0.88)	0.7893	0.8088	0.7367
vehicleP	**0.9788**	0.9693 (0)	0.9539 (0.91)	0.9681	0.9670	0.9634
voteP	0.9576	**0.9703** (0)	0.9525 (0.84)	0.8509	0.9703	0.9533
waveform	**0.8812**	0.8652 (0)	0.8600 (0.93)	0.8634	0.8620	0.8220
wdbcP	**0.9595**	0.9507 (0)	0.9418 (0.88)	0.9489	0.9507	0.9138
wine	**0.9722**	0.9722 (0)	0.9605 (0.89)	0.7514	0.9719	0.9500
wpbcP	0.7989	**0.8036** (0)	0.7477 (0.91)	0.7850	0.7750	0.6911
NSL-KDD	0.9828	0.9766 (0)	0.9849 (0.88)	0.9610	0.9613	**0.9976**
SDWEC-A	$\lambda = 0.1$ $\beta = 35$ $\gamma = 5$ $\epsilon = 0.1$, Mean sparsity 0.00					
SDWEC-B	$\lambda = 10$ $\beta = 15$ $\gamma = 15$ $\epsilon = 1.0$, Mean sparsity 0.90					

4.3.2. Computational Complexity Analysis

In this section, Computational Complexity of SDWEC has been analyzed. Computation complexity of every line is given in Table 4.3 and and final computational complexity is determined. In Table 4.3, m stands for the number of data, l stands for the number of classifiers, and k stands for the iteration count.

Table 4.3 Computational complexity of SDWEC

Line #	and Code in Alg 1	Complexity	Notes
6	$\hat{w} \leftarrow w$	$O(l)$	
7	$\gamma_r \leftarrow \frac{\gamma}{\lvert\hat{w}\rvert+\epsilon}$	$O(l)$	
8	construct S as diagonal form of S_s	$O(ml)$	$S \leftarrow S_s$ sparse diagonal matrix $(m \times m)$ (Eq 4.4)
9	v_B	$O(l)$	
9	C	$O(l)$	C sparse diagonal matrix
10	SH	$O(ml)$	$\underset{m\times l}{X} = \underset{m\times m}{S} \times \underset{m\times l}{H}$
10	X^TX	$O(l^3)$	$X \leftarrow SH$, $X^T : O(l^2)$, $X^TX : O(l^3)$
10	$\frac{2\lambda}{m}[X^TX] + \frac{2C+2}{l}$	$O(l^2)$	
10	$M \leftarrow \frac{2\lambda}{m}X^TX + \frac{2C+2}{l}$	$O(l^3) + O(l^2) + O(l^3)$	
11	X^Ty	$O(l^2)$	
11	$\frac{2\hat{w}-V_B}{l}$	$O(l)$	
11	$b \leftarrow \frac{2\lambda}{m}X^Ty + \frac{2\hat{w}-V_B}{l}$	$O(l^2) + O(l)$	
12	solve $Mw = b$	$O(l^3)$	M dense, symmetric, real, positive definite $\rightarrow$ cholesky solver $\rightarrow O(\frac{2}{3}l^3)$
13	$w = max(w, 0)$	$O(l)$	

Computational complexity of for loop is $O(ml) + C_1O(l^3) + C_2O(l^2) + C_3O(l)$. Since $l \ll m$, dominant term is $O(ml)$ for the SH multiplication in line 10 of the Algorithm 1, where S is a diagonal matrix. Our iteration count is k, then final computational complexity of SDWEC is **O**(**kml**), that is linear in k, m, and l. This computational complexity analysis shows the computational efficiency of the proposed minimization.

Table 4.4 shows training time of SDWEC on various datasets. NSL-KDD (100778) dataset has 25 times more rows than waveform (4000) dataset. And training time of NSL-KDD (25.95) is about 25 times of waveform (0.96). These results show that practical execution times are in alignment with theoretical computational complexity analysis. Slight differences between theoretical analysis and actual execution times are due to implementation issues and caching in CPU architectures.

Table 4.4 SDWEC training time on various datasets,

Dataset	**Features**	**Rows**	**Time (sec.) l classifier count**			
			$l = 100$	$l = 200$	$l = 500$	$l = 1000$
breast-cancer	9	547	0.05	0.10	0.48	1.63
ionosphereP	34	280	0.04	0.07	0.31	1.01
wpbcP	33	155	0.03	0.06	0.26	0.89
wdbcP	30	456	0.05	0.09	0.44	1.34
wine	13	143	0.03	0.05	0.23	0.91
waveform	21	4000	0.43	0.96	3.01	7.78
voteP	16	186	0.03	0.07	0.24	0.97
vehicleP	18	667	0.06	0.18	0.73	1.83
sonarP	60	167	0.03	0.06	0.23	0.83
heartC	13	239	0.03	0.07	0.25	1.02
NSL-KDD	39	100778	12.73	25.95	80.23	204.59

4.4. Chapter discussion

In this chapter, a novel sparsity driven ensemble classifier method, SDWEC, has been presented. An efficient and accurate solution for original cost function (hard to minimize, non-convex, and non-differentiable) has been developed. Proposed solution uses a novel convex relaxation technique for *sign* function, and a novel adaptive *log-sum-exp* approximation that reduces numerical overflows. Computational complexity of SDWEC has been investigated theoretically and experimentally. SDWEC has a linear computational complexity in number of classifier used (l), number of instances in data (m), and number of algorithm iterations (k). SDWEC has been compared with other ensemble methods in well-known UCI and NSL-KDD datasets. According to the experiments, SDWEC decreases number of classifiers used in ensemble without significant loss of accuracy. By tuning parameters of SDWEC, a more sparse ensemble –thus, better testing time– can be obtained with a small decrease in accuracy.

5. CONCLUSION

In this thesis study, two different method for classifier ensembles were proposed and implemented in machine learning based intrusion detection systems. First method (GA-FS-WF) used heuristic optimization technique genetic algorithms to find best weights and features for IDS dataset NSL-KDD. Second method (SDWEC) used convex optimization techniques to find weights for pre-trained classifiers. Different from GA-FS-WF, second method SDWEC is tested on other datasets to show its generality to other domains. Even though both methods are suitable for use in IDS and other domains, SDWEC is faster to use and it gives guarantees that it converges near global optimal point. In addition, computational complexity of SDWEC is also given. Full dataset of NSL-KDD was used in both methods (GA-FS-WF and SDWEC) and compared with latest methods in the literature. According to our results in measured classifier performance metrics (accuracy, training and testing time and others), both methods are viable methods in machine learning based intrusion detection systems.

In the future studies, using other optimization techniques are planned to apply in this domain. For example ant colony optimization algorithms is very suitable to find suitable number of base classifiers and their weights while also selecting features in the dataset. Different ants will carry different classifiers and use different paths (features) to reach their destination. Those ants (classifiers) that use same paths to use same destination will combine to reach best solution. In this way, finding optimum number of classifiers to ensemble will be solved too.

As can be seen from Chapter 2, wide variety of base machine learning algorithms are applied on KDD99 and its derivative NSL-KDD and wide variety of hybrid algorithms are proposed on this datasets. Different from most studies, ours was one of the rare studies that proposed using convex optimization techniques in this dataset. Since convex optimization techniques are very popular in other domains, such as image processing and computer vision, we think that their usage will increase in intrusion detection in the future.

REFERENCES

[1] John Von Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, Champaign, IL, USA, 1966.

[2] Wikipedia. Timeline of computer viruses and worms. Web, 2010. URL `http://en.wikipedia.org/wiki/Timeline_of_computer_viruses_and_worms`.

[3] T.M. Chen and J.M. Robert. The evolution of viruses and worms. *Statistical methods in computer security*, 2004.

[4] Fred Cohen. Computer viruses : Theory and experiments. *Computers and Security*, 6(1):22 – 35, 1987. ISSN 0167-4048. doi: DOI:10.1016/0167-4048(87)90122-2. URL `http://www.sciencedirect.com/science/article/pii/0167404887901222`.

[5] MITRE. Cve common vulnerability exposure list, 2017. URL `https://cve.mitre.org/data/downloads/index.html`.

[6] R. Anderson. Why information security is hard-an economic perspective. In *Computer Security Applications Conference, 2001. ACSAC 2001. Proceedings 17th Annual*, pages 358–365. IEEE, 2001.

[7] ICQ407-235. Ddos black web site, 2017. URL `https://ddos-black.info`.

[8] Graham Cluley. Hire a ddos attack for as little as five dollars, 2016. URL `https://www.tripwire.com/state-of-security/featured/hire-a-ddos-attack-for-as-little-as-5/`.

[9] Robert McMillan. With botnets everywhere, ddos attacks get cheaper. PC World-News, 2009.

[10] Joyce Lofstrom. Himss cybersecurity survey. Technical report, HIMSS, 2016.

[11] Karen Scarfone and Peter Mell. *Guide to intrusion detection and prevention systems (IDPS)*. NIST, 2007.

[12] Dorothy E. Denning. An intrusion-detection model. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, 13(2):222–232, 1987.

[13] Robin Sommer and Vern Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, SP '10, pages 305–316, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-0-7695-4035-1.

[14] William H. Allen. Mixing wheat with the chaff: Creating useful test data for ids evaluation. *IEEE Security and Privacy*, 5:65–67, July 2007. ISSN 1540-7993. doi: 10.1109/MSP.2007.92. URL http://portal.acm.org/citation.cfm?id=1308457.1309257.

[15] S.T. Brugger. Kdd cup 99 dataset (network intrusion) considered harmful, 2007.

[16] Carlos A. Catania and Carlos García Garino. Automatic network intrusion detection: Current techniques and open issues. *Computers & Electrical Engineering*, 38(5):1062 – 1072, 2012. ISSN 0045-7906. doi: http://dx.doi.org/10.1016/j.compeleceng.2012.05.013. URL http://www.sciencedirect.com/science/article/pii/S0045790612001073. Special issue on Recent Advances in Security and Privacy in Distributed Communications and Image processing.

[17] Neminath Hubballi and Vinoth Suryanarayanan. False alarm minimization techniques in signature-based intrusion detection systems: A survey. *Computer Communications*, 49:1 – 17, 2014. ISSN 0140-3664. doi: http://dx.doi.org/10.1016/j.comcom.2014.04.012. URL http://www.sciencedirect.com/science/article/pii/S0140366414001480.

[18] Zakira Inayat, Abdullah Gani, Nor Badrul Anuar, Muhammad Khurram Khan, and Shahid Anwar. Intrusion response systems: Foundations, design, and challenges. *Journal of Network and Computer Applications*, 62: 53 – 74, 2016. ISSN 1084-8045. doi: http://dx.doi.org/10.1016/j.jnca.2015.12.006. URL http://www.sciencedirect.com/science/article/pii/S1084804515002994. Key:Review.

[19] Mohiuddin Ahmed, Abdun Naser Mahmood, and Jiankun Hu. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, 60:19 – 31, 2016. ISSN 1084-8045. doi: http://dx.doi.org/10.1016/j.jnca.2015.11.016. URL http://www.sciencedirect.com/science/article/pii/S1084804515002891.

[20] Sannasi Ganapathy, Kanagasabai Kulothungan, Sannasy Muthurajkumar, Muthusamy Vijayalakshmi, Palanichamy Yogesh, and Arputharaj Kannan. Intelligent feature selection and classification techniques for intrusion detection in networks: a survey. *EURASIP Journal on Wireless Communications and Networking*, 2013(1):271, 2013. doi: 10.1186/1687-1499-2013-271. URL http://dx.doi.org/10.1186/1687-1499-2013-271.

[21] C. Kolias, G. Kambourakis, and M. Maragoudakis. Swarm intelligence in intrusion detection: A survey. *Computers and Security*, 30(8):625–642, November 2011. ISSN 0167-4048. doi: 10.1016/j.cose.2011.08.009. URL http://dx.doi.org/10.1016/j.cose.2011.08.009.

[22] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. A detailed analysis of the kdd cup 99 data set. In *Proceedings of the Second IEEE international conference on Computational intelligence for security and defense applications*, CISDA'09, pages 53–58, 2009.

[23] Wenke Lee and Salvatore J. Stolfo. A framework for constructing features and models for intrusion detection systems. *ACM Transactions on Information and System Security*, 3:227–261, November 2000. ISSN 1094-9224. doi: http://doi.acm.org/10.1145/382912.382914. URL http://doi.acm.org/10.1145/382912.382914.

[24] Chih-Fong Tsai, Yu-Feng Hsu, Chia-Ying Lin, and Wei-Yang Lin. Intrusion detection by machine learning: A review. *Expert Systems with Applications*, 36(10):11994 – 12000, 2009. ISSN 0957-4174. doi: DOI:10.1016/j.eswa.2009.05.029. URL http://www.sciencedirect.com/science/article/pii/S0957417409004801.

[25] Hung-Jen Liao, Chun-Hung Richard Lin, Ying-Chih Lin, and Kuang-Yuan Tung. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16 – 24, 2013. ISSN 1084-8045. doi: http://dx.doi.org/10.1016/j.jnca.2012.09.004. URL http://www.sciencedirect.com/science/article/pii/S1084804512001944. Key:Review.

[26] Chirag Modi, Dhiren Patel, Bhavesh Borisaniya, Hiren Patel, Avi Patel, and Muttukrishnan Rajarajan. A survey of intrusion detection techniques in cloud. *Journal of Network and Computer Applications*, 36(1):42 – 57, 2013. ISSN 1084-8045. doi: http://dx.doi.org/10.1016/j.jnca.2012.05.003. URL http://www.sciencedirect.com/science/article/pii/S1084804512001178. Key:Review.

[27] Hua Yang, Tao Li, Xinlei Hu, Feng Wang, and Yang Zou. A survey of artificial immune system based intrusion detection. *The Scientific World Journal*, 2014:11, 2014. doi: 10.1155/2014/156790. URL http://dx.doi.org/10.1155/2014/156790.

[28] Saber Elsayed, Ruhul Sarker, and Daryl Essam. Survey of uses of evolutionary computation algorithms and swarm intelligence for

network intrusion detection. *International Journal of Computational Intelligence and Applications*, 14(04):1550025, 2015. doi: 10.1142/S146902681550025X. URL http://www.worldscientific.com/doi/abs/10.1142/S146902681550025X.

[29] Gianluigi Folino and Pietro Sabatino. Ensemble based collaborative and distributed intrusion detection systems: A survey. *Journal of Network and Computer Applications*, 66:1 – 16, 2016. ISSN 1084-8045. doi: http://dx.doi.org/10.1016/j.jnca.2016.03.011. URL http://www.sciencedirect.com/science/article/pii/S1084804516300248. Key:Review.

[30] R. K. Cunningham, R. P. Lippmann, D. J. Fried, S. L. Garfinkel, I. Graf, K. R. Kendall, S. E. Webster, D. Wyschogrod, and M. A. Zissman. Evaluating intrusion detection systems without attacking your friends: The 1998 darpa intrusion detection evaluation. Technical report, MASSACHUSETTS INST OF TECH LEXINGTON LINCOLN LAB, 1999.

[31] R.P. Lippmann, D.J. Fried, I. Graf, J.W. Haines, K.R. Kendall, D. McClung, D. Weber, S.E. Webster, D. Wyschogrod, R.K. Cunningham, and M.A. Zissman. Evaluating intrusion detection systems: the 1998 darpa off-line intrusion detection evaluation. In *DARPA Information Survivability Conference and Exposition, 2000. DISCEX '00. Proceedings*, volume 2, pages 12 –26 vol.2, 2000. doi: 10.1109/DISCEX.2000.821506.

[32] Richard Lippmann, Joshua W. Haines, David J. Fried, Jonathan Korba, and Kumar Das. The 1999 darpa off-line intrusion detection evaluation. *Computer Networks*, 34:579–595, October 2000. ISSN 1389-1286. doi: 10.1016/S1389-1286(00)00139-0. URL http://portal.acm.org/citation.cfm?id=361116.361124.

[33] K. Kendall. A database of computer attacks for the evaluation of intrusion detection systems. Master's thesis, MIT - Massachusetts Institute of Technology, 1999.

[34] KDD. Intrusion detector learning, 1999.

[35] Bernhard Pfahringer. Winning the kdd99 classification cup: bagged boosting. *SIGKDD Explor. Newsl.*, 1:65–66, January 2000. ISSN 1931-0145. doi: http://doi.acm.org/10.1145/846183.846200. URL http://doi.acm.org/10.1145/846183.846200.

[36] Maheshkumar Sabhnani and Gursel Serpen. Why machine learning algorithms fail in misuse detection on kdd intrusion detection data set. *Intell.*

Data Anal., 8:403–415, September 2004. ISSN 1088-467X. URL http://portal.acm.org/citation.cfm?id=1293805.1293811.

[37] Adel Sabry Eesa, Zeynep Orman, and Adnan Mohsin Abdulazeez Brifcani. A novel feature-selection approach based on the cuttlefish optimization algorithm for intrusion detection systems. *Expert Systems with Applications*, 42(5):2670 – 2679, 2015. ISSN 0957-4174. doi: http://dx.doi.org/10.1016/j.eswa.2014.11.009. URL http://www.sciencedirect.com/science/article/pii/S0957417414006952.

[38] Shi-Jinn Horng, Ming-Yang Su, Yuan-Hsin Chen, Tzong-Wann Kao, Rong-Jian Chen, Jui-Lin Lai, and Citra Dwi Perkasa. A novel intrusion detection system based on hierarchical clustering and support vector machines. *Expert Systems with Applications*, 38(1):306 – 313, 2011. ISSN 0957-4174. doi: http://dx.doi.org/10.1016/j.eswa.2010.06.066. URL http://www.sciencedirect.com/science/article/pii/S0957417410005701.

[39] Feng Jiang, Yuefei Sui, and Lin Zhou. A relative decision entropy-based feature selection approach. *Pattern Recognition*, 48(7):2151 – 2163, 2015. ISSN 0031-3203. doi: http://dx.doi.org/10.1016/j.patcog.2015.01.023. URL http://www.sciencedirect.com/science/article/pii/S0031320315000424.

[40] Feng Jiang, Yuefei Sui, and Cungen Cao. An incremental decision tree algorithm based on rough sets and its application in intrusion detection. *Artificial Intelligence Review*, 40(4):517–530, 2013. ISSN 0269-2821. doi: 10.1007/s10462-011-9293-z. URL http://dx.doi.org/10.1007/s10462-011-9293-z.

[41] Wafa Alsharafat. Applying artificial neural network and extended classifier system for network intrusion detection. *The International Arab Journal of Information Technology*, 10(3), May 2013.

[42] Gulshan Kumar and Krishan Kumar. Design of an evolutionary approach for intrusion detection. *The Scientific World Journal*, 2013:14, 2013. doi: 10.1155/2013/962185. URL http://dx.doi.org/10.1155/2013/962185.

[43] Tieming Chen, Xu Zhang, Shichao Jin, and Okhee Kim. Efficient classification using parallel and scalable compressed model and its application on intrusion detection. *Expert Systems with Applications*, 41(13):5972 – 5983, 2014. ISSN 0957-4174. doi: http://dx.doi.org/10.1016/j.eswa.2014.04.009. URL http://www.sciencedirect.com/science/article/pii/S0957417414002048.

[44] Samaneh Rastegari, Philip Hingston, and Chiou-Peng Lam. Evolving statistical rulesets for network intrusion detection. *Applied Soft Computing*, 33:348 – 359, 2015. ISSN 1568-4946. doi: http://dx.doi.org/10.1016/j.asoc.2015.04.041. URL `http://www.sciencedirect.com/science/article/pii/S1568494615002689`.

[45] Yang Yi, Jiansheng Wu, and Wei Xu. Incremental svm based on reserved set for network intrusion detection. *Expert Systems with Applications*, 38(6):7698 – 7707, 2011. ISSN 0957-4174. doi: http://dx.doi.org/10.1016/j.eswa.2010.12.141.

[46] Wenying Feng, Qinglei Zhang, Gongzhu Hu, and Jimmy Xiangji Huang. Mining network data for intrusion detection through combining svms with ant colony networks. *Future Generation Computer Systems*, 37:127 – 140, 2014. ISSN 0167-739X. doi: http://dx.doi.org/10.1016/j.future.2013.06.027. URL `http://www.sciencedirect.com/science/article/pii/S0167739X13001416`. Special Section: Innovative Methods and Algorithms for Advanced Data-Intensive ComputingSpecial Section: Semantics, Intelligent processing and services for big dataSpecial Section: Advances in Data-Intensive Modelling and SimulationSpecial Section: Hybrid Intelligence for Growing Internet and its Applications.

[47] Wathiq Laftah Al-Yaseen, Zulaiha Ali Othman, and Mohd Zakree Ahmad Nazri. Multi-level hybrid support vector machine and extreme learning machine based on modified k-means for intrusion detection system. *Expert Systems with Applications*, 67:296 – 303, 2017. ISSN 0957-4174. doi: http://dx.doi.org/10.1016/j.eswa.2016.09.041. URL `http://www.sciencedirect.com/science/article/pii/S0957417416305310`.

[48] Fatemeh Amiri, MohammadMahdi Rezaei Yousefi, Caro Lucas, Azadeh Shakery, and Nasser Yazdani. Mutual information-based feature selection for intrusion detection systems. *Journal of Network and Computer Applications*, 34(4):1184 – 1199, 2011. ISSN 1084-8045. doi: http://dx.doi.org/10.1016/j.jnca.2011.01.002. URL `http://www.sciencedirect.com/science/article/pii/S1084804511000038`. Advanced Topics in Cloud Computing.

[49] Salma Elhag, Alberto Fernández, Abdullah Bawakid, Saleh Alshomrani, and Francisco Herrera. On the combination of genetic fuzzy systems and pairwise learning for improving detection rates on intrusion detection systems. *Expert Systems with Applications*, 42(1):193 – 202, 2015. ISSN 0957-4174. doi: http://dx.doi.org/10.1016/j.eswa.2014.08.002. URL `http://www.sciencedirect.com/science/article/pii/S0957417414004783`.

[50] J. McHugh. Testing intrusion detection systems: A critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM Transactions on Information and System Security*, 3(4): 262–294, 2000.

[51] M.V. Mahoney and P.K. Chan. An analysis of the 1999 darpa/lincoln laboratory evaluation data for network anomaly detection. In *Recent Advances in Intrusion Detection*, pages 220–237. Springer, 2003.

[52] S Terry Brugger and Jedadiah Chow. An assessment of the darpa ids evaluation dataset using snort, 2005.

[53] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, March 2003. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=944919.944968`.

[54] kdnuggets. New poll: Primary programming language for analytics, data mining, data science. `http://goo.gl/ESGonu`, 2015.

[55] Atilla Özgür and Hamit Erdem. A review of kdd99 dataset usage in intrusion detection and machine learning between 2010 and 2015. *PeerJ Preprints*, 2016. doi: 4:e1954v1https://doi.org/10.7287/peerj.preprints.1954v1.

[56] Oktay Yıldız, Mesut Tez, H. Şakir Bilge, M. Ali Akcayol, and İnan Güler. Meme kanseri sınıflandırması İçin veri füzyonu ve genetik algoritma tabanlı gen seçimi. *Journal of the Faculty of Engineering and Architecture of Gazi University*, 27(3), 2012. ISSN 1304-4915. URL `http://www.mmfdergi.gazi.edu.tr/article/view/1061000567`.

[57] Javier Pérez-Rodríguez, Alexis Germán Arroyo-Peña, and Nicolás García-Pedrajas. Simultaneous instance and feature selection and weighting using evolutionary computation: Proposal and study. *Applied Soft Computing*, 37:416 – 443, 2015. ISSN 1568-4946. doi: http://dx.doi.org/10.1016/j.asoc.2015.07.046. URL `http://www.sciencedirect.com/science/article/pii/S1568494615005062`.

[58] Şeref Sağıroğlu, Esra Nergis Yolaçan, and Uraz Yavanoğlu. Zeki saldırı tespit sistemi tasarımı ve gerçekleştirilmesi. *Journal of the Faculty of Engineering and Architecture of Gazi University*, 26(2):325–340, 2011.

[59] Taner Tuncer and Yetkin Tatar. Fpga tabanlı programlanabilir gömülü saldırı tespit sistemi gerçekleştirilmesi. *Journal of the Faculty of Engineering and*

Architecture of Gazi University, 27(1), 2012. ISSN 1304-4915. URL http://www.mmfdergi.gazi.edu.tr/article/view/1061000504.

[60] Tim Bass. Intrusion detection systems and multisensor data fusion. *Communications of the ACM*, 43:99–105, April 2000. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/332051.332079. URL http://doi.acm.org/10.1145/332051.332079.

[61] Ludmila I. Kuncheva, James C. Bezdek, and Robert P.W. Duin. Decision templates for multiple classifier fusion: an experimental comparison. *Pattern Recognition*, 34(2):299 – 314, 2001.

[62] Yong Wang, Huihua Yang, Xingyu Wang, and Ruixia Zhang. Distributed intrusion detection system based on data fusion method. In *Intelligent Control and Automation, 2004. WCICA 2004. Fifth World Congress on*, volume 5, pages 4331–4334 Vol.5, June 2004. doi: 10.1109/WCICA.2004.1342330.

[63] Yong Zhang, Hongrui Zhang, Jing Cai, and Binbin Yang. A weighted voting classifier based on differential evolution. *Abstract and Applied Analysis*, 2014: 6, 2014.

[64] J. Sylvester and N. V. Chawla. Evolutionary ensemble creation and thinning. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 5148–5155, 2006.

[65] Y. Maghsoudi, A. Alimohammadi , M.J. Valadan Zoej, and B. Mojaradi. Weighted combination of multiple classifiers for the classification of hyperspectral images using a genetic algorithm. In *ISPRS Commission I Symposium ”From Sensors to Imagery”*, 2006.

[66] Murat Dener, M. Ali Akcayol, Sinan Toklu, and Ömer Bay. Zamana bağlı dinamik en kısa yol problemi İçin genetik algoritma tabanlı yeni bir algoritma. *Journal of the Faculty of Engineering and Architecture of Gazi University*, 26(4), 2013. ISSN 1304-4915. URL http://www.mmfdergi.gazi.edu.tr/article/view/1061000687.

[67] Mehmet Üstündağ, Engin Avcı, Muammer Gökbulut, and Fikret Ata. Dalgacık paket dönüşümü ve genetik algoritma kullanarak zayıf radar sinyallerinin gürültüden arındırılması. *Journal of the Faculty of Engineering and Architecture of Gazi University*, 29(2), 2014. ISSN 1304-4915. URL http://www.mmfdergi.gazi.edu.tr/article/view/1061001387.

[68] Barış Gürsu. Ceza fonksiyonuyla durdurmalı genetik algoritmalar ile transformatör merkezlerinde optimum aşırı akım röle koordinasyonu. *Journal of the Faculty of Engineering and Architecture of Gazi University*, 29(4), 2014. ISSN 1304-4915. URL http://www.mmfdergi.gazi.edu.tr/article/view/1061001785.

[69] Christopher Kruegel, Fredrik Valeur, Giovanni Vigna, and Richard Kemmerer. Stateful intrusion detection for high-speed network's. In *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*, pages 285–293. IEEE, 2002.

[70] Ludmila I Kuncheva. *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons, 2005.

[71] Adem Kalınlı and Özgür Aksu. Baskın gen seçimi operatörüne dayalı genetik algoritma modeli. *Journal of the Faculty of Engineering and Architecture of Gazi University*, 26(4), 2013. ISSN 1304-4915. URL http://www.mmfdergi.gazi.edu.tr/article/view/1061000682.

[72] Ludmila I. Kuncheva and Christopher J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51(2):181–207, 2003.

[73] Seung-Ho Kang and Kuinam J. Kim. A feature selection approach to find optimal feature subsets for the network intrusion detection system. *Cluster Computing*, 19(1):325–333, 2016. ISSN 1573-7543. doi: 10.1007/s10586-015-0527-8. URL http://dx.doi.org/10.1007/s10586-015-0527-8.

[74] Clayton R. Pereira, Rodrigo Y.M. Nakamura, Kelton A.P. Costa, and João P. Papa. An optimum-path forest framework for intrusion detection in computer networks. *Engineering Applications of Artificial Intelligence*, 25(6):1226 – 1234, 2012. ISSN 0952-1976. doi: http://dx.doi.org/10.1016/j.engappai.2012.03.008. URL http://www.sciencedirect.com/science/article/pii/S0952197612000644.

[75] Neda Afzali Seresht and Reza Azmi. Mais-ids: A distributed intrusion detection system using multi-agent ais approach. *Engineering Applications of Artificial Intelligence*, 35:286 – 298, 2014. ISSN 0952-1976. doi: http://dx.doi.org/10.1016/j.engappai.2014.06.022. URL http://www.sciencedirect.com/science/article/pii/S0952197614001444.

[76] Dewan Md. Farid, Li Zhang, Chowdhury Mofizur Rahman, M.A. Hossain, and Rebecca Strachan. Hybrid decision tree and naïve bayes classifiers for

multi-class classification tasks. *Expert Systems with Applications*, 41(4, Part 2):1937 – 1946, 2014. ISSN 0957-4174. doi: http://dx.doi.org/10.1016/j.eswa.2013.08.089. URL http://www.sciencedirect.com/science/article/pii/S0957417413007100.

[77] Raman Singh, Harish Kumar, and R.K. Singla. An intrusion detection system using network traffic profiling and online sequential extreme learning machine. *Expert Systems with Applications*, 42(22):8609 – 8624, 2015. ISSN 0957-4174. doi: http://dx.doi.org/10.1016/j.eswa.2015.07.015. URL http://www.sciencedirect.com/science/article/pii/S0957417415004753.

[78] Sangeeta Bhattacharya and Subramanian Selvakumar. Lawra: a layered wrapper feature selection approach for network attack detection. *Security and Communication Networks*, 8(18):3459–3468, 2015. ISSN 1939-0122. doi: 10.1002/sec.1270. URL http://dx.doi.org/10.1002/sec.1270.

[79] Mehdi Mohammadi, Bijan Raahemi, Ahmad Akbari, and Babak Nassersharif. New class-dependent feature transformation for intrusion detection systems. *Security and Communication Networks*, 5(12):1296–1311, 2012. ISSN 1939-0122. doi: 10.1002/sec.403. URL http://dx.doi.org/10.1002/sec.403.

[80] Qiang Liu, Jianping Yin, Victor C. M. Leung, Jun-Hai Zhai, Zhiping Cai, and Jiarun Lin. Applying a new localized generalization error model to design neural networks trained with extreme learning machine. *Neural Computing and Applications*, 27(1):59–66, 2016. ISSN 1433-3058. doi: 10.1007/s00521-014-1549-5. URL http://dx.doi.org/10.1007/s00521-014-1549-5.

[81] Eduardo De la Hoz, Emiro De La Hoz, Andrés Ortiz, Julio Ortega, and Beatriz Prieto. Pca filtering and probabilistic som for network intrusion detection. *Neurocomputing*, 164:71 – 81, 2015. ISSN 0925-2312. doi: http://dx.doi.org/10.1016/j.neucom.2014.09.083. URL http://www.sciencedirect.com/science/article/pii/S0925231215002982.

[82] Yoav Freund and Robert E Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, pages 23–37. Springer, 1995.

[83] Bartosz Krawczyk and Michał Woźniak. Diversity measures for one-class classifier ensembles. *Neurocomputing*, 126:36 – 44, 2014.

[84] Leo Breiman. Bagging predictors. *Machine Learning*, 24:123–140, August 1996.

[85] Ludmila I. Kuncheva and Juan J. Rodríguez. A weighted voting framework for classifiers ensembles. *Knowledge and Information Systems*, 38(2):259–275, 2014.

[86] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *Evolutionary Computation, IEEE Transactions on*, 1(1):67 –82, apr 1997. ISSN 1089-778X. doi: 10.1109/4235.585893.

[87] David H. Wolpert. *The Supervised Learning No-Free-Lunch Theorems*, pages 25–42. Springer London, London, 2002.

[88] Y. Freund, R. Schapire, and N. Abe. A short introduction to boosting. *JOURNAL-JAPANESE SOCIETY FOR ARTIFICIAL INTELLIGENCE*, 14: 771–780, 1999.

[89] P. Gurram and H. Kwon. Sparse kernel-based ensemble learning with fully optimized kernel parameters for hyperspectral classification problems. *IEEE Transactions on Geoscience and Remote Sensing*, 51(2):787–802, Feb 2013. ISSN 0196-2892. doi: 10.1109/TGRS.2012.2203603.

[90] Heesung Lee, Euntai Kim, and Witold Pedrycz. A new selective neural network ensemble with negative correlation. *Applied Intelligence*, 37(4):488–498, 2012.

[91] Jin Tian and Nan Feng. Adaptive generalized ensemble construction with feature selection and its application in recommendation. *International Journal of Computational Intelligence Systems*, 7(sup2):35–43, 2014.

[92] Li Zhang, Wei-Da Zhou, and Fan-Zhang Li. Kernel sparse representation-based classifier ensemble for face recognition. *Multimedia Tools and Applications*, 74(1):123–137, 2015.

[93] Sunghan Kim, Fabien Scalzo, Donatello Telesca, and Xiao Hu. Ensemble of sparse classifiers for high-dimensional biological data. *International Journal of Data Mining and Bioinformatics*, 12(2):167–183, 2015.

[94] Nan Li and Zhi-Hua Zhou. Selective ensemble under regularization framework. In *Multiple Classifier Systems: 8th International Workshop, MCS 2009*, pages 293–303, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[95] Hyunjoong Kim, Hyeuk Kim, Hojin Moon, and Hongshik Ahn. A weight-adjusted voting algorithm for ensembles of classifiers. *Journal of the Korean Statistical Society*, 40(4):437 – 449, 2011.

[96] Shasha Mao, L.C. Jiao, Lin Xiong, and Shuiping Gou. Greedy optimization classifiers ensemble based on diversity. *Pattern Recognition*, 44(6):1245 – 1261, 2011.

[97] Li Zhang and Wei-Da Zhou. Sparse ensembles using weighted combination methods based on linear programming. *Pattern Recognition*, 44(1):97 – 106, 2011.

[98] Noam Goldberg and Jonathan Eckstein. Sparse weighted voting classifier selection and its linear programming relaxations. *Information Processing Letters*, 112(12):481 – 486, 2012.

[99] A. B. Santos, A. de A. Araújo, and D. Menotti. Combiner of classifiers using genetic algorithm for classification of remote sensed hyperspectral images. In *2012 IEEE International Geoscience and Remote Sensing Symposium*, 2012.

[100] Xu-Cheng Yin, Kaizhu Huang, Hong-Wei Hao, Khalid Iqbal, and Zhi-Bin Wang. Classifier ensemble using a heuristic learning with sparsity and diversity. In *Neural Information Processing: 19th International Conference, ICONIP 2012*, Berlin, Heidelberg, 2012.

[101] Yuxin Meng and Lam-For Kwok. Enhancing false alarm reduction using voted ensemble selection in intrusion detection. *International Journal of Computational Intelligence Systems*, 6(4):626–638, 2013.

[102] S. L. J. L. Tinoco, H. G. Santos, D. Menotti, A. B. Santos, and J. A. dos Santos. Ensemble of classifiers for remote sensed hyperspectral land cover analysis: An approach based on linear programming and weighted linear combination. In *2013 IEEE International Geoscience and Remote Sensing Symposium - IGARSS*, 2013.

[103] V. Hautamäki, T. Kinnunen, F. Sedlák, K. A. Lee, B. Ma, and H. Li. Sparse classifier fusion for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(8):1622–1631, Aug 2013. ISSN 1558-7916. doi: 10.1109/TASL.2013.2256895.

[104] Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 2006.

[105] Mehmet Umut Sen and Hakan Erdogan. Linear classifier combination and selection using group sparse regularization and hinge loss. *Pattern Recognition Letters*, 34(3):265 – 274, 2013.

[106] S. Mao, L. Xiong, L. C. Jiao, S. Zhang, and B. Chen. Weighted ensemble based on 0-1 matrix decomposition. *Electronics Letters*, 49(2):116–118, January 2013. ISSN 0013-5194. doi: 10.1049/el.2012.3528.

[107] Xu-Cheng Yin, Kaizhu Huang, Hong-Wei Hao, Khalid Iqbal, and Zhi-Bin Wang. A novel classifier ensemble method with sparsity and diversity. *Neurocomputing*, 134:214 – 221, 2014.

[108] Xu-Cheng Yin, Kaizhu Huang, Chun Yang, and Hong-Wei Hao. Convex ensemble learning with sparsity and diversity. *Information Fusion*, 20:49 – 59, 2014.

[109] Shasha Mao, Licheng Jiao, Lin Xiong, Shuiping Gou, Bo Chen, and Sai-Kit Yeung. Weighted classifier ensemble based on quadratic form. *Pattern Recognition*, 48(5):1688 – 1706, 2015.

[110] S. Shukla, J. Sharma, S. Khare, S. Kochkar, and V. Dharni. A novel sparse ensemble pruning algorithm using a new diversity measure. In *2015 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, pages 1–4, Dec 2015. doi: 10.1109/ICCIC.2015.7435815.

[111] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[112] S. G. Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, Dec 1993. ISSN 1053-587X. doi: 10.1109/78.258082.

[113] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx`, March 2014.

[114] Michael Grant and Stephen Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited, 2008. `http://stanford.edu/~boyd/graph_dcp.html`.

[115] Fatih Nar, Atilla Ozgur, and A. Nurdan Saran. Sparsity-driven change detection in multitemporal SAR images. *IEEE Geoscience and Remote Sensing Letters*, 13(7), 2016.

[116] Francis Bach, Rodolphe Jenatton, Julien Mairal, and Guillaume Obozinski. Optimization with sparsity-inducing penalties. *Foundations and Trends® in Machine Learning*, 4(1):1–106, 2012.

[117] Dongdong Ge, Xiaoye Jiang, and Yinyu Ye. A note on the complexity of l p minimization. *Mathematical programming*, 129(2):285–299, 2011.

[118] J. A. Tropp. Just relax: convex programming methods for identifying sparse signals in noise. *IEEE Transactions on Information Theory*, 52(3):1030–1051, March 2006.

[119] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[120] IEEE standard for binary floating-point arithmetic, 1985. Note: Standard 754–1985.

[121] Thomas Pock, Daniel Cremers, Horst Bischof, and Antonin Chambolle. An algorithm for minimizing the mumford-shah functional. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1133–1140. IEEE, 2009.

[122] Mohanad Albayati and Biju Issac. Analysis of intelligent classifiers and enhancing the detection accuracy for intrusion detection system. *International Journal of Computational Intelligence Systems*, 8(5):841–853, 2015. doi: 10.1080/18756891.2015.1084705. URL http://dx.doi.org/10.1080/18756891.2015.1084705.

[123] Jamal Hussain, Samuel Lalmuanawma, and Lalrinfela Chhakchhuak. A two-stage hybrid classification technique for network intrusion detection system. *International Journal of Computational Intelligence Systems*, 9(5): 863–875, 2016. ISSN 1875-6891. doi: {10.1080/18756891.2016.1237186}.