Numerical Analysis for Relevant Features in Intrusion Detection (NARFid)

THESIS

José Andrés González, Captain, USAF

AFIT/GCE/ENG/09-02

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

AFIT/GCE/ENG/09-02

Numerical Analysis for Relevant Features in Intrusion Detection (NARFid)

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Computer Engineering

José Andrés González, BS

Captain, USAF

March 2009

AFIT/GCE/ENG/09-02

Numerical Analysis for Relevant Features in Intrusion Detection (NARFid)

José Andrés González, BS

Captain, USAF

Approved:

/signed/ 5 March 2009

Michael J. Mendenhall, Maj, PhD date
(Chairman)

/signed/ 5 March 2009

Barry E. Mullins, PhD date
(Member)

/signed/ 5 March 2009

Gilbert L. Peterson, PhD date
(Member)

AFIT/GCE/ENG/09-02

Abstract

Identification of cyber attacks and network services is a robust field of study in the machine learning community. Less effort has been focused on understanding the domain space of real network data in identifying important features for cyber attack and network service classification. Motivations for such work allow for anomaly detection systems with less requirements on data “sniffed” off the network, extraction of features from the traffic, reduced learning time of algorithms, and ideally increased classification performance of anomalous behavior.

This thesis evaluates the usefulness of a good feature subset for the general classification task of identifying cyber attacks and network services. The generality of the selected features elucidates the relevance or irrelevance of the feature set for the classification task of intrusion detection. Additionally, the thesis provides an extension to the Bhattacharyya method, which selects features by means of inter-class separability (Bhattacharyya coefficient). The extension for multiple class problems selects a minimal set of features with the best separability across all class pairs.

Several feature selection algorithms (e.g., accuracy rate with genetic algorithm, RELIEF-F, GRLVQI, median Bhattacharyya and minimum surface Bhattacharyya methods) create feature subsets that describe the decision boundary for intrusion detection problems. The selected feature subsets maintain or improve the classification performance for at least three out of the four anomaly detectors (i.e., classifiers) under test. The feature subsets, which illustrate generality for the intrusion detection problem, range in size from 12 to 27 features. The original feature set consists of 248 features. Of the feature subsets demonstrating generality, the extension to the Bhattacharyya method generates the second smallest feature subset. This thesis quantitatively demonstrates that a relatively small feature set may be used for intrusion detection with machine learning classifiers.

Acknowledgements

A thank you to my wife for her encouragement and putting up with me (I mean the long hours). I get so focused sometimes that she truly brings me out of it. In fact, my mother would say, “if my head wasn't attached, I would leave it at home.”

Additionally, thanks to my adviser, Maj Mendenhall, for his guidance. Dr. Seuss provides a keen description, “We like our Mike and this is why: Mike does all the work when the hills get high.” [50] His insights for surmounting the “mental hills” have been invaluable in my coursework and thesis.

José Andrés González

Table of Contents

List of Figures

List of Tables

Numerical Analysis for Relevant Features in Intrusion Detection (NARFid)

I. Introduction

The new United States (US) Air Force mission statement expands the Air Force's role from the physical domains of air and space to include cyberspace. The adoption of computer networking as part of this mission domain highlights the dependencies of the US' physical war-making capabilities, national infrastructure, and economy on computer systems and communication technologies. The US national infrastructure and defense systems are placed at risk due to the low cost and minimal knowledge required to cause harm in the cyberspace domain. The ability to recognize threats is critical to the US Air Force's ability to find, fix, track, target and engage in cyberspace.

1.1 Background

The United States heavily relies on information technologies in the private and public sectors to maximize effectiveness through automation, communication, situational awareness, and decision support. The loss or degraded operation of commercial and government services due to cyber attacks could cause havoc on productivity, overall performance and safety, if critical infrastructures are affected. Entrenched usage of information technologies is core to the business community, the government, and a way of life for many. Even though the commercial sector has pushed hard for network and computer security improvements over the past decade, a virtual "arms race" between unethical hackers and security professionals has resulted with no expected end to the exploit cycle [79]. Without flawless systems with absolute security, novel methods of detecting and responding to attacks are required.

1.1.1 National Cyberspace Security Strategy.

The importance of cyberspace and information technologies illicits planning at all levels of government. At the highest level, the *National Strategy to Secure Cyberspace* [53] provides a framework to deter, prevent and respond to threats covering the physical and logical cyber infrastructure. The strategy emphasizes the need for cooperation among US government, other public entities, private institutions, international organizations, and governments to secure cyberspace. The Department of Homeland Security has a central role in facilitating this national strategy. The Department of Defense's (DoD's) role is to secure the defense industrial base, quickly attribute the source of cyber attacks, and develop capabilities to prevent attacks from reaching critical systems and infrastructures.

A foundational principle in deterring attacks in cyberspace is removing anonymity from the offender and holding them accountable. The Internet and most common operating systems were developed as open systems. Cyber attack techniques highlight these characteristics by obfuscating their origins so the source of a packet is not necessarily the source of the attack. The botnet used against Estonia is a prime example, where attacks came from all over world [65]. No state or other actor claimed responsibility or was deemed culpable. Implementing a secure cyberspace with accountability will require a concerted effort in research and development of software and operating systems, and security-oriented standard development. The *National Strategy to Secure Cyberspace* addresses all of these areas and relies heavily on market forces to correct these issues along with government research and development.

1.1.2 Network Centric Warfare.

Communication systems reside at the heart of the United States warfighting capability and ability to implement network centric warfare. The DoD strongly pursues the concept of network centric warfare since a "networked force" has been identified as a decisive factor to success [52]. The DoD intends to accelerate processes and provide data integration for seamless operations by optimizing members' capabilities with information technologies. The concept pro-

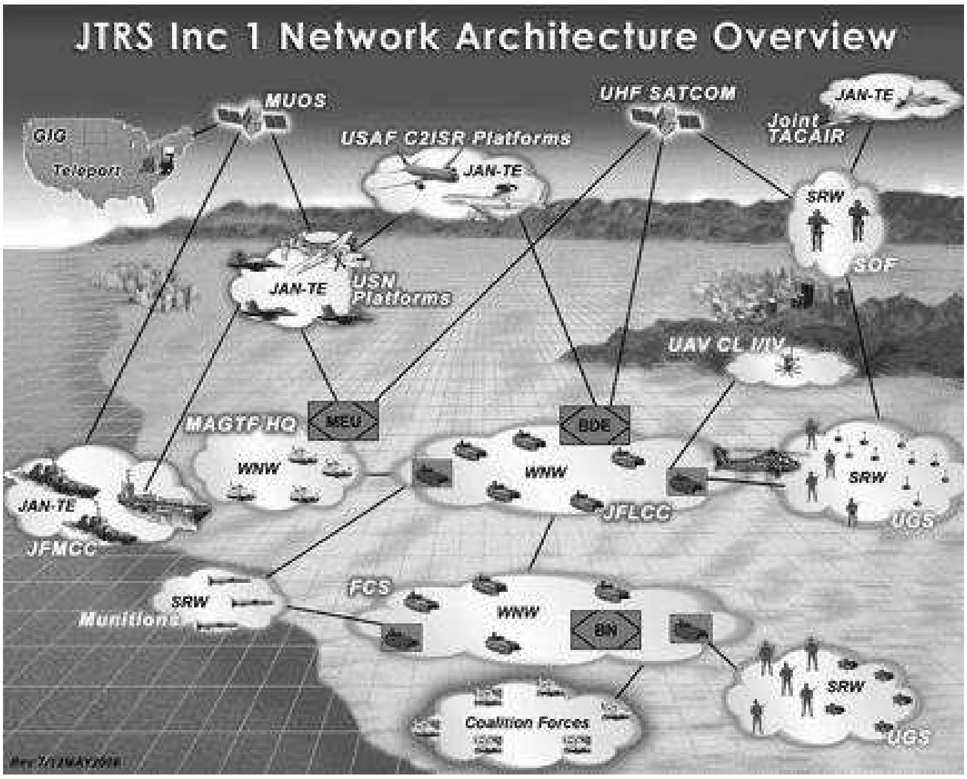


Figure 1.1: Joint Tactical Radio System Network: Denotes the extensive integration of DOD systems with network communication capabilities

vides an increased warfighting advantage through information sharing of situational awareness and commander's decisions. The warfighter then benefits from the speed of command, and increased lethality and survivability. The network centric operations concept will accelerate the deployment of networked forces and capabilities placing an even heavier burden on the DoD's communication systems. Network centric operations blanket the operational environment and are entwined with operation systems. For example, Figure 1.1 depicts the planned Joint Tactical Radio System, which will integrate operational and tactical systems to the DoD's Global Information Grid. Extensive developmental and operational test are performed on warfighting systems to mitigate vulnerabilities and operational flaws. Even so, the military's unclassified and classified network must be monitored for threats.

1.2 Problem Statement

Identification of cyber attacks and network services is a robust field of study in the machine learning community [1,2,5,11,18,20–23,25,27,39,42,44,45,48,49,54,58–62,64,66,72,77,80,81,85]. Less effort has been focused on understanding the domain space of real network data to identify important features for cyber attack and network service classification. Motivations for identifying the most relevant features allows for anomaly detection systems with fewer requirements on data "sniffed" off the network, extraction of features from the traffic, reduced learning time of algorithms, and ideally increased performance in accuracy.

For the purpose of intrustion detection, this thesis seeks to minimize the number of features while obtaining statistically-significant improvement in classification, and decipher the relationships and traits of the selected feature subsets. Feature selection of a high dimensional data may allow for improvement of a classifier's generalizability and comprehensibility, and data simplicity [46]. Generalizability of a classifier provides the ability of a classifier to model the data for the sake of improving classification accuracy. Predictive accuracy (or error) is often the metric for assessing generalizability; the higher the accuracy the greater the performance of the model on unseen data. Comprehensibility of a classifier entails understanding the relationships within data that support accurate classification. The simpler the model, the easier it is to understand the relationships generated between the features. Data simplicity pertains to the number of features and potential samples. In real world scenarios, features are often real numbers resulting in an infinite number of possible samples. In a contrived scenario provide by Liu and Motoda [46], one may have a maximum of two binary features that results in only four possible samples. Dimensionality reduction by selecting a subset of features and a discretization of the features provides for simpler models that require less learning and improved accuracy.

This thesis evaluates the usefulness of a good feature subset for the general classification task of identifying cyber attacks and network services. The generality of the selected features elucidates the relevance or irrelevance of the feature set for the classification task [46]. Feature selection utilizing a single classifier for verification may lead to a feature subset that only performs well for the implemented classifier. Classification performance across a varied group of classifiers demonstrates that the feature subset provides inherent classification advantage vice preferring the bias of a single classifier.

1.3 Scope

This research focuses on the ability to find cyber attacks over a network. Current methods for intrusion detection systems analyze network traffic predominantly by signature-based and policy-based methods. Current research of anomaly-based methods assesses various machine learning techniques in clustering and classifying network traffic. This thesis places a greater focus on developing and validating the features (e.g., statistics, metrics or other measures) of network traffic used to describe a network flow for identifying cyber attacks. The feature sets may be further optimized by determining a minimal set for clustering and classification. The thesis will pursue machine learning approaches to analyzing network traffic for the purpose of detecting anomalous behavior with network traffic. Machine learning is used in selecting, validating and decreasing the dimensionality of relevant features.

1.4 Organization

The following chapters discuss areas of machine learning and their application to the problem of intrusion detection by seeking a useful set of features to describe the decision boundary. Chapter II delves into the established research in the area of intrusion detection, classification methods and feature selection algorithms. Chapter III outlines a methodology for assessing useful features for the intrusion detection problem and recommends an extension to a current feature selection method. Chapter IV provides results of the experiments and analysis of why certain features may be beneficial. Chapter V summarizes this thesis' effort, details the contribution of this work, and recommends future work.

II. Background

Research in the area of intrusion detection appears to be in two “camps.” The first camp revalidates established results for the classification of network traffic, with little apparent extension to the field of study. For example, several works demonstrate the effectiveness of the multilayer perceptron to classify traffic types from monitored traffic [11, 22, 25, 44, 62, 64]. A majority of whom utilize the same data sets developed by a Defense Advanced Research Projects Agency (DARPA) study [27, 45]. Analysis of well performing classifiers for a given application is certainly warranted; there are numerous extensions to the multilayer perceptron in regards to intrusion detection (e.g., string handling and layered approaches [23, 49]). Redundant and gratuitous works, which profess similar conclusions, serve little utility in extending the current body of knowledge.

The second camp of research assesses novel methods in developing a framework for anomaly detection. It is this body of work that this thesis intends to extend by assessing relevant features for inclusion in an intrusion detection system. The Network and Operating Systems group at the University of Cambridge addresses important facets to the intrusion detection system. A key component from a research perspective is data for experimentation. The University of Cambridge group has developed a broad feature set as well as a test suite (see Section 4.1.1). Additionally, the group addresses multiple components of the intrusion detection system to include monitoring of data and detection methods [2, 58, 60]. The monitoring system is capable of examining network traffic at gigabit speeds and extracting features pertinent to classification off-line. The group has demonstrated extensions to established classification methods for intrusion detection, (e.g., multilayer perceptron and Näive Bayes classifiers), which is discussed further in Section 2.2.2. An area lacking is analysis of the features relevant to the intrusion detection problem. Identification of a relevant feature subset allows for the possible online extraction of those features for (near) real-time classification. This chapter continues by discussing the generic framework of intrusion detection and pertinent work in the field of machine learning for anomaly detection and feature selection.

2.1 Intrusion Detection

Intrusion Detection Systems (IDS) serve as a defensive tool against cyber attacks on a network, host, or application. A network-based IDS monitors network traffic over a single or multiple network nodes. A host-based IDS monitors a given machine's operating system, software, and files. An application-based IDS monitors a specific application-level system such as a database or accounting system. The basic framework of a typical IDS is illustrated by Figure 2.1. A given system is monitored and data is extracted for real-time or post analysis with an IDS. The core function of the IDS identifies malicious behavior within the monitored system or network. Pending on the IDS type, responses may vary from blocking ports, quarantining files, or revoking user access. The malicious behavior may have an internal or external source. An internal bad actor is a legitimate user or process performing a disallowed operation. An unauthorized user or process gaining access and executing commands constitutes an external bad actor. Data gathering methodologies, detection engines, and responses may function irrespective of source or be tailored to address a given system threat. As discussed, an IDS contains several components, but much of the difficulty resides in the central component, the detector. There are two methodologies for implementing a detector within an IDS: anomaly and signature detection.

Anomaly detection annotates outliers, often termed abnormal, from all traffic. The identified outliers are expected to be indicative of malicious traffic. This methodology is stigmatized with having large false alarm rates [3, 39]. In general, anomaly detection may be implemented via various methods like agents, expert systems and machine learning techniques. This research address methods for feature selection and classification pertinent to anomaly detection. These methods are directly applicable to other domains that contain high dimensional data requiring classification, regres-

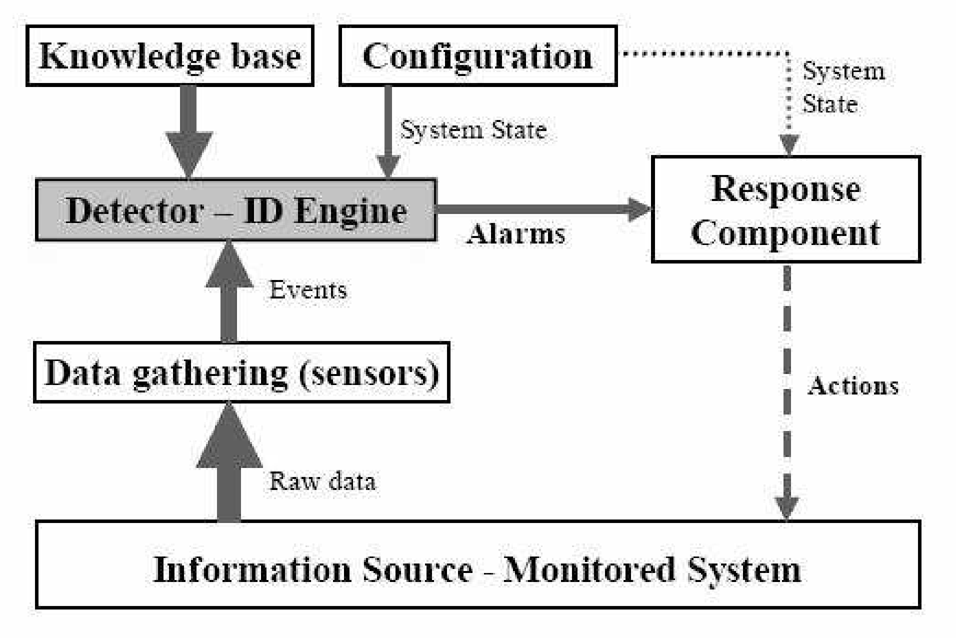


Figure 2.1: This is an architecture of a typical intrusion detection system from [40]

sion, or clustering. Previous work on the techniques and challenges for network traffic classification are discussed in Section 2.2.

Signature detection (commonly known as misuse detection) may utilize pattern matching techniques and/or rule-based models to compare observed traffic with an established "library" of known attacks. It provides a high confidence in detection since the attacks are known and described in detail in a library. Issues arise with these types of systems due to the lag in adding new signatures to a given library or missed variations of known attacks.

Although both methodologies fundamentally support the same capability, the two techniques address the problem of intrusion detection from opposite angles. Signature-based detection uses the attributes of all possible (known) attacks to find an attack. Where anomaly-based detection baselines the operation of the system (to include network) and determines attacks based on variations from the baseline. This description makes certain assumptions to imply the effectiveness of a given approach. Signature-based detection requires knowledge of all possible attacks to always be successful and anomaly-based detection requires in-depth knowledge of how a system is suppose to "behave." How to define the behavior of a system has significant complexity. A signature-based approach relies on maintaining an up-to-date file of the malicious software. Anomaly-based detection relies on updates of "appropriate" behavior of the system to assess changes and recognize the actions of malicious software. The output of a signature-based system is deterministic. A given file or collection of network traffic must match a signature of malicious code to illicit a detection. Many anomaly-based detection methods utilize stochastic or statistical approaches. The detection bases its conclusion on probabilistic evidence of whether a behavior is malicious or not, which may require further assessment. Since anomaly detection is not limited to a set of identified malicious code, it will have the capability to find new cyber attacks with a degree of certainty (or uncertainty pending your perspective).

2.2 Related Work

There is a large body of research into the use of machine learning algorithms for traffic classification and identification of network attacks for anomaly detection. Traditionally, well-known port numbers are used to distinguish types of network traffic. Of course, this approach is not feasible for cyber attacks, but many applications obscure their high-level application by using a well-known port of an alternate application. This is common among peer-to-peer networks. Vice solely relying on port numbers, supervised and unsupervised techniques utilize statistical trends among like flows to perform classification or clustering. Claffy and Paxson [15, 67] illustrate the separation of network traffic volume distributions and transmitted byte statistics that make machine learning techniques plausible from data collected with a network "sniffer." Specifically, flow duration and packet volume distinguished Domain Name System packets from other types for network traffic.

2.2.1 Feature Selection. Dash and Liu define feature selection as selecting the *minimally* sized subset of features, which does not significantly reduce classification accuracy and maintains the original class distribution given all of the features [17]. The adoption of a large number of "ad hoc" features for real-world classification problems is to be expected, since an adequate understanding of the domain is unlikely. Features may be redundant, uninformative, or even distractors for a given application. Features may have no relevance alone, but together with other features they may provide synergies for significant performance gains [26]. An ideal classification process would involve only relevant features to minimize running time, improve learning performance, and illustrate understanding of the specific classification problem.

Feature selection is an intractable, exponentially hard, $\mathrm{O}(2^N)$, search problem where N is the number of features and $2^N - 1$ is the number of feature subsets, ignoring the null set, to be searched for determining optimal performance [47]. The number of combinations with k elements drawn from a set of n elements [57] may be calculated with

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}. \tag{2.1}$$

The number of subset combinations to be searched is proved with the binomial theorem, when x and y are equal to 1, as follows:

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^{n-k} y^k$$

$$(1+1)^n = \sum_{k=0}^{n} \binom{n}{k} 1^{n-k} 1^k$$

$$2^n = \sum_{k=0}^{n} \binom{n}{k}. \tag{2.2}$$

By ignoring the null set ($k = 0$), the precise search space may be calculated:

$$2^n = \sum_{k=1}^{n} \binom{n}{k} + \binom{n}{0}$$

$$2^n - 1 = \sum_{k=1}^{n} \binom{n}{k}. \tag{2.3}$$

There are two major components to the optimization task of feature subset selection for classification, as with other optimization problems [17]. First, a *generation method* traverses subsets utilizing tree-based, stochastic, or other search methods. Second, an *evaluation measure* estimates the (expected) performance of a given subset.

Generation methods are search methods of the feature subset space. In choosing a search method, one would want a complete search that is guaranteed a solution (e.g., breadth-first search). Whether the solution is optimal or not for a given evaluation measure would be beneficial. Search methods take advantage of costs, heuristics and stochasticity in pruning the space. Running times are significantly less than if one performs an exhaustive search of every possible combination. Evaluating features individually or using a heuristic reduces the search problem time complexity to O(N^2) or less [17]. For example, a heuristic may be used to reduce the search space by greedily selecting features so the space is reduced by one each iteration. In this case, the search space is O(N^2) since the number of features to search goes from N to one and is calculated by $N + ... + 2 + 1 = N(N-1)/2$. Additionally, the use of a classifier for determining the *evaluation measure* would hinder the running time by the complexity of the classification technique, which may be significant for iterative methods.

Evaluation measures provide a cost for the performance optimization of a feature or feature subset in classification. Measures are based on distance, information theory, dependency, consistency, or classifier accuracy rate. Distance, information theory, dependency, and consistency measures allow for broad application of a selected feature set since the algorithms depend on intrinsic characteristics and relationships of the features, including class label at times [17]. Utilizing classification as the *evaluation measure* provides a direct performance measure of the pertinent task on a representative test set. Without assessing the actual performance of the feature subset, one would not be able to determine any synergies from grouping certain subsets of features together. It is not necessarily true that the best-performing feature set for a given classifier is universally useful for all classification techniques.

Blum and Langley [7] categorize feature selection methods into three camps: embedded, filter, and wrapper. An embedded method performs feature selection as part of a learning algorithm. A filter method removes irrelevant features utilizing internal characteristics of the features, vice an external learning algorithm to quantify performance. Lastly, a wrapper method utilizes a learning algorithm to evaluate subsets in the search space. Feature selection methods may contain characteristics of multiple groupings in Blum's [7] and/or Dash's [17] taxonomy.

Specifically for the intrusion detection problem, Lunt [48] attests to the lack of *a priori* knowledge on the effectiveness of a given set of features for classification. For some classifiers like Näive Bayes, redundant and irrelevant features chosen due to intuition may significantly encumber network traffic classification performance [60]. Predominantly, discussion of feature selection algorithms for the purpose of intrusion detection focus on increasing the performance of a given classifier with a reduced subset [13, 14, 80]. For example, Auld [2] and Zainal [89] compare the constituents of selected subsets on differing data sets to illustrate consistency among selected features. They demonstrate high classification rates after feature selection of characteristic discriminators that do not include reliance on port numbers. On the other hand, Moore [60] discusses the separation between classes in feature space as an important feature characteristic for classification. To illustrate the point, Figure 2.2 shows a high degree of separability among attack and WWW Transmission Control Protocol (TCP) connections using the mean bytes in an Ethernet packet (feature 177 from the experimental data set discussed in Section 4.1.1) as a descriptor of the TCP connection.

2.2.1.1 Bhattacharyya Method. Utschick [84] discusses feature selection based on the separability between classes for a given feature for a multiple-class classification problem. The Bhattacharyya coefficient is used as a measure of the sep-

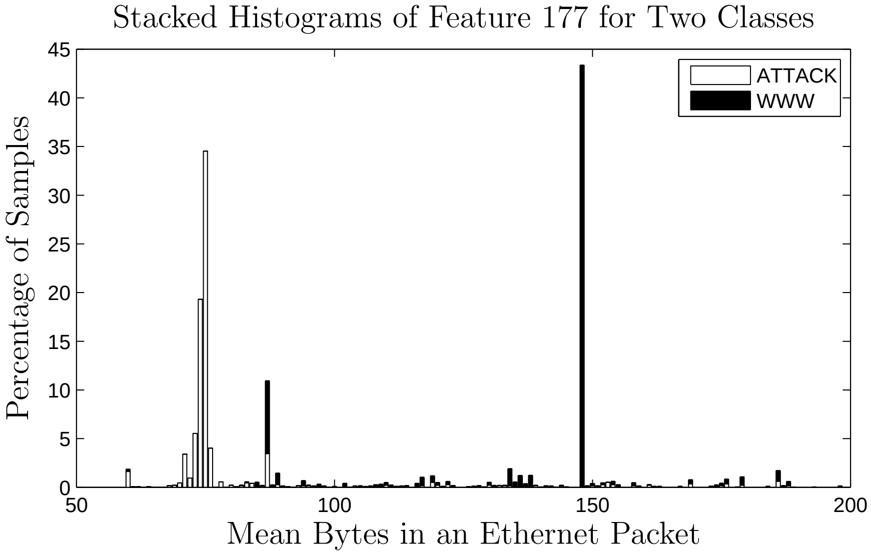


Figure 2.2: The plot illustrates histograms of feature 177 from the perspective of two classes. The histograms demonstrate the separability between attack and WWW traffic

arability of two classes, a and b, for a given feature f. The Bhattacharyya coefficient for a given feature, B_f, is calculated by

$$B_f = \sum_{i}^{k} \sqrt{p_i^a p_i^b}, \tag{2.4}$$

where there are k bins of the data, and p_i represents the probability (or contribution) of a bin in a feature's histogram in respect to samples of a given class, a or b. The Bhattacharyya coefficient ranges from [0,1], where a value of one indicates that the two distributions are identical and zero indicates absolutely no overlap of the distributions. The mean Bhattacharyya coefficient is used as the measure for comparing features among multiple classes, where there are $C(C-1)/2$ possible pairs of C classes and the smallest mean is the most important feature. Sorting by the mean is analogous to sorting by the median Bhattacharyya coefficient, since both represent differing perspectives of the "center" of a sample set. Correlation is shown between the predictive accuracy of a classifier and the Bhattacharyya coefficient of the features used in training. Feature selection by way of mean Bhattacharyya coefficient is extended by

Benediktsson [4] to consider feature selection based on a weighted average of separability if one seeks to optimize the classifier for a given class or set of classes. Utschick and Benediktsson [4,84] assume the distributions of the features among the classes are Gaussian in order to utilize the Bhattacharyya coefficient. Thacher, *et al.* [82] show the Gaussian assumption need not be made; the Bhattacharyya coefficient may be used as a measure for a data set with any distribution of the features. Additionally, Thacher argues that the Bhattacharyya coefficient is an absolute similarity metric, vice a measure of relative separation.

Utilizing the Bhattacharyya coefficient results in a O(S) runtime in respect to the number of samples (S). This is due to the fact that there is a constant number of passes through all the data for the sake of generating counts and binning the class histogram of the feature. The runtime in respect to the number of features (N) would still be O($NlogN$) due to the sorting of the features by Bhattacharyya value. For the purposes of this thesis, this feature selection approach is referred to as the mean or median Bhattacharyya method. The runtime performance for this method is on the low end in comparison to the other feature selection methods discussed. The sorting of features by the mean Bhattacharyya coefficient does not contribute to the worst case runtime performance as long as the number of samples is significantly greater than the number of features.

2.2.2 Classification. Laskov [42] shows that supervised algorithms greatly outperform unsupervised algorithms for "known" attacks. For "unknown" attacks, both techniques perform equally poor. Performance of the unsupervised algorithms are consistently below 80% accuracy. For supervised learning, Laskov defines an attack as "known" if it exists in the test and training sets and "unknown" if it exists only in the test set. Unsupervised algorithms are not typically trained with class label information so only the test sets are used. There is a clear advantage for favoring supervised learning algorithms for this application. However, the effort required in labeling data and the uncertainty in the true label for training makes supervised techniques laborious and potentially unreliable. Extensive preparation of training data may be necessary to accurately portray the classification of network flows. A reliable mechanism of clustering network flows for classification would provide a step in the direction for automated training of a supervised classification method. Unsupervised clustering algorithms are successfully adapted as classifiers with some papers reporting accuracies in the 80 - 90% range [5, 20, 90]. This thesis deals with supervised learning techniques for the purpose of feature selection.

The multilayer perceptron with backpropagation (BP) provides the capability to analyze traffic in a non-linear construct and handle incomplete or noisy data. These characteristics are directly applicable to a real computer network, which will have failures, lost packets, and complex relations among flow statistics and a flow's behavior. Furthermore, the feedforward through a BP for classification is extremely fast and of benefit for this application with massive volumes of flows. As more information on a specific site's traffic is known, a BP may be retrained to provide increased accuracy. Of most interest, BPs may correctly identify previously unseen attacks, where a signature-based method would not. BPs relax the need for accurate modeling of statistical distributions of a given metric. Furthermore, BPs have the advantage of learning and performing well with a host of features, which may not be important to the task, by minimizing their contribution to reduce error. Numerous works [2, 11, 18, 22, 25, 44, 62, 64, 81] demonstrate the effectiveness of BPs on the classification task to identity network services and cyber attacks. A majority of which succeed at detection rates of 90% with some reaching an unbiased estimate of 99%.

Despite its success, the BP has its share of problems. Training a BP often requires a significant amount of data to provide adequate representation of the domain. Unlike signature-based methods that use expert systems and rule-based approaches, a BP serves as a "black box" and does not provide an intuitive or meaningful structure for imparting the knowledge learned aside from the achieved output [31]. Additionally, a BP biases the output classification to a given class if it constitutes a significantly large portion of the training set. A BP learns the given class with more training steps than alternate classes causing the bias [2].

The Näive Bayes classifier with Gaussian distribution is another common classification model used for exploring the problem of identifying network services and cyber attacks. Moore and John [33, 60] illustrates the classification potential of a variant of the Näive Bayes classifier utilizing kernels, while highlighting the deficiency of the algorithm's assumption. For complex data sets, the features are likely not independent. When modeling the distribution of the features utilizing kernels, the classification performance is significantly increased compared to using a Näive Bayes classifier assuming a Gaussian distribution. This may be explained by features with multi-modal distributions. Nguyen [66] demonstrates the capacity of the Näive Bayes classifier to statistically model flows and identify network services with a sliding window. The sliding window only captures a portion of the flow so classification occurs with missing data and accounts for real-time changes in traffic over a network. Specific to intrusion detection, the Näive Bayes classifier provides ample classification performance even with a small set of features in detecting Internet worms [1].

2.3 Machine Learning Algorithms

This section discusses methods for classification and feature selection, which are further investigated in this work. Most methods exclusively function as a classification or feature selection algorithm. Two algorithms, C4.5 decision tree with pruning and Generalized Relevance Learning Vector Quantization Improved (GRLVQI), perform both functions in the development of the classifier. The classification algorithms consist of the following:

1. C4.5 Decision Tree
2. Näive Bayes Classifier
3. Multilayer Perceptron with backpropagation (BP)
4. Generalized Relevance Learning Vector Quantization Improved (GRLVQI)

C4.5 decision tree develops a decision tree that segments the domain space based on entropy-derived feature values. The Näive Bayes classifier, in this thesis, models classes as joint probability distribution functions (PDFs) as the product of Gaussian distributions. Each dimension is assumed Gaussian and independent. The estimates of the joint PDFs, the likelihood of the class, and a maximum likelihood are used in the classification process. The BP utilizes a directed graph that performs a nonlinear combination of the features to model a class. Lastly, GRLVQI uses prototype vectors to represent the boundary of the modeled classes.

This research delves into various feature selection algorithms in assessing a working set of features for intrusion detection. All of these algorithms are capable of handling continuous, discrete and nominal data and capable of dealing with large data sets [17]. The possible combinations of feature selection generation and measures are substantial. This survey of feature selection methods covers the following algorithms:

1. RELIEF-F
2. Probability of Error and Average Correlation Coefficient (POEACC)
3. Decision Tree Method
4. Generalized Relevance Learning Vector Quantization Improved (GRLVQI)
5. Classifier Accuracy Rate and various search methods

RELIEF-F provides a feature specific weighting based on a distance within the given dimension. Probability of Error and Average Correlation Coefficient provides a feature specific ranking based on dependency and classifier performance of a given feature. The Decision Tree Method utilizes the C4.5 decision tree with pruning to generate a subset based on information theory and probability of error. GRLVQI utilizes a distance-based approach for a feature relevance, which is iteratively updated with each training step of generating the classifier. The final feature selection methods utilize the classifier accuracy rate evaluation measure with a genetic algorithm and best first search to find a well performing feature subset.

```
Client Port (2) <= 20: FTP-DATA
Client Port (2) > 20
|    Server Port (1) <= 53
|    |    Server Port (1) <= 23
|    |    |    Server Port (1) <= 21: FTP-CONTROL
|    |    |    Server Port (1) > 21: INTERACTIVE
|    |    Server Port (1) > 23
|    |    |    Server Port (1) <= 25: MAIL
|    |    |    Server Port (1) > 25: SERVICES
|    Server Port (1) > 53
|    |    Server Port (1) <= 80
|    |    |    RTT avg a b (119) <= 0.213446
|    |    |    |    max data wire (15) <= 251
|    |    |    |    |    med data wire b a (176) <= 60.5
|    |    |    |    |    |    FFT a b (Frequency #4) (232) <= 0.003286
|    |    |    |    |    |    |    RTT samples a b (113) <= 1: WWW
|    |    |    |    |    |    |    RTT samples a b (113) > 1: ATTACK
|    |    |    |    |    |    FFT a b (Frequency #4) (232) > 0.003286: WWW
|    |    |    |    |    med data wire b a (176) > 60.5: P2P
|    |    |    |    max data wire (15) > 251
|    |    |    |    |    zero win adv a b (91) <= 0
|    |    |    |    |    |    RTT min a b (115) <= 0.002001
|    |    |    |    |    |    |    avg segm size a b (85) <= 150
|    |    |    |    |    |    |    |    mean data ip (20) <= 603.75: WWW
|    |    |    |    |    |    |    |    mean data ip (20) > 603.75
|    |    |    |    |    |    |    |    |    triple dupacks b a (142) <= 0: ATTACK
|    |    |    |    |    |    |    |    |    triple dupacks b a (142) > 0: WWW
tree continues...
```

Figure 2.3: This is an example decision tree generated from the test set discussed in Section 4.1.1

2.3.1 C4.5 Decision Tree. A tree functions as the classifier, where the nodes represent features for a decision and the edges pertain to values or conditions for all possible outcomes. The algorithm intends to provide a classification capability by uncovering "the structure of the domain" [70]. Figure 2.3 illustrates an example decision tree generated by the data set discussed in Section 4.1.1. A sample is classified by following a path from the root to a leaf, where the leaf node represents the class. The tree is generated by dividing a sample set into two or more partitions based on a set of mutually exclusive outcomes for a decision node. A decision node constitutes a test on a selected feature. This process iteratively splices the partitions to develop additional decision nodes until each partition contains a single class or no further improvement may be made. The algorithm performs a greedy selection of tests for decision nodes, based on entropy gain. Extensions to the algorithm utilize alternate evaluation functions (e.g., Gini index, misclassification rate) [9]. For each decision node, tests are examined on the entire feature set resulting in multiple decision nodes with a given feature, but distinct tests. For a nominal feature, the test constitutes an edge for each possible value of the feature or a subset of distinct values may define an edge. For a continuous feature, a demarcating value is selected to split the set into 2 partitions.

To select a feature and test for a decision node, the gain ratio for entropy of the partitions is maximized. The standard form of entropy in bits (i.e., *expected value of information* in bits), H, is

$$H = -\sum_{i=1}^{n} p_i \times \log_2 p_i, \tag{2.5}$$

where there are n values, and p_i is the probability of a given event. Entropy may also be calculated by

$$H = \sum_{i=1}^{n} p_i \times I, \tag{2.6}$$

where $I = -\log_2 p_i$ is the amount of *information* in a given event. Entropy, H, is used multiple times in the splitting process to assess the amount of information in a set based on three perspectives. The three perspectives of a set's entropy are in regards to the classes in the set, the classes in partitions of the set and the size of the set's partitions (e.g., $info(S)$, $info_X(S)$, and $split\,info(\mathbf{X})$, respectively).

The entropy of the training set or a partitioned subset, S, in regards to separating classes is determined by

$$info(S) = -\sum_{j=1}^{k} \left[\frac{freq(C_j, S)}{|S|} \times \log_2 \frac{freq(C_j, S)}{|S|} \right], \tag{2.7}$$

where there are k classes, C_j is a given class, $freq(C_j, S)$ is the count of the given class in set S, and $|S|$ is the cardinality of the set. Entropy of the classes in partitions for a given test, X, on a set is determined by

$$info_X(S) = \sum_{i=1}^{n} \left[\frac{|S_i|}{|S|} \times info(S_i) \right], \tag{2.8}$$

where there are n partitions and S_i is a given partition. The information gain of the test, X, is calculated as

$$gain(X) = info(S) - info_X(S). \tag{2.9}$$

The gain must be normalized to prevent situations where a single uninformative feature causes partitions consisting of a single sample or very small sets. Partitions of a single sample result in an $info_X(S) = 0$ so the gain would be maximized, but the classifier would provide no utility. This situation is alleviated by determining the entropy of a set in regards to partition size for a given test as

$$split\,info(\mathbf{X}) = -\sum_{i=1}^{n} \left[\frac{|S_i|}{|S|} \times \log_2 \frac{|S_i|}{|S|} \right]. \tag{2.10}$$

In Equation 2.10, $split\,info(\mathbf{X})$ approaches zero for uninformative splits with small numbers of samples in the partitions. By maximizing the gain ratio,

$$gain\,ratio(\mathbf{X}) = gain(\mathbf{X}) / split\,info(\mathbf{X}), \tag{2.11}$$

one will be able to maximize the information gain of a given test, while preventing uninformative splits. An additional constraint may restrict partitions to “reasonable” number of samples (like two) to avoid trivial splits of individual partitions, where the *gain ratio* would be acceptable. As mentioned earlier, the C4.5 algorithm divides subsets with new tests until a given set has one class or the tests do not improve the gain ratio significantly. The class of the leaf is determined by the mode of samples class labels pertaining to the given path. The resulting decision tree “over fits the data” [70] so the C4.5 algorithm then prunes the tree of decision nodes that do not facilitate classification. Each time the tree is built, it is pruned by a holdout set within the training set. The holdout set is treated as a random sample set for which the probability of error may be estimated for the whole population of potential samples. The number of errors is modeled as a binomial distribution. The binomial distribution is appropriate due to a set number of samples (N) and the errors are independent. Hence, the occurrence of an error has no impact on any other error. The probability of error is calculated by

$$p = \frac{e + 0.5}{N}, \tag{2.12}$$

where N is number of holdout samples and e is the number of misclassified samples. The C4.5 method assumes the single observation of errors represents the mean of the binomial distribution of errors for a given branch. The addition of 0.5 to the assumed mean of errors is a “half-unit correction for continuity” to improve the approximation of the binomial distribution as a Normal distribution [57, 78]. The Normal approximation of the binomial distribution is used to estimate the upper confidence limit at a given confidence level of the probability of error. The Normal approximation may not be technically valid for a small sample size, but it is used as an estimation of error. By using the Wilson Score interval [86], the probability of the confidence limit is determined by

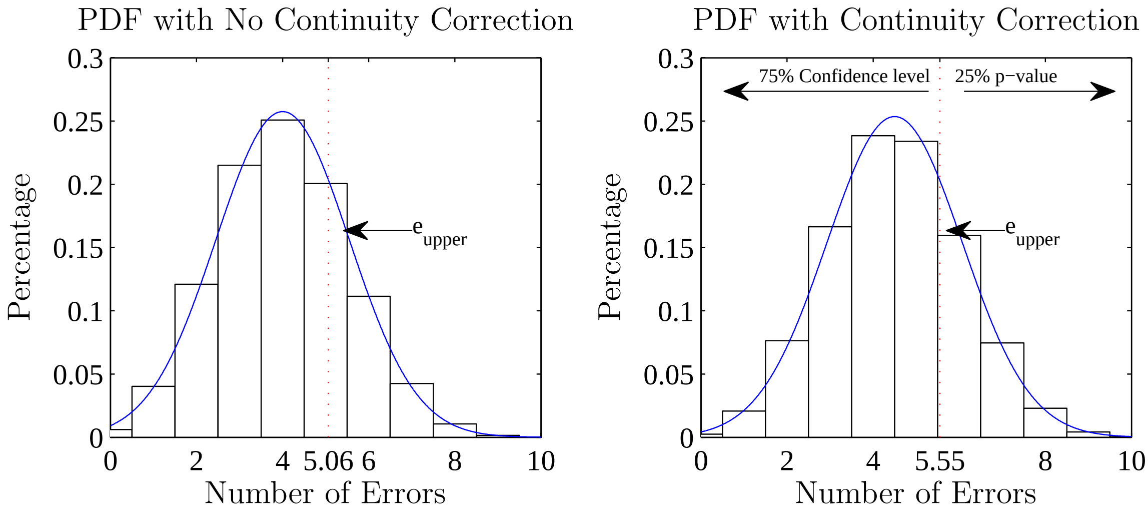
$$p_{upper} = \frac{p + \frac{z_\alpha^2}{2N} + z_\alpha \sqrt{\frac{p}{N} - \frac{p^2}{N} + \frac{z_\alpha^2}{4N^2}}}{1 + \frac{z_\alpha^2}{N}}, \tag{2.13}$$

where z_α is the $100 \times (1 - \alpha)^t h$ percentile of the Normal distribution for the one-sided confidence limit. Quinlan [70] notes that the default confidence level is a nonsensical 25%. One-sided confidence levels less than 50% account for occurrences in a tail of the Normal distribution that goes out towards infinity (i.e., not inclusive of the mean) and two-sided confidence levels that low would have no meaningful interpretation. Quinlan is actually referring to a p-value, α, of 0.25 and with a one-sided confidence level of 75%. The 75th percentile of the Normal distribution, $z_{0.25}$, is 0.67. Figure 2.4 (b) illustrates the discussed interpretation of the confidence level. The upper confidence limit of the number of predicted errors is calculated with

$$e_{upper} = p_{upper} * N. \tag{2.14}$$

To illustrate the error estimate of the upper confidence limit, Figure 2.4 shows two plots of a binomial distribution (with $p = 0.4$ and $N = 10$) overlayed with the corresponding Normal approximation and a horizontal line to denote the upper confidence limit. Without the correction for continuity, Figure 2.4 (a) underestimates the upper confidence limit of error (5.06) [69]. The Normal approximation does not account for the entire bin with the mean (4) in the lower 50% of the distribution causing a smaller upper confidence limit. Figure 2.4 (b) illustrates a upper confidence limit of 5.55 by shifting the distributions to the right with the correction for continuity and provides a more accurate estimate of error [69]. By beginning at the bottom of the tree and traversing up, each subtree is evaluated to determine if replacement with a potential leaf or branch reduces e_{upper}. By replacing with minimal predicted errors, the tree theoretically minimizes the error rate. The C4.5 decision tree is an extension of the algorithm, known as ID3, which preprunes the tree by stopping branch growth for features that do not have a statistically-significant association with a class and utilizes the *gain* for selecting decision nodes [68].

The Decision Tree method (DTM) utilizes the pruned C4.5 decision tree discussed as a classifier to select relevant features. Once the pruned tree has been de-



(a) (b)

Figure 2.4: The Probability Density Functions (PDFs) illustrate the difference in the upper confidence limit of the error (e_{upper}) pending whether the correction of continuity is used or not. The confidence level for the upper limit is 75%

veloped, every feature remaining in the tree is chosen for the subset of features. This method illustrates an embedded approach to feature selection. Cardie [12] shows the success of using DTM with k-nearest neighbor for reducing the feature set and selecting matches for natural language processing.

2.3.2 Näive Bayes Classifier. Näive Bayes classifier uses Bayes theorem to determine the conditional probability of a class given a set of features [33]. The algorithm makes a couple of simplifying assumptions. All of the features are assumed to be independent and there are no hidden attributes that affect the classification of a sample. The largest posterior identifies the class of the sample. One determines the posterior of a sample using Bayes theorem,

$$p(C = c|\mathbf{X} = \mathbf{x}) = \frac{p(C = c)p(\mathbf{X} = \mathbf{x}|C = c)}{p(\mathbf{X} = \mathbf{x})}, \tag{2.15}$$

where C and $\mathbf{X}$ are the random variables for the class and the sample vector, respectively. The prior, $p(C = c)$, may be estimated by counting the occurrences in the training set or by assuming equiprobable classes. With the independence assumption, the likelihood of the sample vector is the product of the likelihood of all of the features,

$$p(\mathbf{X} = \mathbf{x}|C = c) = \prod_i p(\mathbf{X_i} = \mathbf{x_i}|C = c), \tag{2.16}$$

where X_i is the random variable for a given feature in the sample vector and there are i features. For a numeric feature, the likelihood of the given feature, $p(\mathbf{X_i} = \mathbf{x_i}|C = c)$, is estimated by determining the unknown parameters of a given distribution. Typically, Näive Bayes uses the Gaussian distribution,

$$p(\mathbf{X_i} = \mathbf{x_i}|C = c) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}, \tag{2.17}$$

in estimating the likelihood. The unknown parameters for a Gaussian are the mean, μ, and standard deviation, σ. For a nominal feature, the likelihood is the probability of the discrete attribute, X_i given a class. The normalizing constant may be calculated with

$$p(\mathbf{X} = \mathbf{x}) = \sum_j p(C_j)p(\mathbf{X} = \mathbf{x}|C_j), \tag{2.18}$$

where C_j is the class label and j are the class indices [85]. The normalizing constant does not need to be calculated since one only needs to normalize the numerator of the posterior for all the classes to obtain the posterior for each class.

2.3.3 Multilayer Perceptron. A multilayer perceptron with backpropagation (BP) is composed of multiple layers of processing elements (PEs) that emulate neurons to provide a nonlinear model for classification [6, 73, 75]. Each PE takes in as input a linear combination of the outputs of every node from the previous layer and an associated weight. The product of input and weight pairs are summed and manipulated with an activation function to generate the output to the next layer. Typical activation functions include the sigmoid and hyperbolic tangent. The BP will have an input layer, zero or more hidden layers, and an output layer in a directed graph, which is fully connected from one layer to the next. Figure 2.5 illustrates the conceptual

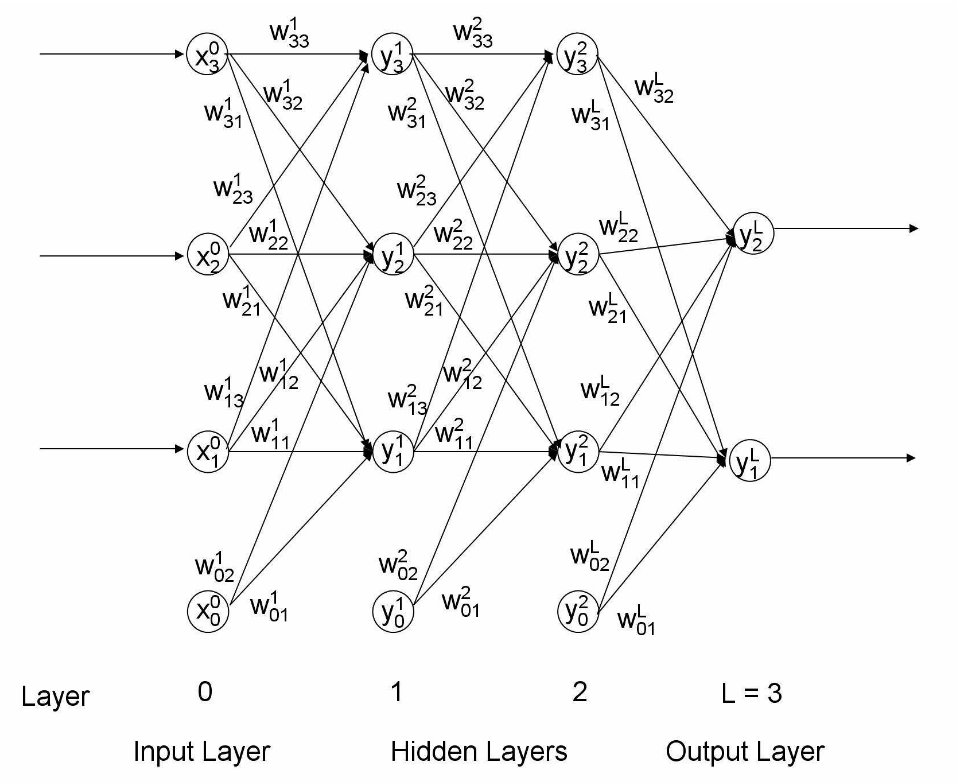


Figure 2.5: Pictured here is a conceptual layout of a multilayer perceptron with three input nodes (x_i^0), two hidden layers of processing elements (PEs) and two output PEs for classification. The input nodes (x_i^0) and PEs (y_i^j) are fully connected from one layer to the next. The variables i and j indicate a reference label and layer, respectively

layout of a BP with one input layer, two hidden layers and one output layer. The nodes in the input layer consist of a single feature from a sample, so the samples fed into the BP example (Figure 2.5) would only have three features. Each layer after the input layer consists of nodes that function as PEs. Hidden layers lie between the input and output layers. They add non-linearity to a decision boundary (classifier) or when representing data (regression). The output layer is simply the final layer of PEs, whose outputs are used for classification. The network learns from labeled samples and encodes the output as a bit vector with a single high (e.g., 001, 010, and 100) denoting the classification. The learning process occurs via backpropogation through the network to update the edge weights based on the error in classification generated

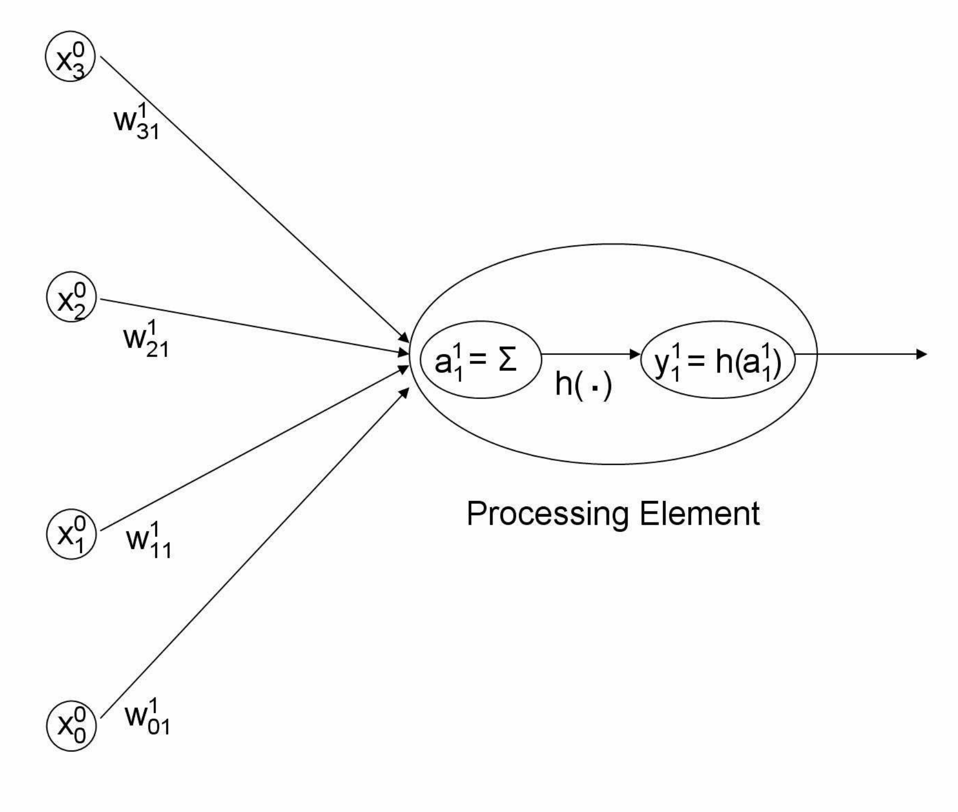


Figure 2.6: The processing element (PE) is an example from the first hidden layer in Figure 2.5. The PE calculates the linear combination (a_1^1) from its inputs (x_i^0) to include a bias (x_0^0) and their respective weights (w_{i1}^0). An activation function (h($\cdot$)) is applied to a_1^1 to generate the PE's output (y_1^1). The variables i indicates a reference label for the nodes

for a given input sample. This type of neural network may get large with increased input features, nodes per hidden layer and the number of hidden layers.

As illustrated by Figure 2.6, each PE consists of two parts [73]. The first part of a PE calculates the linear combination (a_k^l) of M inputs for the kth node in the lth layer by

$$a_k^l = \sum_{j=0}^{M} w_{kj}^l \, y_j^{l-1}. \tag{2.19}$$

The weight, w_{kj}^l, is for layer, l, and pertains to the input to node k from node j of the preceding layer. The output (y_j^{l-1}) from node, j, in the preceding layer, $l-1$ is an input for the PEs in layer, l. As illustrated in Figure 2.6, the outputs of the nodes in the input layer are referred to as x_j^0, vice y_j^0. A bias, w_{k0}^l, for node k in layer l is inserted to allow for shifts in the linear combination. To allow for such shifts, y_0^{l-1} is always equal to one. In the second part of a PE, the linear combination, a_k^l, is then transformed with an activation function to obtain the output of the given PE as shown by

$$y_k^l = h(a_k^l). \tag{2.20}$$

The use of a given activation function, $h(\cdot)$, may require scaling or normalization of the input sample's features to obtain the desired performance. The algorithm relies on nonlinear activation functions since linear combinations of linear activation functions are strictly linear. There would be no benefit in adding hidden layers. Propagating a given sample through the BP network constitutes the feedforward process that is used for classification. Often, classification is performed by having a single output node for every class, where the desired response is a one for the output PE corresponding to the class of interest and all the other class output PEs have a response of zero.

Training the network involves propagating the classification error back through the network to adjust the weights between nodes. To perform the backpropagation process, the nonlinear activation function must be differentiable so $h'(x)$ exists. First, one calculates the errors, δ_k^l, of the output layer, L using

$$\delta_k^L = (T_k - y_k^L)\, h'(a_k^L), \tag{2.21}$$

where y_k^L is the result of the feedforward process and output of the last layer, L, and T_k equals the expected value of the output node (e.g., zero or one). The weights of a layer may then be adjusted by

$$w_{kj}^l = w_{kj}^l + \alpha\, \delta w_{kj}^l. \tag{2.22}$$

The weight update allows for a learn rate, α, and accounts for the error in the edge weight, δw_{kj}^l. The edge weight is determined by the node's error in the succeeding layer, δ_k^l and the output from the node in the preceding layer, y_j^{l-1} by

$$\delta w_{kj}^l = \delta_k^l \, y_j^{l-1}. \tag{2.23}$$

The node's error in the preceding layer, δ_k^l, is calculated with

$$\delta_k^l = h'(a_k^l) \sum_z w_{zk}^{l+1} \, \delta_z^{l+1}, \tag{2.24}$$

where w_{zk}^{l+1} is the weight of the edge from node k in the preceding layer to node z in succeeding layer $l+1$, and δ_z^{l+1} is the error of the succeeding node, z. One recursively calculates the weights and errors of nodes in preceding layers until the first layer of PEs is updated. This results in an update of all of the weights within the BP to allow for training to a desired accuracy or within a given training time.

2.3.4 Generalized Relevance Learning Vector Quantization Improved. Generalized Relevance Learning Vector Quantization Improved (GRLVQI) models the boundary among classes with prototypes [56]. This version of learning vector quantization implements the concepts of in-class conditional update [56], relevance of features [29], conscience learning [19], and the LVQ2.1 prototype update strategy [37]. Classification of a given sample is determined by assigning the class label of the closest prototype. A set number of prototypes are associated to each class and are randomly initialized. Each feature has a relevance, λ_i, which is initialized to the reciprocal of the number of features and lies within the range [0,1], where larger values have a greater relevance. For each training sample, an in-class and out-of-class winning prototype is found by determining the closest prototypes in relevance-weighted squared Euclidean distance. The relevance-weighted squared Euclidean distance is

$$d = \sum_{features} \lambda_i (x_i - w_i)^2, \tag{2.25}$$

where x_i is the value of a given sample's feature and w_i is the value of a given prototypes feature. The prototypes are updated based on the in-class conditional update rule, where only the in-class prototype is updated if the sample is classified correctly. Otherwise, both the in-class and out-of-class prototypes are updated. The in-class (w_i^J) and out-of-class (w_i^K) prototypes of a feature i are updated by a gradient descent from a given time-step, t, by

$$w_i^J(t+1) = w_i^J(t) + \frac{4\,\epsilon^J(t)\,d^K\,f'|_{\mu(x)}}{(d^J + d^K)^2} \times \lambda_i \left(x_i - w_i^J(t)\right), \text{ and} \tag{2.26}$$

$$w_i^K(t+1) = w_i^K(t) - \frac{4\,\epsilon^K(t)\,d^J\,f'|_{\mu(x)}}{(d^J + d^K)^2} \times \lambda_i \left(x_i - w_i^K(t)\right). \tag{2.27}$$

The in-class variables include $\epsilon^J(t)$ for the learn rate at a given time step (t), and d^J for the relevance-weighted squared Euclidean distance. The out-of-class variables are denoted with a superscript K. The derivative of the sigmoid ($f'|_{\mu(x)}$) is evaluated for a given value of the misclassification function, $u(x)$:

$$\mu(x) = \frac{d^J - d^K}{d^J + d^K}. \tag{2.28}$$

The misclassification function of a sample (x)produces results on the interval [-1,1]. Values less than zero identify samples that have been classified correctly and values greater than zero identify misclassified samples. After the prototypes are updated, the relevance, λ_i, of the attributes are modified by

$$\lambda_i(t+1) = \lambda_i(t) - \frac{2\,\epsilon^\lambda\,f'|_{\mu(x)}}{(d^J + d^K)^2}\left(d^K\left(x_i - w_i^J(t)\right)^2 - d^J\left(x_i - w_i^K(t)\right)^2\right), \tag{2.29}$$

where ϵ^λ is the relevance learn rate [30]. If a relevance is less than zero, it is set to zero. Additionally, all the relevances are normalized to sum to one.

The GRLVQI algorithm attempts to maintain consistent in-class prototype winner selection via conscience learning [19]. Conscience learning has been shown to increase classification performance with a hyperspectral data set for DeSieno's LVQ2.1

classification paradigm [56]. A biased distance, d_{Bias}, is used in determining the winning in-class prototype and updates of weight and relevance, where the prototype bias, B^P, is dependent on the prototype selection frequency, F^P and the distance to the prototype from the input sample $d^P = d^J$. The biased distance [1] is determined by

$$d_{Bias} = d^P\,(1 - B^P). \tag{2.30}$$

The winning prototypes frequency is updated using a user-defined parameter, β, by

$$F^P_{new} = F^P_{old} + \beta\,(1 - F^P_{old}). \tag{2.31}$$

All the other prototypes with the same class have their frequency updated by

$$F^P_{new} = F^P_{old} + \beta\,(0 - F^P_{old}). \tag{2.32}$$

The bias is calculated with another user-defined parameter, γ, and the number of prototypes with the same class, P:

$$B^P = \gamma\left(\frac{1}{P} - F^P_{old}\right). \tag{2.33}$$

This algorithm refines the updates of the prototypes by means of a learn schedule for the learn rates, γ and β. The learn schedule will likely be data set specific. This learning algorithm may be used as a feature selection method of the embedded type. The feature relevances are used as weightings to select a subset. GRLVQI provides an iterative approach to developing the feature weights, which is refined as the classifier learns. This is distinct from the C4.5 algorithm which prunes features after the generation of the classifier.

[1] In reality, the d_{Bias} is computed as Equation 2.30 [55]. The original publication of GRLVQI specifies it as $d_{Bias} = d^P - B^P$ [56].

2.3.5 RELIEF-F. RELIEF-F is an extension of the RELIEF algorithm developed by Kira and Rendell [35, 36] and expanded by Kononenko in [38]. The RELIEF algorithm for the two class problem estimates the quality of features based on within feature dimension distances. The algorithm searches for the two nearest neighbors of a given sample. One is the in-class nearest neighbor denoted the *nearest hit*. The other is the out-of-class nearest neighbor denoted the *nearest miss*. The weights of the features are adjusted based on single feature distances from the *nearest hit* and *nearest miss*. The RELIEF algorithm works as follows: set all weights $W[A] = 0$

```
for $i$ = 1 to $m$
    randomly select an instance $R$
    find nearest hit $H$ and nearest miss $M$
    for A = 1 to number_of_features
        $W[A] = W[A] - \text{diff}(A, R, H)/m + \text{diff}(A, R, M)/m$
```

where m is the number of neighbors to search (at most the training set) and diff$(\cdot)$ is the distance between the values of a given feature, A, for two instances (e.g., a given instance (R) and nearest hit (H). For continuous features, the difference is the normalized distance for the interval [0, 1]. For nominal values, the difference is either one for distinct values or zero for identical values. Dash and Liu [17] incorrectly assess the time complexity of the RELIEF algorithm by stating it "requires linear time in the number of given features and number of instances." The RELIEF algorithm has a linear runtime of in respect to the number of features (N), if the result remains unsorted. In terms of the number of samples (S), the runtime complexity is O($SlogS$) since the nearest in- and out-of-class neighbor must be found for each sample.

RELIEF-F is one of two modifications to the RELIEF algorithm to account for missing values, noisy data, and multiple classes. RELIEF-F accounts for a *nearest miss*, M(C), for each class and the prior probability of the classes, $P(C)$. The weighting of a feature is determined by the expected difference from the nearest miss.

RELIEF-F is similar to the RELIEF algorithm above, with a change in the calculation of the weighting,

$$\begin{aligned} W[A] = W[A] - \mathrm{diff}(A, R, H)/m + \cdots \\ \sum_{C \neq \mathrm{class}(R)} P(C) \cdot \mathrm{diff}(A, R, M(C))/m. \end{aligned}$$

Kononenko shows that RELIEF-F doubles the classification accuracy with noisy data, and significantly improves classification with noise-free data in his experiments [38].

2.3.6 Probability of Error and Average Correlation Coefficient.

Mucciardi and Gose [63] discuss seven methods for selecting features. These methods seek to overcome the computationally intractable problem of searching all possible combinations of dimension subsets in finding a "good" subset for classification. The authors use the term "good" because the methods do not find optimal subsets. This thesis discusses three of the seven techniques to include probability of error (POE), average correlation coefficient (ACC), and the weighted sum (POEACC). The three approaches are examples of filter methods.

The first method, POE, ranks features in ascending order of the likelihood of classification error. The POE associated with a feature is determined by classifying the training set by the given feature. The resulting percentage of misclassified samples is then used as the POE for the given dimension. POE inherently makes the assumption that the features independently influence the classification performance.

The second method, ACC, ranks features by their average correlation coefficients. The correlation used here is the absolute value of Pearson's correlation coefficient, r, of a pair of features and is calculated by

$$r_{x,y} = \left| \frac{cov(x, y)}{s_x s_y} \right|, \tag{2.34}$$

where $cov(x, y)$ is the sample set covariance of a pair of features (x and y) and s is a sample set standard deviation of a given feature. A correlation value of one indicates a linear relationship between the two sets (x and y) and a value of zero indicates no linear relationship. The first feature is selected by the lowest POE. The second feature is selected with the smallest correlation to the highest ranked feature (lowest POE). Additional features are selected in order of lowest mean correlation to the set of features already ranked. The mean correlation of a given feature, x with a set of selected features is determined by

$$r_{mean} = \frac{\sum_F r_{x,y_i}}{F}, \tag{2.35}$$

where there are F selected features and y_i is a given selected feature. For example, the 10^{th} feature is selected based on its mean correlation with the first 9 ranked features.

The third method, weighted sum or POEACC, ranks the features in order of smallest to largest of a weighted sum of POE and ACC. Again, the ordering begins with the feature with the lowest POE. Succeeding features are chosen by minimizing the weighted sum,

$$W = w_1 * (\mathrm{POE}) + w_2 * (\mathrm{ACC}). \tag{2.36}$$

The weights, w_1 and w_2, must sum to 1. Before the first iteration, the POE values for the features are normalized by

$$\mathrm{POE}_i^{norm} = (\mathrm{POE}_i - \mathrm{POE}_{\mathrm{min}})/(\mathrm{POE}_{\mathrm{max}}\mathrm{POE}_{\mathrm{min}}). \tag{2.37}$$

With each iteration of selecting the next feature, ACC is also normalized in the same fashion. As stated by Mucciardi and Gose, the rescaling "represent[s] the true measure of importance of the terms in Equation 2.36". This method functions as POE or ACC depending on whether w_1 or w_2 is set to 1, respectively.

It is not intuitively obvious to this author how correlation is determined for nominal data in the POEACC algorithm. The Correlation-Based Feature Selection algorithm implements a method of determining a weighted Pearson's correlation coefficient of nominal data by using the prior of the possible nominal values [28]. This probabilistic method for determining the correlation of nominal data is incorporated into the POEACC to handle the requirements of the data set.

2.3.7 Classifier Accuracy Rate. This section addresses an alternate type of feature selection that are referred to as wrapper methods. These methods literally search the combinatorial space of feature subsets. As an example, one may have 248 features which will result in over 4.5×10^{74} possible combinations. One may search the subset space and evaluate a given feature utilizing a classifier's accuracy rate. The classification accuracy provides a direct performance measure of the subset's capacity to define the domain boundary. To be able to efficiently and effectively select a subset of features, one must take advantage of heuristics, local search optimizations and stochastic sampling to direct the search to a local but useful solution. This thesis discusses two search algorithms genetic algorithms and best first search.

2.3.7.1 Genetic Algorithms. Genetic algorithms take advantage of stochasticity to search a breadth of the state space by evolving states over a series of generations [76]. The search method modifies the state's genotype, representation of the subset, in traversing the space to identify potential solutions called a phenotype. A combination of random successors and successors developed by combining parents are used to search the space. The combination of parent genotypes is called crossover and random selection is considered a mutation. One optimizes a fitness function to identify and retain valued states (i.e., a defined number of feature subsets) for a given generation. In this case, the fitness function is the classifier accuracy rate. The search for a well performing phenotype typically ends after a set number of generations.

2.3.7.2 Best First Search. A heuristic-based search called best first search greedily traverses the combinatorial space of features via a tree structure [76]. The classifier accuracy rate serves as the heuristic in guiding the search path. To expand the search region and reduce the chance of ending in a relatively poor performing local maxima, a backtracking capability is used to traverse back up the tree from stale paths that do not increase the performance.

2.4 Summary

This chapter discusses the framework of intrusion detection and the methodologies for detecting cyber attacks. The two major methodologies include signature-based and anomaly-based detection. Several anomaly-based machine learning approaches are reviewed. Research on classification techniques for intrusion detection have been studied in depth with a large interest in the area of BPs. On the other hand, feature selection of network flows has been limited to improving performance for a given classifier. Finally, a survey of feature selection methods outlines varied approaches, which spans the feature selection taxonomy. This review of the literature has not found an indepth analysis of features for improving performance, minimizing data extraction requirements, or validating a useful feature subset for the intrusion detection domain.

III. Feature Selection Methodology

Feature subsets and subset generality for intrusion detection are determined by assessing subset classification performance and analyzing the features composing the subsets based on domain knowledge. Figure 3.1 illustrates the methodology in creating a subset for each feature selection method and evaluating the subsets for generality. Feature selection methods generate feature weightings and subsets, which are ranked to develop an ordered list of subsets in association with each feature selection method. The subsets are ordered by the number of highest ranked features composing the subset. A common classifier evaluates the performance of the subsets and down select to a single subset for each feature selection algorithm. The selected subsets are evaluated for generality based on their performance on a diverse grouping of classifiers. This chapter continues by discussing the motivation for the work, methods to establish and evaluate features, an extension to a feature selection method, and the evaluation techniques for classification performance and statistical analysis.

3.1 Intrusion Detection and Feature Selection

Intrusion detection provides an application and motivation for the feature selection objective of this thesis. An intrusion detection system obtains information from sensors within a network to record and possibly process raw data [40]. The detection system generates alarms, which may generate an action back on the network or notify a network administrator. Feature selection may significantly improve classification performance and/or reduce the required capacity to monitor, manipulate, evaluate, and store the massive amount of real-time data expected to be seen over a network. Some feature selection methods add a significant amount of processing time and may require retraining of a classifier (e.g., C4.5 Decision Tree) so real-time implementation of some feature selection methods is not feasible without extraordinary processing capacity. Feature selection can play a role in reducing the quantity of data read off the network by utilizing it as a preprocessing step in developing a classifier-based anomaly

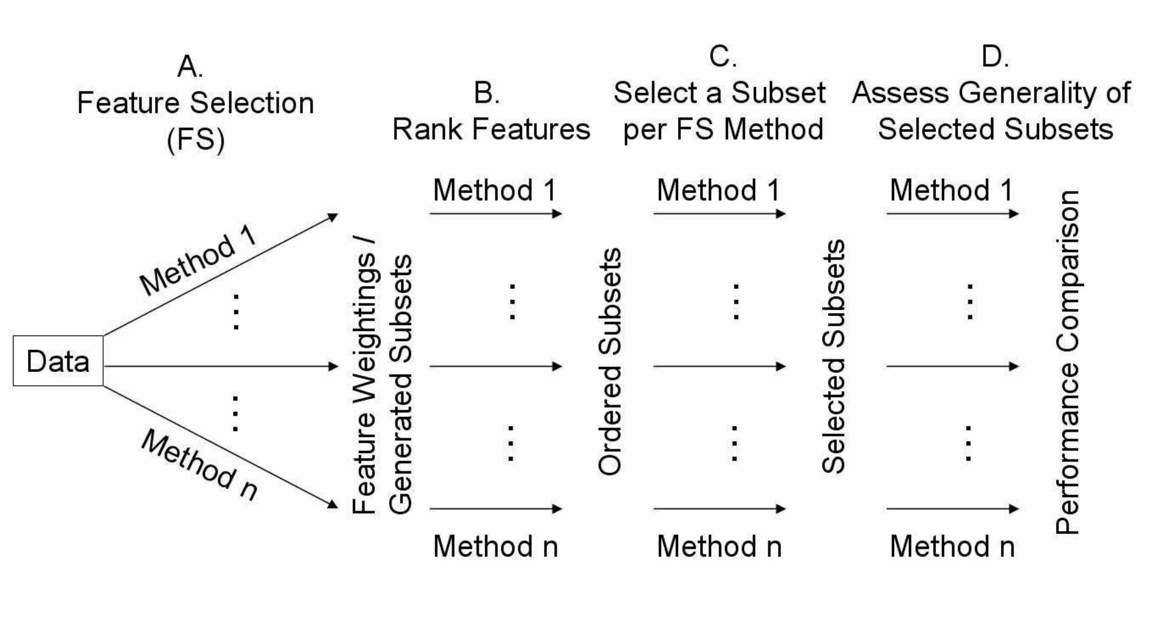


Figure 3.1: The feature analysis process assesses the effectiveness of feature selection methods, and analyzes feature subset generality

detector. Additionally, feature selection may be utilized to enhance the classification performance via post-processing analysis of recorded traffic.

Figure 3.2 illustrates the feature selection process for determining and validating a good feature subset. In this thesis, a number of feature selection algorithms are compared to determine a good feature set. The feature selection algorithms perform the generation and evaluation methods, and a stopping criteria in obtaining a subset from a previously extracted and processed set of features. The classification methods under test identify cyber attacks and network services for the evaluation and validation components of the process. The validation piece involves testing the selected feature subset for performance in several cases for usefulness such as predictive accuracy, number of features selected and the generality of the features. The intrusion detection system is not integrated into a genuine architecture for the purpose of network surveillance. Although, processed data from real network traffic is used in selecting features and testing the effectiveness of subsets. The intended results of this

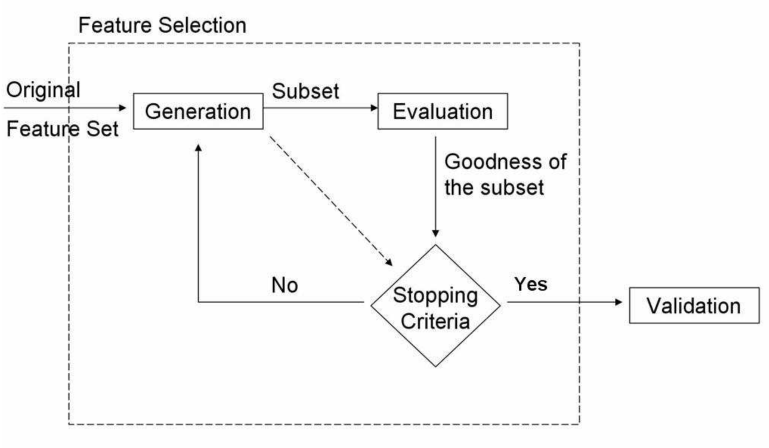


Figure 3.2: This is the feature selection process from Dash and Liu [17]. Subsets of features are generated and evaluated. Based on a stopping criteria, the highest evaluated subset is selected

work includes a validated feature subset for intrusion detection and an analysis as to why the features function well.

3.2 Establishing and Evaluating Feature Subsets

One may accomplish an evaluation of features in a direct or indirect fashion. Direct evaluation involves *a priori* knowledge of the features. Absolute understanding of the features and their role in classifying attacks or network services would mean an expert may just select the optimal feature set. In actuality, a person understands aspects of the domain, which would allow for identification of a modicum of representative features or analysis as to why certain features perform well. For example, an expert would come to the correct conclusion that well-known port numbers provide a very good method for classifying network services. However, port numbers would be uninformative when a flow is masquerading as another network service (e.g., spoofing attack or peer-to-peer traffic). Analysis of other flow features may provide the needed information to classify these flows. Indirect evaluation requires assessment of the performance of a classifier to determine a good feature subset. Even though a specific feature selection technique may provide an optimal subset for the given evaluation method, the selected subset would not be optimal for the general classification task [17]. This thesis indirectly evaluates feature subsets with classifier performance and directly evaluates the relevance of the selected features.

A set of feature selection methods are used to evaluate important features for performing intrusion detection (Figure 3.1 (A)). The chosen feature selection algorithms for comparison span the taxonomy discussed in Section 2.2.1 to include: RELIEF-F, the C4.5 Decision Tree Method (DTM), Probability of Error and Average Correlation Coefficient (POEACC), Generalized Relevance learning Vector Quantization Improved (GRLVQI), classifier accuracy rate with genetic algorithms and best first search, and median Bhattacharyya methods. Additionally, the comparison includes the minimum surface Bhattacharyya method, which is a proposed extension to the median Bhattacharyya method (see Section 3.3). Table 3.1 outlines some differences and commonalities between the methods. These methods utilize varied approaches for finding and measuring relevant features such as distance, dependencies, information theory, separability, actual performance, individual weightings of features, stochastic subset search, and heuristic subset search. The methods encompass all three categories of Blum and Langley's taxonomy [7] and cover a majority of Dash and Lui's evaluation and generation methods [17]. Individually, the methods provide a capacity for handling noisy, irrelevant, redundant, and/or interdependent data. Furthermore, all of the feature selection methods handle continuous, discrete and nominal data. These methods are not couched to be the best approaches for this domain, since top performing feature selection methods have not been identified in the literature. This diverse group of feature selection algorithms allows for varied "perspectives" to accentuate relevant features that may not be selected otherwise. Consistency among the subsets suggests strengthened evidence of a feature's relevance.

3.2.1 Ordering Subsets for Evaluation. This section discusses the method for ranking and ordering subsets for selecting a subset per feature selection method (Figure 3.1 (B)). Some feature selection algorithms such as GRLVQI, POEACC and

Table 3.1: Characteristics of Feature Selection Methods

Feature Selection Method	Evaluation Measure	Generation Method	Blum and Langley Category
Best First	Accuracy Rate	Best First Search	Wrapper
C4.5	Entropy	feature evaluation	Embedded
Genetic	Accuracy Rate	Genetic Algorithm	Wrapper
RELIEF-F	Feature distance	Class-based feature evaluation	Filter
POEACC	Correlation	Feature evaluation	Filter
GRLVQI	Sample distance	Prototype-based feature evaluation	Embedded
Median Bhattacharyya	Separability	Inter-class feature evaluation	Filter
Minimum Surface Bhattacharyya	Separability	Inter-class feature evaluation	Filter

RELIEF-F provide a list of weighted features. A rank is assigned to each feature based on its weighting, where the highest and lowest weighted features receive a rank of one and the number of features, respectively. For a set of T-Monte-Carlo runs with k-folds each, the median rank is used to order the complete set of features. If there is a tie based on median rank, it is broken by the skewness of the features' rank results over the folds. Smaller skewness results in a feature coming first in the order, since positive skew results in a right-handed tail (tending toward larger rank) and negative skew indicates a left-handed tail (tending toward smaller rank). Ordered subsets for a weighted feature selection algorithm may be generated by starting with the first feature of smallest median rank and adding successively ranked features iteratively. For example, the three highest ranked features from the POEACC algorithm are initial window in bytes from client to server (rank 1: feature 95), minimum inter-arrival time (rank 2: feature 3), and the variance of the inter-arrival time (rank 3: feature 9). The first three ordered subsets are:

- Subset 1: 95
- Subset 2: 95, 3
- Subset 3: 95, 3, 9.

Some feature selection algorithms (DTM and classifier accuracy rate with genetic and best first search) select a feature subset for each fold. For a set of T-Monte-Carlo runs with k-folds each, features are ranked by the number of folds, where a feature is included in a selected subset. Features that appear in the greatest number of folds (maximum of $T \times k$ folds) are ranked highest. For example, the seven highest

Table 3.2: Top Four Features from Accuracy Rate with Genetic Algorithm

Feature Index	Feature Name	Times Selected for Subset	Rank
1	Server Port	300	1
2	Client Port	300	1
83	Minimum Segment Size from Client to Server	299	2
129	Maximum Full-size RTT sample	286	3
86	Average Segment Size from Server to Client	278	4
105	Truncated Packets from Client to Server	276	5
168	First Quartile Size of Data Control Portion of Packet	276	5

ranked features from the accuracy rate with a genetic algorithm method are listed in Table 3.2. Sets of features included in the same number of folds are considered a grouping (e.g., features 1 and 2 are a grouping) and feature 83 is a grouping of one. Ordered subsets are generated by iterating down the set of ranked features. The first ordered subset consists of the grouping with the largest number of folds (i.e., the highest rank). Successively lower-ranked groupings are iteratively added to the preceding ordered subset to generate all the ordered subsets. For accuracy rate with a genetic algorithm, the ordered subsets from the top seven features are:

- Subset 1: 1, 2
- Subset 2: 1, 2, 83
- Subset 3: 1, 2, 83, 129
- Subset 4: 1, 2, 83, 129, 86
- Subset 5: 1, 2, 83, 129, 86, 105, 168.

3.2.2 Determining a Subset for Each Feature Selection Algorithm. With successive subsets for a given feature selection algorithm, a single subset for the specific feature selection algorithm may then be found (Figure 3.1 (C)). There are two typical approaches to selecting a subset [17,46]. The first is to set a weighting threshold or a predefined limit in the number of features. The second method utilizes classification performance to choose a subset as implemented in this thesis. The Näive Bayes classifier discussed in Section 2.3.2 determines the classification performance of each of the ordered subsets. Classification performance is evaluated with the Näive Bayes classifier primarily due to computational speed, which allows for a multitude of subsets to be evaluated within a reasonable amount of time. Approximately 1,800 ordered subsets are evaluated to down select to 8 subsets, one per feature selection method. Furthermore, Witten, *et al.* [87] and Moore [60] notes that the performance of the Näive Bayes classifier is sensitive to redundant and irrelevant features. Hence, the trend in Näive Bayes classification performance is indicative of how well a given feature selection method identifies relevant features.

One expects that the domain space and decision boundary to be adequately represented by more than one feature subset. If a feature selection algorithm appropriately ranks the subsets, one would expect a consistent and dramatic increase in performance until the domain has been adequately represented. Figure 3.3 illustrates the hyperbolic tangent, which is analogous to the behavior anticipated from ordered subsets from a feature selection method. For the ordered subsets from a *perfect* feature selection algorithm, the first subset of the plateau is the smallest subset of the entire feature set that best represents the domain. From the first subset of the plateau, added features cause diminishing returns in classification performance. Based on this expectation of behavior, one may select the first subset where the difference between adjacent subsets stabilizes below a given threshold, which indicates a plateau in performance. Additionally, a multiple comparison of the classification performance from the subsets indicates whether there are statistically-significant differences between the performance of ordered subsets as discussed in Section 3.4.2. This allows one to select a smaller subset in number of features (than the plateau indicates) if the classification performance of the smaller subset is the same as or better than a larger subset (i.e., the initially selected subset due to the plateau).

With a set of selected feature subsets from each algorithm, reduction to a single subset of features to represent the domain is ideal but may not be effective. If a sufficiently small number of distinct features remain in the final set, then an exhaustive search of all combinations likely determines the best performer with the Näive Bayes classifier [8,17]. If the number of features remaining (i.e., union of the selected subsets)

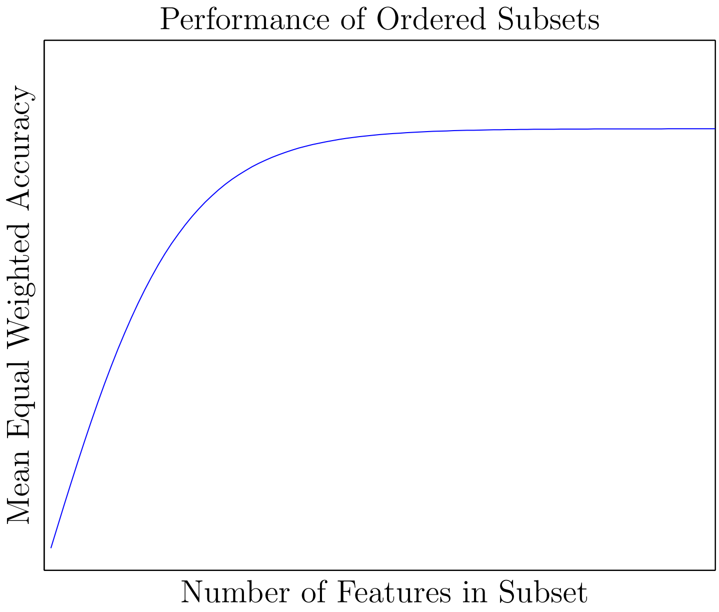


Figure 3.3: The hyperbolic tangent represents the expected trend in classification performance of ordered subsets from feature selection algorithms

is too large, one may re-rank the features and down select to a single subset in a similar manner to that discussed in Section 3.2.1. Neither approach helps in establishing a subset to "appropriately" describe the decision boundary. Since the feature selection algorithms have significant differences among them, their capacity to provide the best subset for the given data set is unequal. Some feature selection algorithms will perform better than others in different respects. Due to the differences in feature selection methods, a cogent method for combining the subsets may not be warranted. For example, some will handle synergies (e.g accuracy rate with best first search) while others will not (e.g., RELIEF-F). Synergistic combinations would likely be lost with the ordered approach. An unsatisfying approach selects the statistically best performing subset. Even this approach may result in a subset which does not have generality. A down selection to a single subset will not be performed; all the feature subsets will be assessed for generality.

3.2.3 Generality of Features. The final step of the process determines whether the selected feature subsets provide an adequate description of the domain space for establishing a set of decision boundaries (Figure 3.1 (D)). A subset demonstrating generality is able to represent the domain in such a manner. One may determine generality by answering the following question: Does the subset increase performance for a set of classifiers [46]? In the worst case, a feature subset would optimize the performance of only a given classifier due to the specific learning method. There are numerous classifiers that may be utilized for assessment of generality. Figure 3.4 is recreated from Holmström, *et al.* [32], which paints the relationships between common classifiers. The schematic relationships provide a basis for selecting differing types of classifiers. The x-axis represents architectural flexibility, which is indicative of error-based correction techniques, incremental learning, or other learning methods that provide adaptation to the training set. The y-axis pertains to neural or nonneural training approaches so neural training would implement processing elements or prototypes requiring "local computations" [32]. Nonneural training is categorized by statistical representations of the training set such as local averages or kernel estimation. C4.5 and Näive Bayes were added to Figure 3.4 based on the thesis author's interpretation of the depiction.

With time as a limiting factor, four classifiers of different learning and identification methods are selected to illustrate the generality of the features: C4.5, Näive Bayes, GRLVQI, and multilayer perceptron with backpropagation (BP). Figure 3.4 shows the four classifiers covering a broad range over the classifier taxonomy. Näive Bayes lies in the lower left-hand quadrant and C4.5 lies near the center-right. BP and GRLVQI lies at the top right, where GRLVQI is an extension of LVQ. Additionally, Table 3.3 denotes the distinctions between the specific classification methods. C4.5 is an information theory based classifier that is assembled using a divide and conquer approach and then prunes based on the probability of error of the subtrees. Quinlan [70] refers to C4.5 as a "logical" classification model since the decision tree may be described with expressions of predicate logic. Näive Bayes utilizes maximum likelihood based on a Gaussian distribution, while treating the features as statistically independent (this is the "näive" assumption). GRLVQI provides a prototype-based

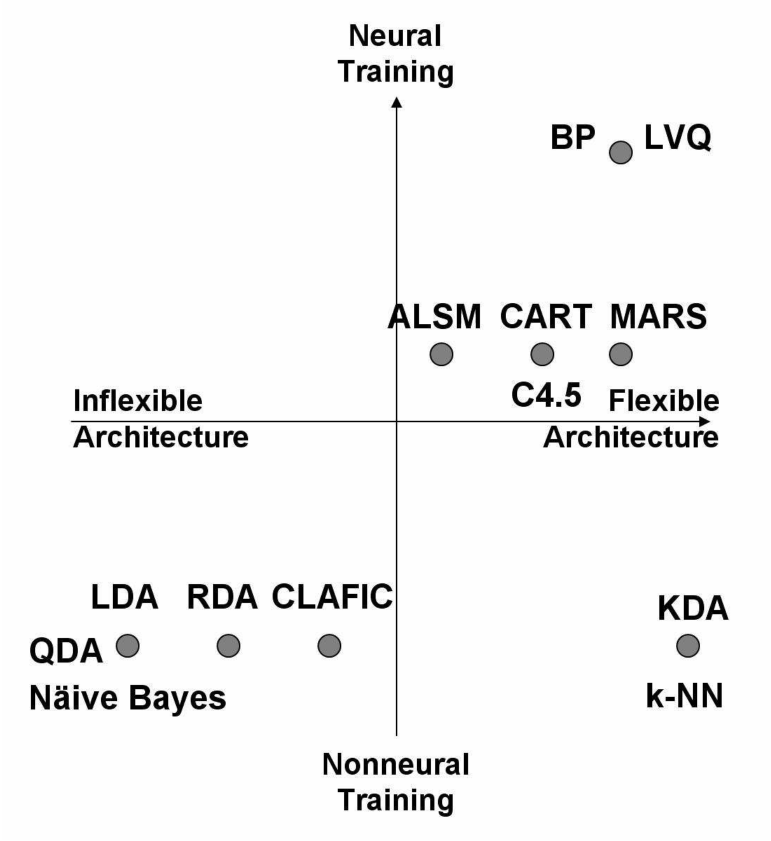


Figure 3.4: The schematic of the classifier taxonomy from [32] illustrates the distinctions in learning and architecture between the classification methods for assessing generality (e.g., C4.5, Näive Bayes, GRLVQI, and BP)

Table 3.3: Characteristics of Classification Methods

Classification Method	Evaluation Method	Classification Model
C4.5	Entropy	Logical
GRLVQI	Sample Distance	Prototype-based
BP	Non-linear combination	Nonlinear
Näive Bayes	Bayes Theorem	Probabilistic

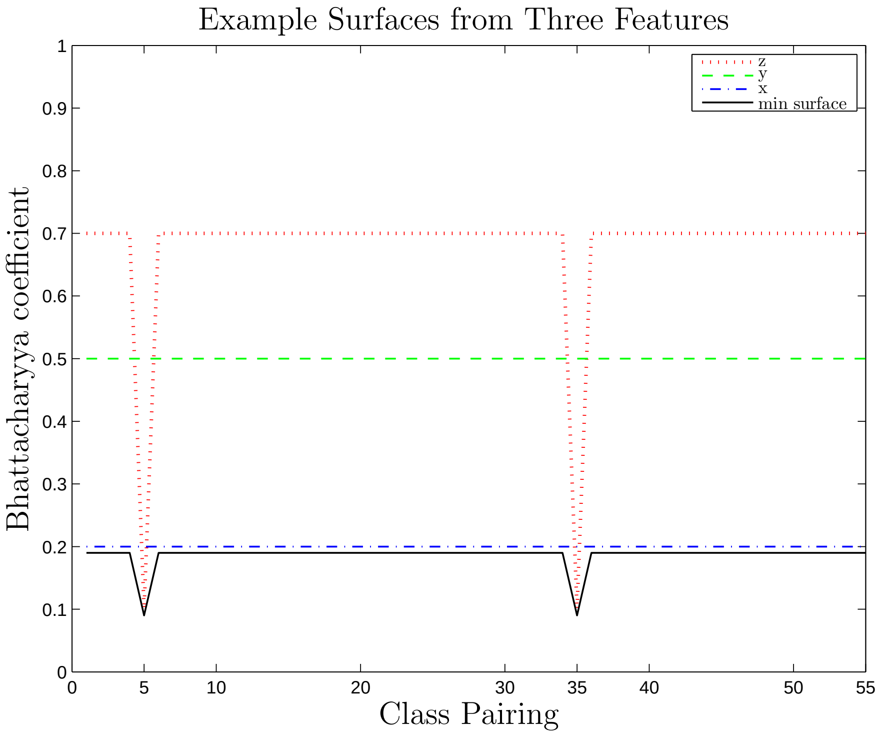


Figure 3.5: Feature z has poor performance in general. For two class pairs (e.g., 5 and 35), feature z provides the best (minimum) separability. The mean performance for this example is 0.2, 0.5, and 0.6782 for x, y, and z, respectively. The current minimum surface consists mostly of feature x and two class pairs of feature z

learning method based on distance, which prioritizes features based on their classification potential. The GRLVQI classifier learns and prioritizes features simultaneously. Finally, the BP is a nonlinear neural network trained based on the backpropogation of error through the network.

Statistically-significant improvement between subsets is determined by multiple comparison of the classification performance results for a given classifier as discussed in Section 3.4.2. For the identified classification algorithms, the classification performance of the selected subsets from each feature selection method are compared to the classification performance of all the features. Performance gains for all the classifiers by a given feature subset demonstrates generality for the network traffic classification problem.

3.3 A Proposed Feature Selection Approach

The mean or class-weighted Bhattacharyya methods for feature selection, from Section 2.2.1.1, is lacking in finding the smallest set of features for the multiple-class classification problem. This section details a new approach for selecting features based on Bhattacharyya coefficients. The approach treats the Bhattacharyya coefficient for the $C(C-1)/2$ pairs of C classes as a surface and sorts the features by iteratively selecting members consisting of the minimum surface. This feature selection approach will be referred to as the minimum surface Bhattacharyya method. For example, Figure 3.5 depicts the surfaces of three features. The mean of the features are 0.2, 0.5 and 0.6782 for features x, y, and z, respectively. The mean of the three surfaces obfuscates features with the best separability for a given class pair. That is, the mean (or median) Bhattacharyya represents average (or median) performance and has no intelligent way of adapting to local "best performance" in its ordering process. By selecting the features creating the minimum surface (e.g., features x and z in the first iteration), the separability for each class pair is optimized in the feature ordering. Hence, features are greedily selected in a manner that preserves the best separability without searching through all combinations of features. The order of the features utilizing the minimum surface are x, z, and then y. If the number of class pairs exceeds the number of features, the ordered list of features from the minimum surface method approaches the median method when there is high variability in the Bhattacharyya values across the class pair space. Modifying the example in Figure 3.5, lets make the three features (x, y, and z) each have a minimum of 0.1 at separate class pairs. A minimum surface method results in a sorted list of features based on the median.

Algorithm III.1 details the pseudocode for implementing the sorting method based on the minimum surface. The proposed minimum surface method for sorting the features maintains an open list of all the features. The method selects all the features that have a minimum value for a given class pair across the entire combinatorial space of pairings. If there is a tie in the minimum value for a given class pair, the feature

Algorithm III.1 Minimum Surface Bhattacharyya Pseudocode: The Pseudocode sorts based on the minimum surface of Bhattacharyya coefficients. If two features lie on the minimum surface at the same class pair, the tie is resolved by the feature with smallest median Bhattacharyya value.

```
Require: Array $f[0, number\_of\_features - 1]$ of $value$ arrays {a $value$ array per feature}
Require: Arrays $value[0, number\_of\_class\_pairs - 1]$ of real numbers {a Bhattacharyya coefficient per class pair}
  $sorted\_list \leftarrow <empty\ list>$
  $open\_list \leftarrow [1, number\_of\_features]$ {list of all feature indices}
  while $open\_list$ is not empty do
    $min\_value \leftarrow \min(f, open\_list)$ {array of minimum Bhattacharyya coefficients for each class pair in $open\_list$}
    $min\_surface \leftarrow <empty\ list>$
    for $i = 0$ to $number\_of\_class\_pairs$ do {find the features that generate the current minimum surface}
      $temp \leftarrow -1$
      for each feature index, $j$, in $open\_list$ do
        if $f[j].value[i] = min\_value[i]$ then
          if $temp$ = -1 or median($f[j].value$) < median($f[temp].value$) then {if multiple features have minimum value, select feature with smallest median}
            $temp \leftarrow j$
          end if
        end if
      end for
      $min\_surface \leftarrow$ union($min\_surface, temp$)
    end for
    sort $min\_surface$ from smallest to largest median Bhattacharyya value
    append $min\_surface$ to end of $sorted\_list$
    $open\_list \leftarrow$ difference_between_sets($open\_list, min\_surface$) {remove features in $min\_surface$ from $open\_list$}
  end while
  return $sorted\_list$ {sorted list of feature indices}
```

with the smallest median Bhattacharyya value is selected. The selected features (composing the minimum surface) may then be sorted by their median value, added to the end of the sorted list, and then removed from the open list. The method is performed iteratively until all of the features have been removed from the open list. The result of method is an ordered list of features, where a subset may be selected based on a threshold or approach outlined in Section 3.2.2.

By sorting the features based on their minimum Bhattacharyya coefficient, one will guarantee the features that provide the best separability for each class will be ordered first. Assuming the correlation of separation and performance is maintained for various classification problems, this method should result in a smaller subset providing comparable performance to that of the mean or median Bhattacharyya method. Fundamentally, the Bhattacharyya coefficient bases a given feature distribution on a sampling of data, not a model of the data. The method assumes the sampling accurately represents the data population in selecting features. Redundant features will impact the capacity in obtaining the minimal set. Although they will not lie on the same minimum surface, the redundant features will be sorted in successive surfaces. Hence, the method does not have a manner for handling highly correlated data. For features that are highly correlated, one of the features may be removed from the open list in advance of the sort to minimize the feature set.

3.4 Evaluation Techniques

This section expands on two aspects of the feature analysis process that are not explicitly depicted in Figure 3.1, but Section 3.2 outlines as criteria or methods of analysis. First, the criteria for portraying classification performance is defined. Second, the statistical analysis methods for analyzing the classification performance results are covered.

3.4.1 Measuring Classifier Performance. A classifier's *equal weighted accuracy* (EWA) is used to compare the classification performance of a given feature subset with another subset or the entire set for a given classification method. The EWA is a non-biased performance estimate, where each class accuracy contributes equally. The EWA is computed as:

$$EWA = \frac{1}{C} \sum_{m=1}^{M} \sum_{c=1}^{C} \frac{1(if x_m \in X_c \,\&\, y_m = c)}{M_c}, \tag{3.1}$$

where M is the number of samples, M_c is the number of samples in class c, C is the number of classes, and $1(\cdot)$ is an an indicator function. The indicator function evaluates true (i.e., 1) if the sample x_m is a member of the class c set (X_c) and the predicted class label, y_m, is class c. Otherwise, the indicator function evaluates false (i.e., 0). A less formal definition is the average of all the class accuracies. For a non equal weighted accuracy (NEWA), the accuracy is calculated by the number of correctly classified samples divided by the total number of samples. The NEWA is biased by the performance of the classes with disproportionately large samples, which is the case for some classes in this thesis as discussed in Section 4.1.1.

3.4.2 Determining the Statistical Significance of Classifier Improvement for Retained Features. Statistical analysis intends to highlight statistically-significant differences among the results. The method for determining statistical significance is dependent on the distribution of a given set of results to be analyzed. The Anderson Darling Test determines the type of test which needs to be performed. Then, one of two methods for performing a multiple comparison are used. At times, the nonparametric multiple comparison method is too conservative so a paired nonparametric test called the Wilcoxon Signed Rank test determines statistical significance between two sets of runs (i.e., classification performance of two subsets).

3.4.2.1 Anderson Darling Test. The Anderson-Darling test is used to determine whether a set of EWA results for T-Monte-Carlo runs follow a Normal distribution with a confidence level of 95%. Analysis of normally distributed and non-normally distributed results warrant distinct techniques due to limiting assumptions and performance attributes of the techniques. The Anderson-Darling test has been demonstrated a more powerful test than other approaches like the χ^2 test and performs well with long tails where the Shapiro-Wilk test suffers [83]. The Anderson-Darling test does not require estimates of key parameters (e.g., mean and standard deviation) and works for any distribution. In testing for normality, a random variable, X, is assumed to be Normal (null hypothesis) and assessed whether the test statistic exceeds the prescibed critical value for a given p-value. Tables of critical values have been established for comparison with the test statistic and may be found in [83]. The Anderson Darling test functions by first normalizing the classification performance results from the runs, X_i, with the mean, $\bar{X}$ and standard deviation, s, using

$$Y_i = \frac{X_i - \bar{X}}{s}. \tag{3.2}$$

The original test statistic, A^2, is calculated using the CDF of the sample data, $F(Y_i)$,

$$A^2 = -n - \frac{1}{n}\sum_{i=1}^{n}(2i-1)\left[\log F(Y_i) + \log(1 - F(Y_{n-i+1}))\right], \tag{3.3}$$

where n is the number of performance samples (i.e., T-Monte-Carlo runs). A modification of the Anderson Darling test statistic for normality by Stephens [16] reduces the required table of critical values to a single vector that does not need to account for the sample size. The modified test statistic is calculated by,

$$A^* = A^2\left[1 + \frac{0.75}{n} + \frac{2.25}{n^2}\right]. \tag{3.4}$$

If A^* exceeds the critical values listed in Table 3.4 from [16], then the sample data set is not normally distributed at the indicated p-value. The procedure is valid for sample sizes greater than eight.

3.4.2.2 Multiple Comparison of Classification Performance Runs.

Classification performance results from unique subsets are compared for statistically-

Table 3.4: Critical Values for Testing Normality using modified Anderson Darling Test from [16]

p-value	0.1	0.05	0.025	0.01	0.005
Critical Value	0.631	0.752	0.873	1.035	1.159

significant differences utilizing multiple comparison methods. The multiple comparison methods are useful for parametric and nonparametric analysis. Parametric analysis assumes a distribution of the results (i.e., Gaussian in this case). Nonparametric analysis allows for comparison of distributions, which do not meet the normality criterion. Both methods are similar in approach but the data is viewed from distinct perspectives. The multiple comparison of normally distributed results utilize the values of a set of runs for a given subset to determine means and standard deviations. The nonparametric method ranks each data sample (run) of all the subsets in the multiple comparison to establish am ordering every sample from smallest to largest. Nonparametric analysis utilizes the rank means and rank standard deviations.

A multiple comparison of the T-Monte-Carlo runs of k-fold cross validation utilizes a family-wise confidence level of 95% to determine a statistically-significant differences [71]. The family-wise confidence level accounts for the compound uncertainty generated by the combination of pairwise comparisons. The pairwise comparisons cause very low family-wise confidences with a combination of paired hypothesis tests. The comparison of multiple sets utilizes $halfwidth$ confidence intervals determined by

$$halfwidth = M_{procedure} * SE, \tag{3.5}$$

where $M_{procedure}$ is a procedure-specific multiplier and SE is the standard error of the pooled sets. Any overlap in confidence intervals centered at the sets' mean or mean rank indicates no statistical difference. The standard error, SE, is calculated as follows

$$SE = s_p \sqrt{\sum_g \frac{1}{n_i}}, \tag{3.6}$$

where s_p is the pooled standard deviation, g the number of groups (i.e., subsets for comparison), and n_i a given groups sample size (i.e., T-Monte-Carlo runs). The pooled standard deviation is defined as

$$s_p = \sqrt{\frac{\sum_g (n_i - 1)s_i^2}{\sum_g n_i - 1}}, \tag{3.7}$$

where s_i is the standard deviation of group i. Nonparametric analysis of the systems may be performed by replacing the standard deviation, s_i with the standard deviation of the ranks.

There are several methods for determining the multiplier such as the Tukey-Kramer and Bonferroni procedures [71]. Tukey-Kramer assumes normally distributed results, where Bonferroni does not. The Tukey-Kramer multiplier provides highly accurate intervals for a given confidence level and is calculated by

$$M_{Tukey-Kramer} = \frac{q_{I,d.f.}(1-\alpha)}{\sqrt{2}}, \tag{3.8}$$

where $q_{I,d.f.}(1-\alpha)$ is the $100 \times (1-\alpha)^t h$ percentile of the Studentized Range distribution for I groups and $d.f.$ degrees of freedom. Alternatively, the Bonferroni multiplier for non-normally distributed results is the $100x(1-\alpha/k)^t h$ percentile of the Student's t distribution, $t_{d.f.}(1-\alpha/2k)$. The number of possible paired comparisons is k and determined by

$$k = \frac{I(I-1)}{2}. \tag{3.9}$$

The confidence level utilizing the Bonferroni multiplier is not predictable like the Tukey-Kramer multiplier. Although, the Bonferroni multiplier always results in a conservative confidence interval [71].

3.4.2.3 Wilcoxon Signed Rank Test. The difference in performance between two classifiers of distinct feature subsets is established with hypothesis testing for a $p < 0.05$. Two generally accepted paired tests include the Student's t-test and Wilcoxon signed-rank test [57]. These methods for hypothesis testing may be applicable depending on whether a given set of performance runs are normally distributed or not. For this thesis, the Wilcoxon sign-rank test is required at times for non-normally distributed results due to the overly conservative nature of the multiple comparison test with the Bonferroni multiplier.

The Wilcoxon signed-rank test compares pairs of observations from two sample sets to determine whether they have the same median. The test assesses the null hypothesis that the difference between the sample sets is zero [24]. Hence, sample sets X and Y contain samples X_1, X_2, X_3, Y_1, Y_2, and Y_3. The absolute differences between the samples, $|X_i - Y_i|$, are ordered and ranked from smallest to largest from one to the number of pairs, n. If there is a tie, the rank is the average rank of the tied differences. The rank is given a sign that matches the true difference between the samples, except for when there is a difference of zero. The sign given to the rank of a difference of zero will hinder the rejection of the null hypothesis. The two-sided test statistics can then be computed by selecting the smaller value of the sums of the absolute of the negative ranks (W_-) or the sum of the positive ranks (W_+). The test statistics may then be compared to a critical value from a table found in a statistics book [24, 57] based on the desired p-value and the number of sample pairs, n. If the test statistic is less than the critical value, the null hypothesis may be rejected and there is a statistically-significant difference in the medians of the sample sets. For example, Table 3.5 illustrates the Wilcoxon signed rank test on two classifiers over four runs. The test statistics are $W_- = 12.5$ and $W_+ = 8.5$ and the critical value for a two-sided test with a p-value of 0.05 is 1. Since neither test statistics are less than one, there is no statistically-significant difference between the classifiers at a two-sided 95% confidence interval.

3.5 Summary

This chapter outlines a methodology for quantitatively selecting a reduced feature subset and assessing the generality of feature subsets with classifier performance.

Table 3.5: Example Wilcoxon Signed Rank Test

Run	Classifier A	Classifier B	Difference	Rank
1	0.70	0.90	-0.20	-5
2	0.95	0.85	0.10	2.5
3	0.95	0.65	0.30	6
4	0.88	0.98	-0.10	-2.5
5	0.90	0.95	-0.05	-1
6	0.80	0.95	-0.15	-4

Statistical methods are covered for the comparative analysis of the performance evaluations. The resulting feature subsets will also be scrutinized against studies into the characterizations of network traffic. The process intends to lead to a well balanced and thorough evaluation of established techniques of feature selection methods to include a proposed extension to a current method for analyzing separability. The results of this work are discussed in the following chapter.

IV. Experimental Results and Analysis

This chapter covers the experimental design, results and validation of results. The experimental design covers the data set and parameters for implementing the experiments. The results detail the analysis for subset selection and assessment of the generality of the selected subsets. Lastly, the results are validated and reaffirmed using the literature on flow characterization of several flow types.

4.1 Design of Experiments

This section discusses particularities of the experimental setup and runs. The rationale for the selecting the data set are reviewed. The composition of the data set and complexity for the given task are also discussed. The section outlines the specifics on the parameters of the machine learning algorithms and the Monte-Carlo runs.

4.1.1 Data Set. Many references pertaining to intrusion detection utilize a training and testing data set developed from Defense Advanced Research Projects Agency's (DARPA's) Intrusion Detection Evaluations [27, 45]. The data sets contain weeks of simulated traffic with labels for identifying malicious from normal traffic. The Massachusetts Institute of Technology Lincoln Laboratory modeled the network traffic of an Air Force Base and inserted known cyber attacks. Lincoln Laboratories performed two evaluations on the simulated data sets in 1998 and 1999. A portion of this data was used to generate a data set for a 1999 competition by the University of California, Irvine, Knowledge Discovery in Data (KDD99) [10]. The DARPA and KDD99 data sets and other artificial data sets are not ideal for training or testing a classifier or assessing classification methodologies. The underlying problem of such data sets result from the generation of background traffic from probabilistic models with attack traffic injected into the flows [74]. Since background traffic conforms to ideal distributions, a classifier based on such a data set does not contend with the noise of real network traffic. Furthermore, McHugh [54] comments that no apparent validation was performed on the artificial background traffic of the DARPA data set, which may bias the results.

Ideally, real network traffic provides the basis for a data set to train a classifier. Moore [61] developed such a data set and documented a detailed description of its features, and method for generation. Moore's data set contains a day of authentic network traffic that was classified by the type of traffic in accordance with the procedure in [59]. The traffic labeled as cyber attacks have been determined utilizing known signatures. The attack traffic are mostly categorized as worms and viruses. A recreated table of the features contained in the data set from [61] is listed in Appendix B.

The data set provides a look into a real world application of the feature selection problem since it is extracted from a day of network traffic and has already been processed for metrics and statistics [61]. The data set contains 377,526 samples of network flows, 248 features, and 12 classes, whose features include nominal, discrete, continuous, missing and noisy values. The samples of the data are restricted to bi-directional Transmission Control Protocol (TCP) flows. A portion of the data set is utilized for this work since the original data set consists of too many network flows to handle in a reasonable amount of time and the author encountered unresolvable heap space issues with the original data set. The reduced data set consists of 40,858 flows. A majority of the flows consist of email and world wide web traffic so they have been limited in the reduced data set. One class has been entirely removed due to insufficient instances. The composition by class of the original and reduced data set is shown in Table 4.1.

The features may be categorized by protocol parameters, performance, volume and size. The categories describe the type of flow characteristics (e.g., features) extracted in creating the data set. The protocol parameters include information taken directly from packet-level headers. Performance pertains to a combination features that are affected by flow and network dynamics (e.g., throughput). Volume includes the quantity of certain distinguishing packet traits. Size encompasses features that describe the flows in terms of bytes. Table 4.2 provides several examples of features by category. The features describe host to host sub flows and aggregate bidirectional

Table 4.1: Data Sets - Number of Instances in each Class

Class	Original Data Set	Reduced Data Set
Games	8	0
Interactive	110	110
Multimedia	576	576
Attack	1,793	1,793
Peer-to-peer	2,094	2,094
Services	2,099	2,099
Database	2,648	2,648
File Transfer Protocol-passive	2,688	2,688
File Transfer Protocol-control	3,054	3,054
File Transfer Protocol-data	5,797	5,797
Mail	28,567	9,999
World Wide Web	328,092	10,000
Total	377,526	40,858

statistics and metrics. Specifically, the features include quartile, min, max, average and median statistics.

Table 4.2: Example Features by Category

Protocol Parameters	Performance	Volume	Size
stream length	inter-arrival time	number of out-of-order packets	average packet size
average window size	throughput	number of acknowledgment packets	total bytes sent
request for max segment size	round trip time	number of retransmissions	amount of control bytes set

The data set presents a complex domain with high dimensionality, varied correlation, multiple feature types and missing values. Analysis of the data sets show that some features are redundant and/or uninformative for the classification task. Utilizing Pearson's correlation coefficient, the feature pairs indicated in Table 4.3 have a perfectly linear association indicated by $|r| = 1$. Of the pairings, only a single member of a pair would need to be assessed for feature selection. Additionally, the data set contains features with no utility since all their values are zero or missing. Uninformative features are noted in Table 4.4 and may also be removed. Many pairs of features contain extremely high correlations in excess of 0.99. For the 248 features, there are 30,628 possible combinations of pairings. There are 74 and 326 feature pairings that have absolute correlations greater than 0.99 and 0.9, respectively. The data set contains features whose values are binary, whole and real numbers. Additionally, nearly a third of values for some features are missing data.

Table 4.3: Pairs of Features with Perfect Correlation

Index A	Feature A	Index B	Feature B
6	mean IAT	198	mean IAT a b
6	mean IAT	205	mean IAT b a
198	mean IAT a b	205	mean IAT b a
7	q3 IAT	199	q3 IAT a b
7	q3 IAT	206	q3 IAT b a
199	q3 IAT a b	206	q3 IAT b a
217	Effective Bandwidth a b	218	Effective Bandwidth b a

Table 4.4: Uninformative Features

Index	Feature
76	urgent data pkts b a
78	urgent data bytes b a
103	truncated data a b
104	truncated data b a
106	truncated packets b a
219	FFT all Frequency # 1
229	FFT a b Frequency # 1
239	FFT b a Frequency # 1

As discussed in Section 3.3, the Bhattacharyya methods do not handle redundant features. These redundant and uninformative features in Tables 4.3 and 4.4 were removed from the Bhattacharyya methods to prevent the similar ranking of highly correlated features. All the remaining feature selection methods utilize all the features. Except for RELIEF-F, all the other feature selection methods handle redundant features. The redundant and uninformative features should also be removed from the RELIEF-F runs, but the RELIEF-F runs have been executed in advance of the analysis of redundant features. Due to the amount of computational time in reevaluating the features for 30 Monte-Carlo runs, the RELIEF-F algorithm was not repeated without the redundant and uninformative features. Since the redundant features would be ranked similarly, the subset selected using RELIEF-F may be larger than if the features were excluded. In the case of this thesis, the size of the selected subset for RELIEF-F was not affected. The redundant and uninformative features referenced in Tables 4.3 and 4.4 do not appear in the selected subset for RELIEF-F. Hence, there is no impact on the results of this thesis by not reevaluating the features with RELIEF-F.

4.1.2 Experimental Parameters. Cross validation allows for the use of independent training and test sets, while testing each sample in the set exactly once. A single run of k-fold cross validation provides an estimation or sample of the performance of the classifier, while reducing the bias of the results [46]. By stratifying the folds, each fold will have a near equal number of samples per class. Stratification ensures a representative number of of each class is in all training and test sets. There are ample samples to perform runs of a stratified 10-fold cross validation since the reduced data set has 40,858 samples with only 2 classes containing less than 1,000 samples. This thesis assesses the performance of a classifier for a given subset of features with 30 runs of stratified 10-fold cross-validation. A run ascribes to a specific seed that partitions the folds for the cross validation of each run. The same seeds, integers 1 through 30, are used for all classifications and feature selection methods to provide uniformity for comparison of performance.

Table 4.5 provides a listing of the parameters for the machine learning algorithms of interest to this work. The C4.5 decision tree parameters are based on Quinlan's discussion [70] as described in Section 2.3.1. Mucciardi and Gose [63] provide the POEACC parameters that perform well in their experiments. As suggested by Kononenko [38], the simplification of using all the instances to determine the weighting is used for RELIEF-F. The parameters for the remaining algorithms (GRLVQI, BP, classifier accuracy rate with best first (BF) search and classifier accuracy rate with genetic algorithm) are based on experimentation, in order to find a reasonable performance in classification performance and runtime. The GRLVQI learn schedule refers to a reduction in the learn rates by a half and the β by 20% every 250k training

Table 4.5: Parameters for Machine Learning Algorithms

Näive Bayes	None	
C4.5	Confidence Factor	0.25
	Min Num of Objects per Leaf	2
GRLVQI	In-Class Learn Rate (ϵ^J)	0.02
	Out-of-Class Learn Rate (ϵ^K)	0.02
	Relevance Learn Rate (ϵ^λ)	0.01
	Training Steps	2M
	Learn Schedule	Every 250k steps
	Number of Prototypes per Class	10
	β for Conscience Learning	0.5
	γ for Conscience Learning	2
BP	Learning Rate	0.3
	Training Epochs	10
	Hidden Nodes	248
POEACC	Classifier	Näive Bayes
	Weight for ACC	0.9
	Weight for POE	0.1
RELIEF-F	Number of Neighbors	All
Acc Rate w/ BF Search	Classifier	Näive Bayes
	Direction	Forward
	Amount of Backtracking	5 nodes
Acc Rate w/ Genetic Algorithm	Classifier	Näive Bayes
	Generations	20
	Crossover Probability	0.6
	Mutation Probability	0.3
Median Bhattacharyya	None	
Minimum Surface Bhattacharyya	None	

steps. The Näive Bayes, and median and minimum surface Bhattacharyya methods do not require input parameters. Although, the performance of the Bhattacharyya methods are sensitive to the binning approach taken for the comparison of histograms.

Implementations of the algorithms for this thesis may be found in the WEKA library [87]. WEKA is a Java library of machine learning algorithms for classification, clustering and feature selection. The library includes a graphical user interface, mechanisms for input and output of data sets, visualization of results and input data, testing of algorithms, and ample documentation. This tool set may be used to implement additional machine learning methods. A few algorithms, which were not included in the WEKA library (e.g., POEACC and GRLVQI), have been implemented to interface with the library and take advantage of the performance analysis and reporting tools. The median and minimum surface Bhattacharyya methods are implemented in Matlab, vice WEKA, to reduce development time. For example, Matlab contains predefined functions for generating histograms, which is used to calculate the Bhattacharyya coefficient.

4.2 Results of Experiments

This section discusses the results and analysis in establishing and evaluating feature subsets for intrusion detection. The subsets for each feature selection method are quantitatively justified. Feature selection methods, which develop the subsets, demonstrate varied capacities in identifying relevant features for the given data set. A few subsets well describe the decision boundary by improving or maintaining performance across a varied grouping of classifiers. Trends in the classifiers are also brought to light based on the comparative classification performances of the subsets.

4.2.1 Subset Selection for Each Feature Selection Algorithm.

By executing the method in Section 3.2.1, each feature selection method provides an ordered list of subsets for evaluation. The subsets of a given feature selection algorithm are referred to by the number of features in a subset. A single subset for each feature selection algorithm is chosen based on statistically-significant classification performance with the Näive Bayes classifier (statistical significance determined as discussed in Section 3.4.2). The leading subset where the classification performance plateaus is selected for the feature selection method for reasons discussed in Section 3.2.2. If a smaller subset performs statistically better or the same than the selected feature subset, then the smaller subset is selected at the expense of the larger subset. This section reviews the assessment of the performance for each feature selection algorithm. Figures A.1 through A.8 in Appendix A illustrate the results and analysis of the ordered subsets' classification performance from the following feature selection methods:

classifier accuracy rate with best first search, Decision Tree (C4.5), classifier accuracy rate with genetic algorithm, RELIEF-F, Probability of Error and Average Correlation Coefficient (POEACC), Generalized Relevance Learning Vector Quantization Improved (GRLVQI), median Bhattacharyya and minimum surface Bhattacharyya methods.

Each figure contains three plots to illustrate the results of the subsets' performances for a given feature selection method. The (a) plots show differences in classification performance (i.e., equal weighted accuracy (EWA)) between consecutive subsets. The difference is calculated by $EWA_{Subset\,(i+1)} - EWA_{Subset\,i}$, where i is the number of features in a given subset. When the difference stabilizes to below 1.5% accuracy, classification performance has reached a representative plateau based on empirical trends of the results. The leading subset of the plateau is initially selected as the selected subset for the given feature selection method. The (b) plots provide a multiple comparison of the mean performance of subsets with normally distributed results to demonstrate statistical significance. The (b) plots do not contain the confidence intervals because the intervals are too small for the given axis. The confidence intervals will be referred to in relation to the mean, μ_{EWA}, when a statistically-significant difference in performance is not clear. The (c) plots provide a multiple comparison of mean ranks with confidence intervals for comparison of non-normally distributed results. The rank confidence intervals will be referred to in relation to the mean rank, μ_{rank}. The family-wise rank confidence intervals may be overly conservative and misleading since the comparison method of mean ranks accounts for a pooled standard error in rank. Hence, analysis by rank negates large differences in performance. Given 3 accuracies of 0.6, 0.9 and 0.91, the ranks are 1, 2 and 3, respectively. If there is a significant overlap of numerous subsets of close performance the family-wise rank confidence interval becomes tremendous in comparison with the actual difference in performance. Alternate methods of analysis may be required for these cases (e.g., a paired hypothesis test or a multiple comparison of a smaller subset of features).

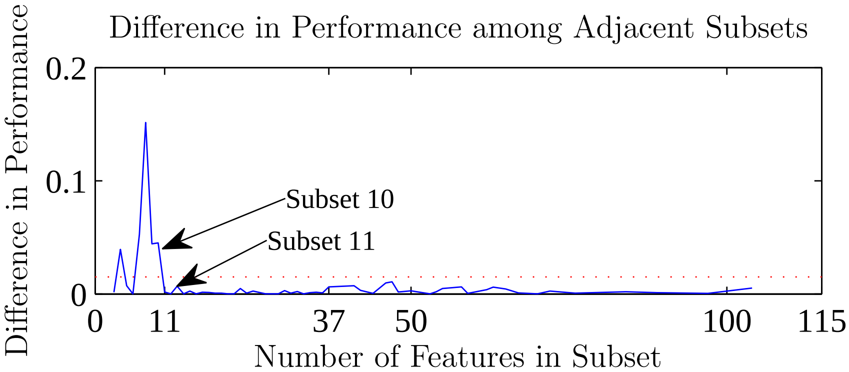


Figure 4.1: The classification performance of ordered subsets, generated by accuracy rate with best first search, stabilizes at subset 11. Based on the given feature selection and classification method, subset 11 contains the most important features

The results in Figure A.1 and interpretation for the classifier accuracy rate with best first search are discussed in detail below. The three plots in Figure A.1 are included in this section (e.g., Figures 4.1, 4.2, and 4.3). Figure 4.1 shows the difference in performance from adjacent subsets by the number of features in a given subset for classifier accuracy rate with best first search. The smallest subset generated contains three features and the classification performance changes significantly for successive subsets. The classification performance stabilizes below 1.5% starting with subset 11. Subset 11 is initially selected as the feature subset for classifier accuracy rate with best first search. Based on the given feature selection and classification method, subset 11 contains the most important features, but statistical analsyis determines whether it is the smallest set of important features.

For classifier accuracy rate with best first search, subset 11 is compared to smaller subsets to determine whether it provides improved classification performance with statistical significance. All results for feature subsets smaller than subset 11 are normally distributed, except for subset 10. Figure 4.2 shows the mean classification performance by the number of features in a subset. For the figure, the family-wise confidence interval is $\mu_{EWA}\pm 8.863\times 10^{-4}$. With such a tight interval, no subset smaller than subset 11 has a classification performance that approaches the performance of

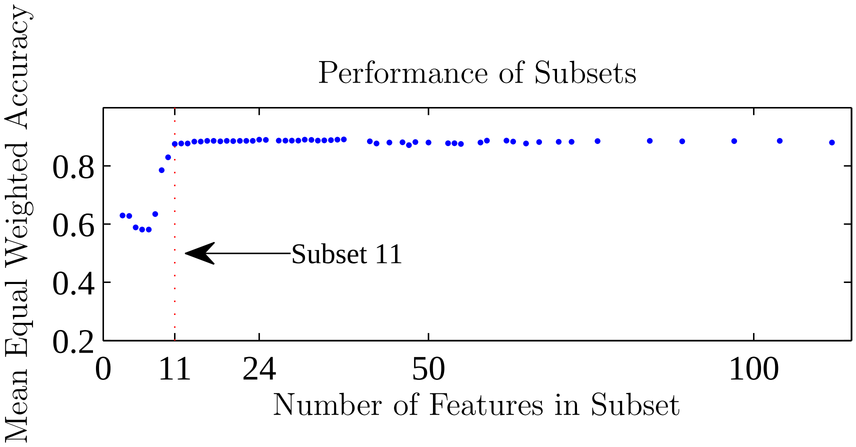


Figure 4.2: Multiple comparison of the mean classification performance is used to determine statistically-significant differences of normally distributed results for ordered subsets from accuracy rate with best first search

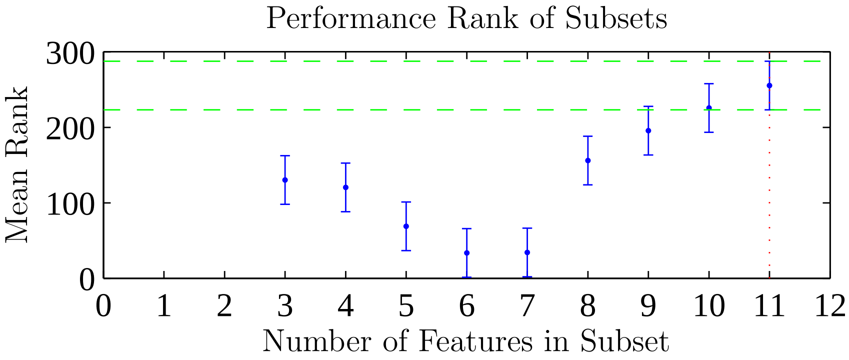


Figure 4.3: Multiple comparison of the mean ranks in classification performance is used to determine statistically-significant differences of non-normally distributed results for ordered subsets from accuracy rate with best first search. Subset 10 is the only non-normally distributed subset smaller than Subset 11

subset 11. Hence, the classification performance results from subset 11 perform better than all the smaller, normally distributed subsets with statistical significance.

Subset 10 is non-normally distributed so analysis of statistical significance using rank is appropriate. Figure 4.3 illustrates the rank-based multiple comparison with confidence intervals. Based on the classification performance rank intervals, subsets 10 and 11 clearly overlap so the results are not statistically different from this perspective. Based on the large difference in mean EWA, a lack of statistical significance is counterintuitive. The family-wise rank confidence interval in this case is too conservative to make an accurate determination of statistical significance. A paired Wilcoxon signed rank test shows the results for subsets 10 and 11 do not have equal classification performance with statistical significance, p-value $= 1.7235 \times 10^{-6}$. Based on the given feature selection and classification method, subset 11 contains the smallest set of important features.

The analysis for the remaining figures (feature selection methods) are analogous to that of the classifier accuracy rate with best first search method. They are summarized below:

- Decision Tree (C4.5): Figure A.2 (a) illustrates that the results stabilize at subset 23. Figure A.2 (b) shows the mean performances are clearly less than subset 23 for smaller subsets. Figure A.2 (c) shows no overlap below subset 23.
- Classifier accuracy rate with Genetic algorithm: Figure A.3 (a) illustrates that the results stabilize at subset 12. Figure A.3 (b) shows the mean performances are clearly less than subset 12 for smaller subsets. Figure A.3 (c) shows no overlap below subset 12.
- RELIEF-F: Figure A.4 (a) illustrates that the results stabilize at subset 19. Figure A.4 (b) shows the mean performances are clearly less than subset 19 for smaller subsets. Figure A.4 (c) shows no overlap below subset 19.
- POEACC: Figure A.5 (a) shows the performance stabilize at subset 86. Due to the poor performance of the feature selection method, many redundant and uninformative features are ranked high. One must relax the 1.5% difference among adjacent subsets to obtain a comparably-sized subset that captures the most relevant features. Subset 12 may be selected with a difference of approximately 5%. Figure A.5 (b) shows the nearest performer to subset 12 is subset 11 and a statistically-significant difference is not clear. The confidence interval around the mean is $\mu_{EWA} \pm 1.4 \times 10^{-3}$. The upper and lower ends of the confidence intervals for subsets 11 and 12 are $\mu_{EWA} + interval = 0.6783$ and $\mu_{EWA} - interval = 0.6904$, respectively. Hence, there is no overlap of the inter-

vals. Subsets one to four have non-normally distributed results. The multiple comparison among all the subsets is too conservative so Figure A.5 (c) shows a multiple comparison between subset 12 and smaller subsets. There is no overlap with subsets one to four.

- GRLVQI: Figure A.6 (a) illustrates that the results stabilize at subset 27. Figure A.6 (b) shows the mean performances are clearly less than subset 27 for smaller subsets. Non-normally distributed results include subsets 1-5, 6, and 12. Figure A.6 (c) shows no overlap with subsets 1-5, 6, and 12 with subset 27.
- Median Bhattacharyya: Figure A.7 (a) illustrates the results do not stabilize until after a severe reduction in performance at subset 71. The subset with the best classification performance below subset 71 is subset 23. Figure A.7 (b) is unclear whether there is an overlap of the confidence intervals. The nearest performer to subset 23 ($\mu_{EWA} - interval = 0.8692$) is subset 22 ($\mu_{EWA} + interval = 0.8668$) and there is no overlap of the intervals. The non-normally distributed results below subset 23 include subsets 1, 2, 6, and 8. Figure A.7 (c) shows no overlap of rank confidence intervals between subset 23 and the non-normally distributed results.
- Minimum Surface Bhattacharyya: Figure A.8 (a) illustrates that the results stabilize at subset 16. Figure A.8 (b) shows the mean performances are clearly less than subset 16 for smaller subsets. Figure A.8 (c) shows no overlap of subset 16 with non-Normal results from subsets 1, 5, and 9.

Table 4.6 compares the selected subset sizes for each feature selection method discussed and subsets generated from two additional feature selection methods from Auld [2] and Moore [60]. The two additional subsets were generated, by Bayesian Neural Network Feature-Interdependent Ranking (BNN) and Fast Correlation-Based Filter (FCBF) methods, from the features in Appendix B, except port numbers. All of the feature selection methods drastically reduce the dimensions of the feature set.

Section 4.2.2 compares the performance of the subsets with four classifiers: Näive Bayes, C4.5, multilayer perceptron with backpropogation (BP), and GRLVQI.

Table 4.6: Performance of Selected Subsets

Feature Selection Method	Number of Selected Features
Best First	11
Decision Tree (C4.5)	23
Genetic	12
RELIEF-F	19
POEACC	12
GRLVQI	27
BNN [2]	20
FCBF [2, 60]	10
Bhattacharyya-median sort	23
Bhattacharyya-min surface sort	16
None (All Features)	248

4.2.2 Subset Comparison by Classifier. Figures A.9 through A.12 in Appendix A illustrate the multiple comparisons of the selected subsets' performances for the Näive Bayes, C4.5, BP, and GRLVQI classifiers. The analysis treats the performance of all the features for a given classifier as a baseline. Performance is based on an improvement or decline in EWA for the subsets selected for each feature selection method in relation to the baseline. The (a) plots show comparison of means with family-wise confidence intervals. The (b) plots show mean ranks with family-wise rank confidence intervals. A similar issue with the multiple comparison of ranks results in overly-conservative rank confidence intervals for non-normally distributed results. A paired hypothesis test may be required for non-normally distributed results. The subsets generating non-normally distributed results for a given classification method are listed below:

- Näive Bayes: C4.5 and FCBF
- C4.5: Accuracy rate with genetic algorithm and median Bhattacharyya
- BP: none

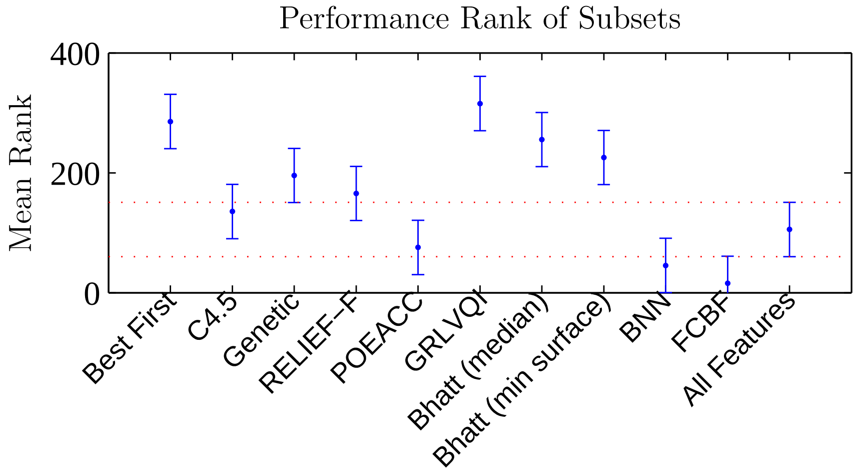


Figure 4.4: Multiple comparison of the mean ranks in Näive Bayes classifier performance is used to determine statistically-significant differences of non-normally distributed results for selected subsets. The method is overly conservative for the non-normally distributed results from C4.5 and FCBF feature selection methods

- GRLVQI: C4.5

The only case, where the figures do not provide an accurate representation of the results, is for the non-normally distributed results for the Näive Bayes classifier in Figure 4.4. The results generated by subsets from C4.5 and FCBF feature selection methods are non-normally distributed. Wilcoxon signed rank tests between the results of C4.5 and *All Features*, and FCBF and *All Features* show unequal performance. Each of the two tests has a p-value equal to 1.73×10^{-6}. The subset from the C4.5 method has a better performance and the FCBF has a worse performance than the baseline of *All Features*.

As indicated by Table 4.7, most of the feature selection methods provide a degree of generality in the selected subsets. The far-left column of the table lists the feature selection method from which a given subset is generated. The remaining two columns provide a count of the number of classifiers, where the selected subset has outperformed or maintained performance in respect to all the features with statistical significance. Only the subset generated by the RELIEF-F algorithm improved or maintained performance for all of the classifiers. The RELIEF-F algorithm weights the features based on the nearest distance from in-class and out-of-class samples along a given dimension (feature). Interestingly, GRLVQI updates feature weightings based on distances to the nearest in-class and out-of-class samples and its selected subset provides good generality by improving or maintaining performance for three out of the four classifiers. Surprisingly, the subset generated from GRLVQI improved performance for every classifier except itself. Ranking the features, based on the Bhattacharyya coefficient, provides a slightly different distance measure approach, which seeks to determine the difference between classes for a given feature. Both methods utilizing Bhattacharyya coefficients improve performance for three classifiers. Regardless of specific approach, this shows distance to be a reliable measure for analyzing features for this data set. The methods that do not perform well for the intent of generality is classifier accuracy rate with best first search, POEACC, and the two methods taken from papers, BNN and FCBF. The feature selection method utilizing accuracy rate with best first search improved performance for a single classifier (Näive Bayes), while maintaining classification performance for another classifier (C4.5). Even though POEACC improved and maintained classification performance for two classifiers (GRLVQI and C4.5, respectively), POEACC performed near the worst of the feature selection methods across all of the classifiers. The subsets from BNN and FCBF clearly do not provide generality by improving classification performance for one classifier (GRLVQI). The methods do not to include server and client ports in their subsets (BNN and FCBF), which demonstrates the importance of ports for the classification task.

Nearly every selected subset increased performance for the Näive Bayes classifier as illustrated in Figure A.9. The feature subsets not containing port numbers, BNN and FCBF, reduced performance approximately 30%. For the most part, the performance of the Näive Bayes classifier is highly susceptible to the included features. Even though most feature subsets increase performance, the increase is highly varied from 3% to 12%. The method utilizing accuracy rate with best first search performs well in optimizing a minimal set of features (11) and increases the Näive Bayes classifica-

Table 4.7: Performance of Selected Subsets in Relation to All Features

Feature Selection Method	Outperform (Number of Classifiers)	Equal Performance (Number of Classifiers)
Best First	1	1
Decision Tree (C4.5)	3	0
Genetic	2	1
RELIEF-F	2	2
POEACC	1	1
GRLVQI	2	1
BNN [2]	1	0
FCBF [2, 60]	1	0
Bhattacharyya-median sort	3	0
Bhattacharyya-min surface sort	3	0

tion performance by 10%. This is entirely reasonable since the heuristic used in the search is the accuracy from the Näive Bayes classifier. This is an example of a feature selection method optimizing for a given classifier. One may suspect that classifier accuracy rate with a genetic algorithm would result in a similar or better optimization for the Näive Bayes classifier. This likely do not occur because of the small number of generations and high mutation rate used for parameters. These parameters are not ideal for optimizing the search; they are used to reduce the search time and broaden the search space within the limited number of generations. In this case, the genetic search parameters result in a the smallest subset with good generality.

Figure A.10 shows the C4.5 classifier maintain outstanding performance across all the subsets. Even the worst performers (e.g., BNN and FCBF), have EWAs greater than 90%. Only the subsets generated from the Decision Tree and Bhattacharyya selection methods improved performance. The C4.5 classifier is robustly consistent regardless of the selected subset with most subsets providing EWA performance in excess of 98%.

The BP is stridently resilient to improvement in classification performance. Figure A.11 shows that no feature subset improved classification performance, which is consistent with Auld [2]. Two subsets (e.g., RELIEF-F and GRLVQI) maintained performance with the entire feature set. Three other subsets (e.g., accuracy rate with best first search, Decision Tree, and accuracy rate with a genetic algorithm) provided mean performance within 2% of that of all the features. The subsets generated by mean and minimum surface Bhattacharyya methods had performance within 4% of all the features. As discussed in Section 2.2.2, BP performs well with noisy and incomplete data so the consistent classification performance from the subsets demonstrates the traits of the classifier.

Performance of GRLVQI is dramatically improved by feature selection with this data set as illustrated by Figure A.12. This susceptibility to the feature set is not intuitive since GRLVQI preferentiates features based on classification performance. Two subsets hindered performance to include the subset generated by GRLVQI as the feature selection method and accuracy rate with best first search. The results suggest that the relevance initialization or update approaches for GRLVQI is suboptimal for its own learning paradigm with this data set. Perhaps, initially weighting the features by their normalized median Bhattacharyya measure between classes or RELIEF-F weighting would be beneficial in guiding the the creation of the domain boundary, vice an equal weighting for all features.

Table A.1 lists the features by the feature selection method. The most commonly selected features include server and client ports, size of the packets/segment sizes, and window sizes/advertisements. Less commonly selected features include Internet Protocol (IP) indicators (e.g., synchronization, acknowledgment, request packets), and timing (e.g., inter-arrival time, round trip time, and idle time). An important question to answer is why the selected features improve performance. Feature selection methods identify features, which characterize unique connection traits of the network traffic classes and provide separability between the classes for classification. For example, the subset generated via RELIEF-F demonstrated generality for all the classifiers and the features provide separability among the classes, as indicated by their Bhattacharyya coefficient. Of the features selected using RELIEF-F, 16 out of 19 features have mean Bhattacharyya coefficients less than 0.5 for all combinations of classes.

Additionally, the results demonstrate that the RELIEF-F algorithm performs poorly in handling highly correlated features. The RELIEF-F rankings of highly correlated data are similar.

In general terms, the selected features provide a relevant representation of the domain space and boundaries as reinforced by research into the characteristics of network traffic. Many of the selected features may be deduced based on the behavior of flow types. A more detailed discussion on established flow characteristics and their validation of the generality of the selected features follows.

4.3 Validation of Results

Self-identification of network service via client and server ports is ranked high for every feature selection method. For good reason, the Internet Assigned Numbers Authority (IANA) has defined well-known server ports (server ports less than 1,024) to a defined network service. Furthermore, IANA registers ports for proprietary applications and those server ports range from 1,024 to 49,151 [51]. A server configured appropriately will accept connections for a given service on the predefined port. Hence, the server port feature will correctly define the application in most cases. An exception is for the FTP protocol where the server initiates the connection for data transfer on port 20. In this case, the "server" sending the data is considered a client in respect to the connection and port 20 is assigned to the client port feature of a FTP-data flow. In addition, network attacks target specific services (e.g., the deloader worm and MS SQL-snake worm connect on ports 445 and 1433, respectively) so the port number may also be used in identifying the type of attack or conceal the attack among regular traffic [88]. Some peer to peer (P2P) traffic utilize predefined ports (e.g., BitTorrent operates on server ports 6881 to 6889). Other P2P applications have grown more elusive and operate on any port number and attempt to conceal the application by operating on port 80 [34]. Reliance on port numbers will lead to misclassifications and studies have shown other important characteristics to describe traffic.

The foremost characteristic of P2P traffic aside from port numbers is packet size [34] (e.g., features 11-23, 81-84, 153-166, 174-187). Differing P2P applications implement varied packet sizes for control packets to manage and search its overlay network. The P2P control packet size may be an indicator, aside from maintaining a high rate of packets that meet the maximum segment size. P2P networks are notorious for utilizing a significant amount of bandwidth over the Internet backbone. A likely indicator is the amount of bytes sent and the effective bandwidth of a flow (e.g., features 43-44, 47-48, 216-218).

Lakhina, *et al.* [41] provide ample discussion on the characteristics of network anomalies, of which attacks are a subset. Denial of Service (DOS) attacks become evident by the number of packets in a flow and the number of flows between a host pair. Furthermore, DOS attacks typically target a given IP address or small set of IP addresses. A high number flows between hosts are indicative of features that monitor the number of packets (e.g., features 31, 32) and a low time since a last connection, feature 209. The high rate of packets in a given flow suggests a low inter-arrival time (e.g., features 3-9, 195-208) and low idle time (e.g., features 109, 110).

A large number of IP flows to multiple ports is characteristic of scanning. Like a DOS attack, a low time since last connection is indicative of multiple connections between two hosts. Scanning is also a short-lived activity that takes advantage of undocumented events in network protocols to evade firewalls. For example, a connection may be initiated with an acknowledgment packet, vice the expected synchronization packet. Informative features may be stream length or duration (e.g features 99-100, 107-108). The data set strictly contains full TCP connections that begin with a synchronization packet so such methods of scanning are not included. Self-propagating worms target a given vulnerability over a set port with a large spike in packets to spread the worm. The features discussed should provide fidelity in characterizing worms.

The features do not provide indicators for analyzing unsuccessful or incomplete connections. These characteristics correspond to attacks such as synchronization flooding, scans and network mapping. A problem from the intrusion detection perspective is that the feature set focuses on flow-level characteristics. Ren, *et al.* [72] shows how network-level trends of synchronization and acknowledgment packets, time, port numbers and IP addresses may be used in intrusion detection.

4.4 Summary

Only a single algorithm, RELEIF-F, demonstrate generality of the selected feature set based on classification performance on all four classifiers. Five other feature selection methods (e.g., Decision Tree, accuracy rate with a genetic algorithm, GRLVQI, mean Bhattacharyya, and minimum surface Bhattacharyya methods) show good generality by maintaining classification performance on three of the classifiers and a majority of them nearly met performance for a fourth classifier. The decision boundary may be well described with 12 to 27 features, while providing significant improvement to classification performance in some cases. The proposed minimum surface Bhattacharyya method generates the second smallest subset of 16 with good generality. Accuracy rate with genetic search creates the smallest subset with good generality.

The (a) plots from Figures A.3 through A.8 in Appendix A demonstrate that the feature selection algorithms (e.g., accuracy rate with genetic algorithm, RELIEF-F, GRLVQI, median Bhattacharyya and minimum surface Bhattacharyya methods) that generated features with generality displayed a unique behavior. The performance reaches a climax with small subsets sizes and tappers off to a worse performance with large subset sizes. The feature selection methods that do not demonstrate generality in their selected features (e.g., accuracy rate with best first search and POEACC) illustrate different characteristics in their subset performance charts. Plot (a) of Figures A.1 and A.5 show a consistent increase or stable performance as the ordered subsets become larger. The climax and plateau illustrates an appropriate ordering of the most relevant features first followed by features that serve as distractors or redundant features. Figure A.5 for the POEACC algorithm illustrates a poor performing feature selection algorithm for the data set with the performance increasing until large subsets. It would be intriguing to observe the performance by subset graphs of the other classification methods. Is the climax an attribute of the ordered features or the Näive Bayes classifier? Do the performances plateau at similar subsets sizes? Additionally, one may rank the selected features in Table A.1 and assess the ordered subsets as discussed in Section 3.2.

V. Conclusions

The goal of this thesis is to assess relevant features for the purpose of intrusion detection on a computer network. The relevant features may allow for improved classification performance for real-time or post-analysis of network traffic. In order to perform the task, a review of the taxonomy of feature selection algorithms and intrusion detection methods are discussed. Prevalent methods in the machine learning intrusion detection field of study are incorporated into the methodology to include a diverse group of feature selection and classification methods. Additionally, an extension to utilizing separability as a feature selection method is proposed. The thesis outlines a set of features and feature types relevant to the intrusion detection task and provides quantitative justification by means of classification performance and qualitative validation from works characterizing network traffic. Specific features have been demonstrated to provide ample characterization of the decision boundary for classification. The feature selection methods analyzed include accuracy rate with best first search, Decision Tree (C4.5), accuracy rate with a genetic algorithm, RELIEF-F, Probability of Error and Average Correlation Coefficient (POEACC), Generalized Relevance Learning Vector Quantization Improved (GRLVQI), median Bhattacharyya and minimum surface Bhattacharyya methods.

5.1 Summary of Results

For a selection of real traffic flows, the results of this thesis demonstrate multiple feature subsets that drastically reduce the size of a feature set while providing ample generality to describe the domain and enhance performance for some classifiers. The figures in Appendix A illustrate several trends for the feature selection and classification methods like confirmation of the hypothesis that classification performance plateaus with added features. Six feature selection methods provide subsets with a degree of generality. Although, if one applied a distinct approach for selecting larger subsets, from the feature selection methods, generality would be found with more feature selection methods. The RELIEF-F feature selection method generates a subset with generality across all the classifiers. The minimum surface Bhattacharyya method generates the second smallest set of features to describe the decision boundary well. Where the POEACC algorithm regularly performs near the bottom of the pack. Accuracy with best first search demonstrates its capacity in selecting the smallest feature set (11), while optimizing the performance of a given classifier. Based on review of the selected feature subsets, the key feature characteristics required for generality include ports, packet size, timing attributes, and other protocol-specific indicators. Analysis of evaluations into network traffic characteristics support the finding of this thesis.

Unintentionally, the thesis also acts as a survey of common classification methods. Contrary to some prior work, the C4.5 demonstrates very good classification across all of the network traffic classes. Previous work implementing C4.5 and other decision trees show mixed results in classifying network services and attacks on simulated and real traffic [1, 21, 43]. A majority of the works utilize the Massachusetts Institute of Technology Lincoln (MIT) Laboratory data set discussed in Section 4.1.1 and show relatively poor accuracy for all attack types except Denial of Service attacks. Sabhnani [77] argues that the poor detection performance of some attack types is due to deficiencies in the MIT data set. This thesis shows a clear performance advantage for C4.5 classifiers, versus alternate methods, with an equal weighted accuracy performance of 98%. The attack types, in this thesis, consitute worms and viruses at a high accuracy, where Agosta [1] demonstrates classification accuracies ranging from 57% to 88% in detecting worms on a homemade data set.

5.2 Contributions

This work provides several contributions to the field of intrusion detection and machine learning. First, generality has been shown for a relatively small subset of features to describe the classification decision boundary of Transmission Control Protocol flows. A second smallest set of 16 features may be extracted in real-time or near real-time to improve the performance of anomaly detection methods utilizing varied machine learning techniques. Second, the C4.5 decision tree classification method has been shown to perform well on real traffic with worms and viruses consisting of the labeled attack traffic. The decision tree may be simply interpreted to develop fast and understandable rule sets for implementation in an expert intrusion detection system. Lastly, an extension, minimum surface Bhattacharyya method, to a current feature selection method demonstrates comparable performance with other methods, while minimizing the size of the feature subset. The minimum surface Bhattacharyya method takes advantage of the intrinsic separability with a data set for multiple class problems, optimizing selection of the most separable features across all class-pair combinations. Additionally, the minimum surface Bhattacharyya method runs with a low time complexity in comparison to other feature selection methods.

5.3 Recommendations for Future Work

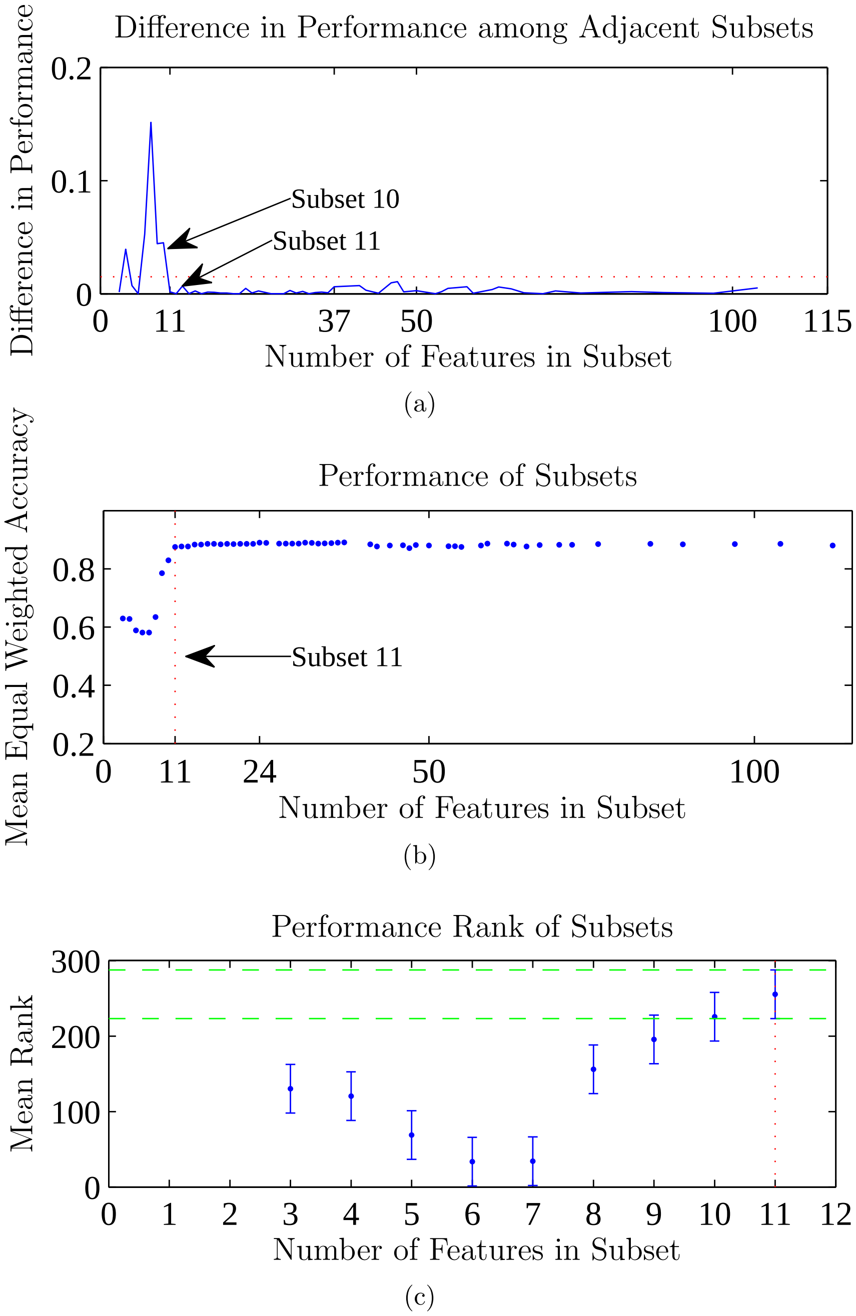
To improve upon current classification and feature selection methods for intrusion detection, a data set is required with ground truth knowledge of the sub-classes constituting the attack class. Development of a data set containing, a broad spectrum of attack types would be ideal. Flows of known attacks may be monitored on an isolated network to obtain the required breadth. Then, the flows from varying sources could be combined to develop a data set with a combination of wild, genuine flows of network services and a plethora of attack types.

In regards to the classification algorithms implemented, there are several unanswered questions which may improve understanding of the algorithms and the domain. For the C4.5 algorithm, there are multiple types of evaluation measure and pruning methods that may be implemented. Is there an optimal measure for segmenting the network flow domain space for classification? The feature evaluation methods showed class-based distance to be effective. For the GRLVQI algorithm, evaluation of alternative feature relevance update methods appears to be warranted. A performance comparison of all features with subsets in Figure A.12 illustrates improved performance for multiple subsets selected from alternate means.

Numerous papers [11, 62, 81] argue that machine learning provides an effective anomaly detection capability. The capability of machine learning algorithms to truly detect anomalous behavior has not been shown empirically. A basic premise for machine learning classifiers requires testing on unseen samples to provide an estimation of the performance of a learned model. A rarely tested hypothesis is the capability to identify unseen attack types [42]. One may develop a training and test set, where the test set contains attack types that are left out of the training set. Successful classification with the feature subset of unseen attacks provides strong evidence of the functionality and the robustness of a feature set and classification technique.

Lastly, the burden of obtaining genuine, labeled network traffic and attacks for analysis would likely be too difficult or involved a task for a production anomaly detector. Additionally, detecting traffic in the wild and trying to label after the fact induces error and limits the traffic to that observed over a given network. The analysis of the effectiveness of training on simulated traffic would be of great benefit in employing a production anomaly detection system based on machine learning. This effort would require work in developing traffic models and validating the effectiveness of classifiers trained on simulated traffic by testing on real network traffic.

Appendix A. Feature Selection Results

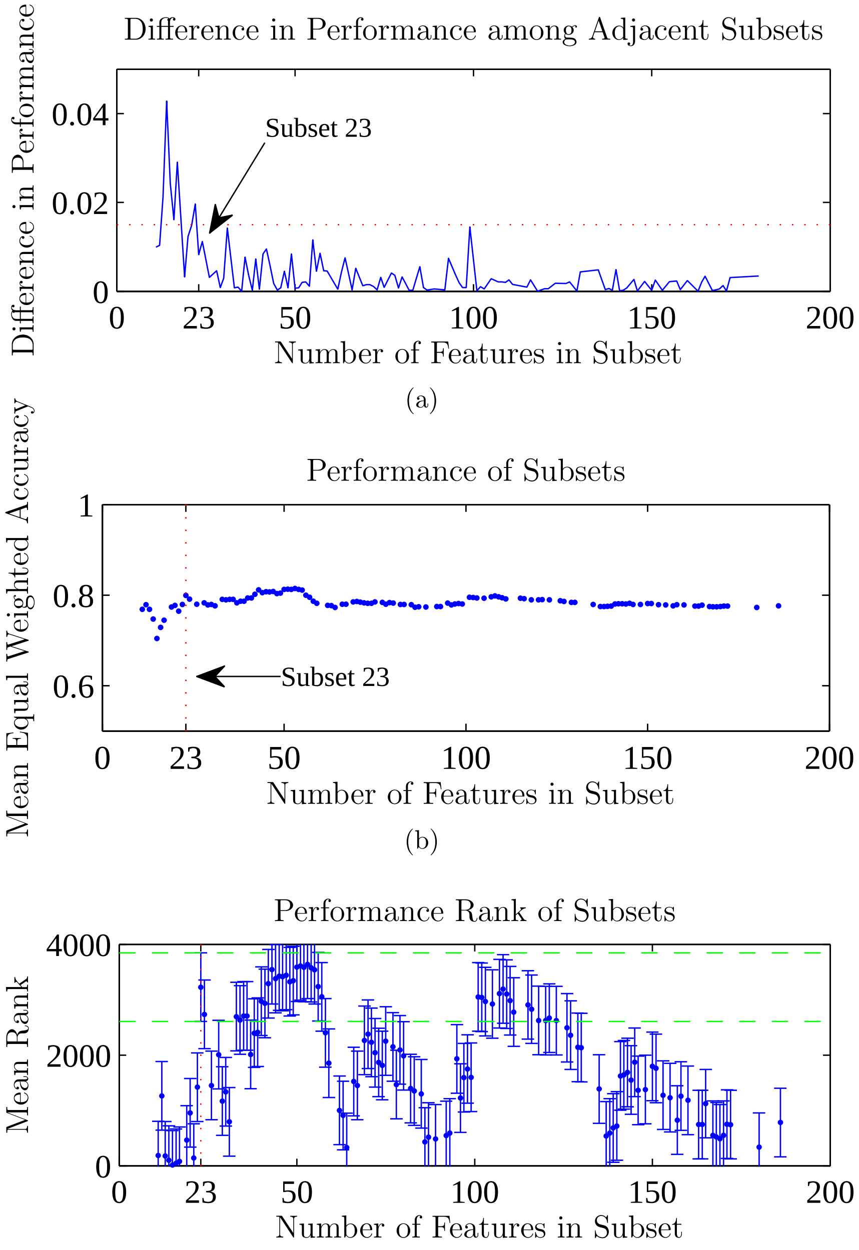


(a)

(b)

(c)

Figure A.1: Feature Selection: accuracy rate with best first search. Analysis of the plots determines a single subset for accuracy rate with best first search. Plot (a) illustrates where the performance plateaus. Plot (b) and (c) provide a multiple comparison of classification performance for determining statistically-significant differences for normally and non-normally distributed results, respectively

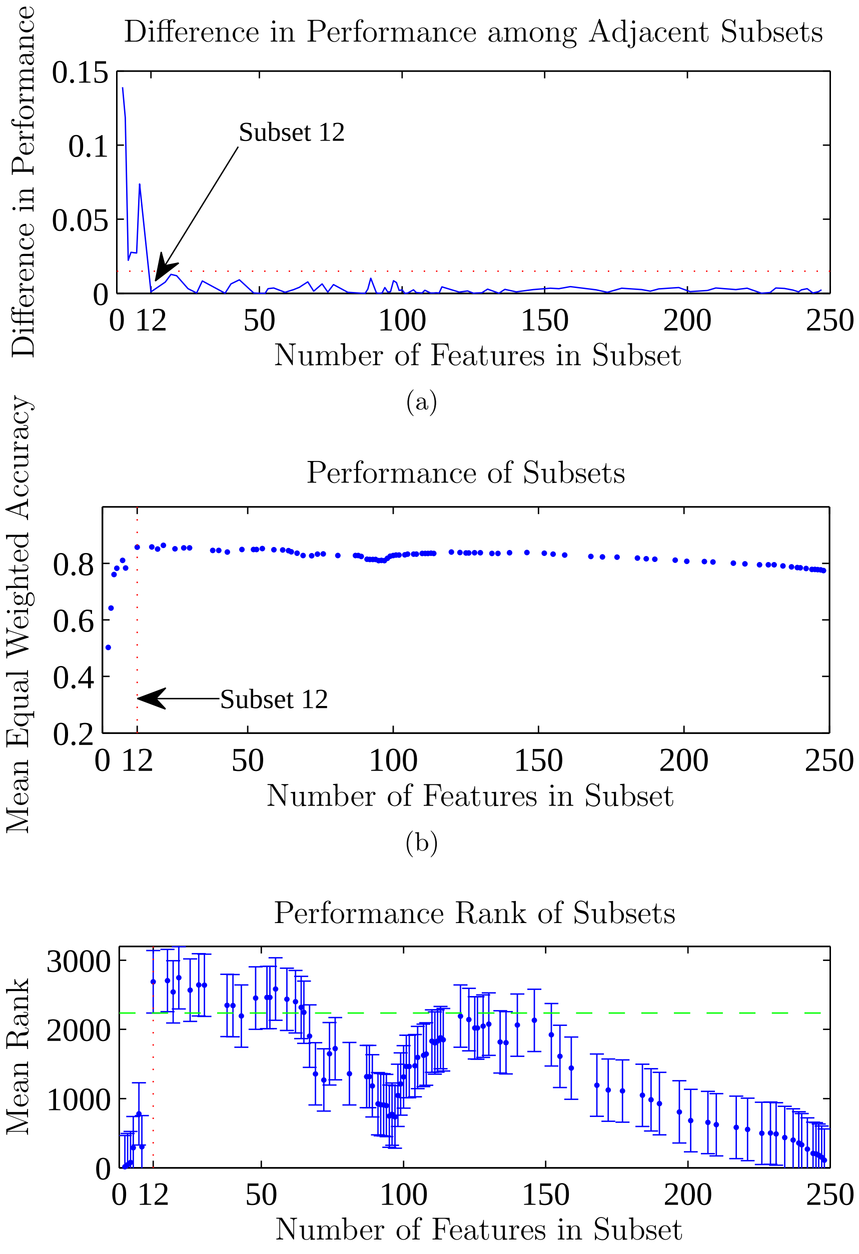


(a)

(b)

(c)

Figure A.2: Feature Selection: decision tree method. Analysis of the plots determines a single subset for decision tree method (C4.5). Plot (a) illustrates where the performance plateaus. Plot (b) and (c) provide a multiple comparison of classification performance for determining statistically-significant differences for normally and non-normally distributed results, respectively

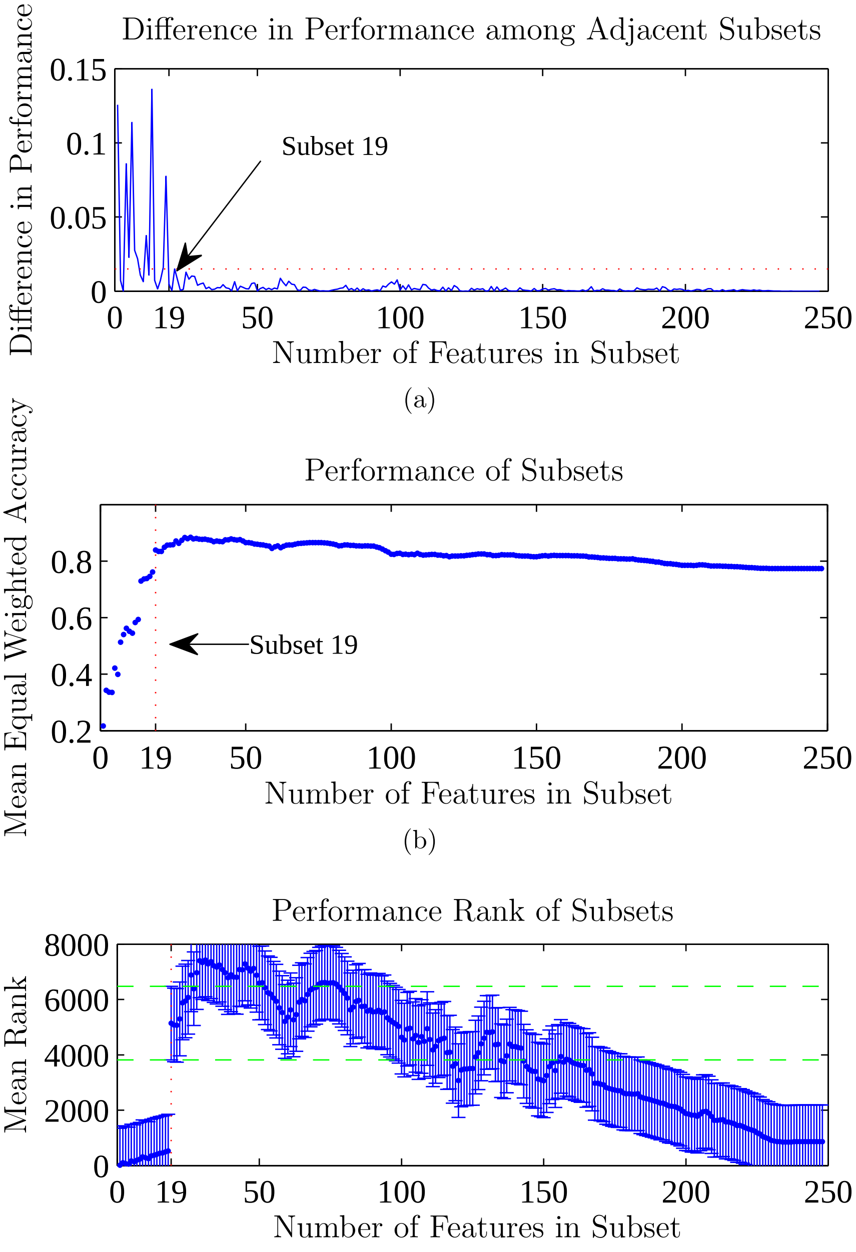


(a)

(b)

(c)

Figure A.3: Feature Selection: accuracy rate with genetic algorithm. Analysis of the plots determines a single subset for accuracy rate with genetic algorithm. Plot (a) illustrates where the performance plateaus. Plot (b) and (c) provide a multiple comparison of classification performance for determining statistically-significant differences for normally and non-normally distributed results, respectively

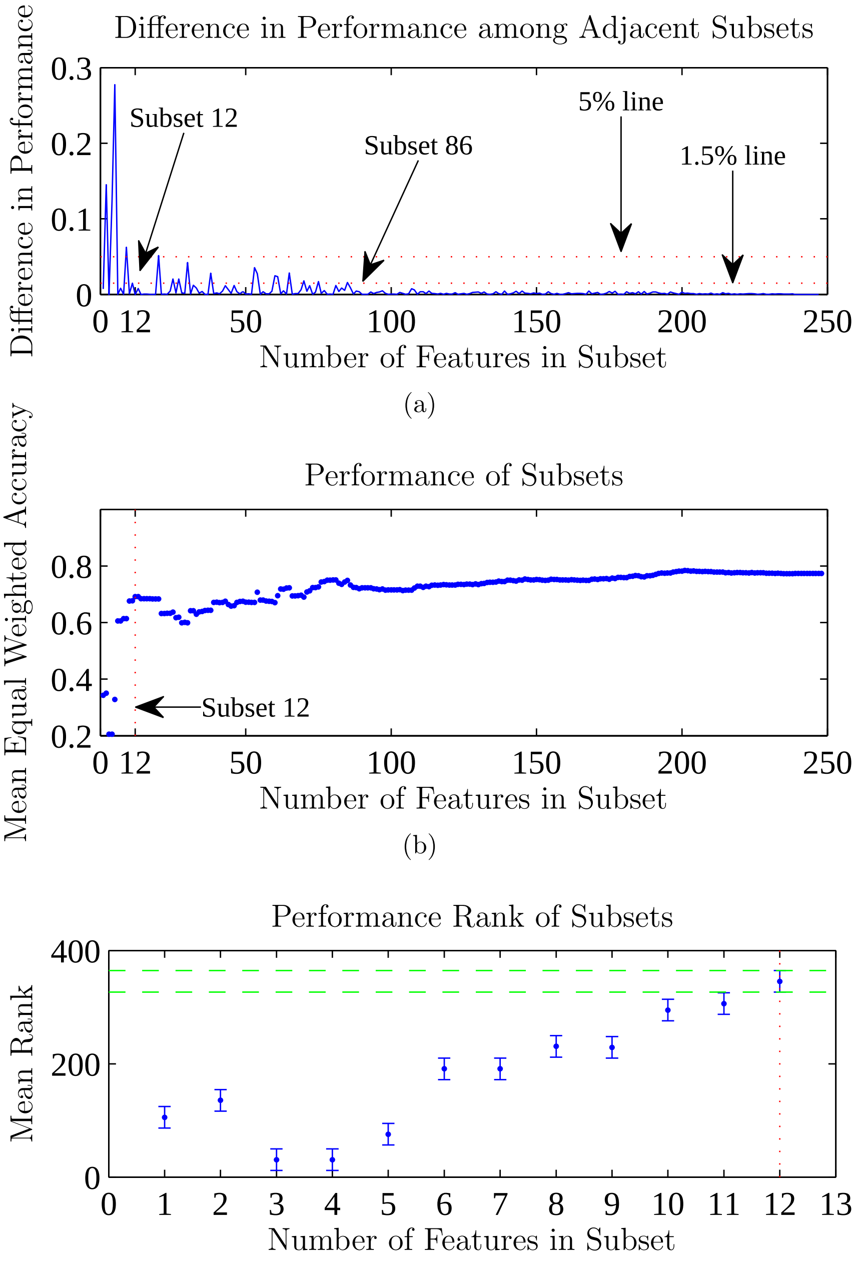


(a)

(b)

(c)

Figure A.4: Feature Selection: Analysis of the plots determines a single subset for RELIEF-F. Plot (a) illustrates where the performance plateaus. Plot (b) and (c) provide a multiple comparison of classification performance for determining statistically-significant differences for normally and non-normally distributed results, respectively

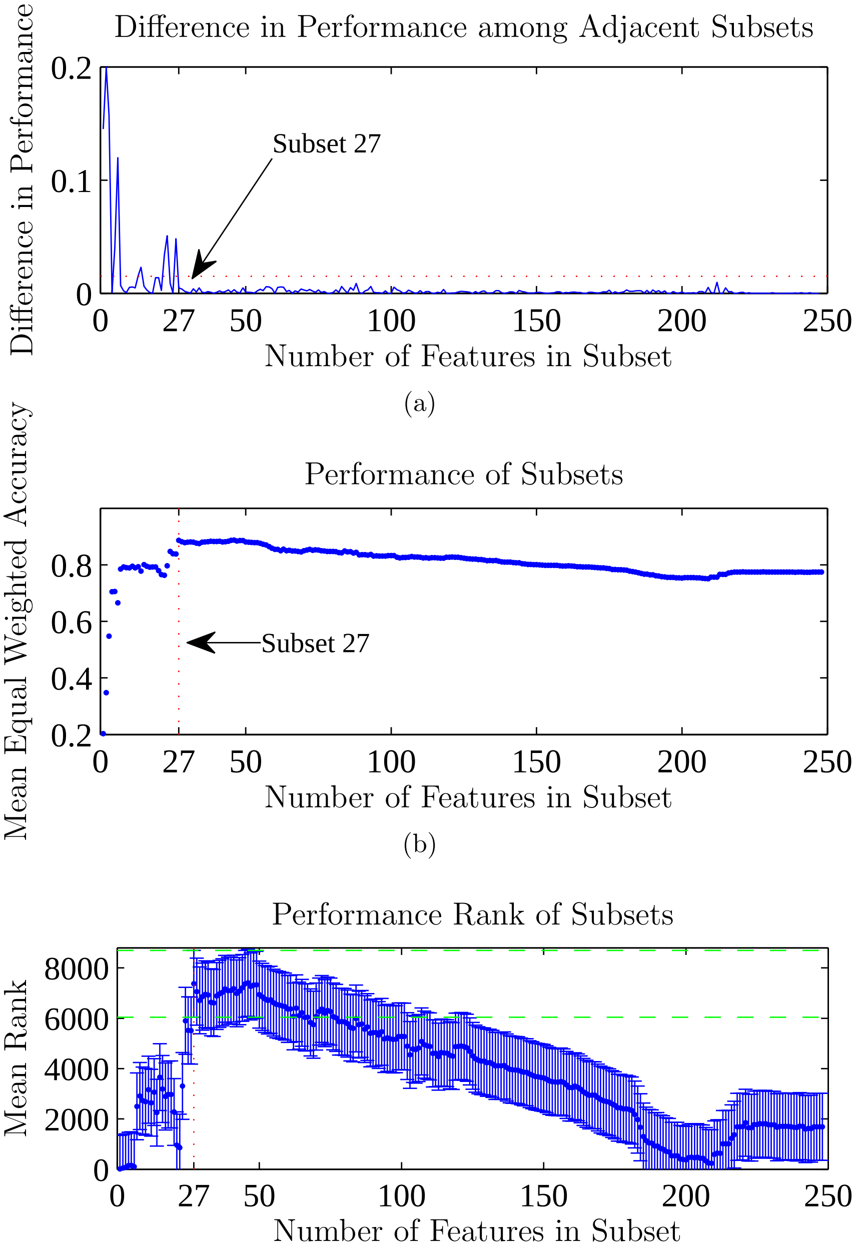


(a)

(b)

(c)

Figure A.5: Feature Selection: probability of error and average correlation coefficient (POEACC). Analysis of the plots determines a single subset for POEACC. Plot (a) illustrates where the performance plateaus. Plot (b) and (c) provide a multiple comparison of classification performance for determining statistically-significant differences for normally and non-normally distributed results, respectively

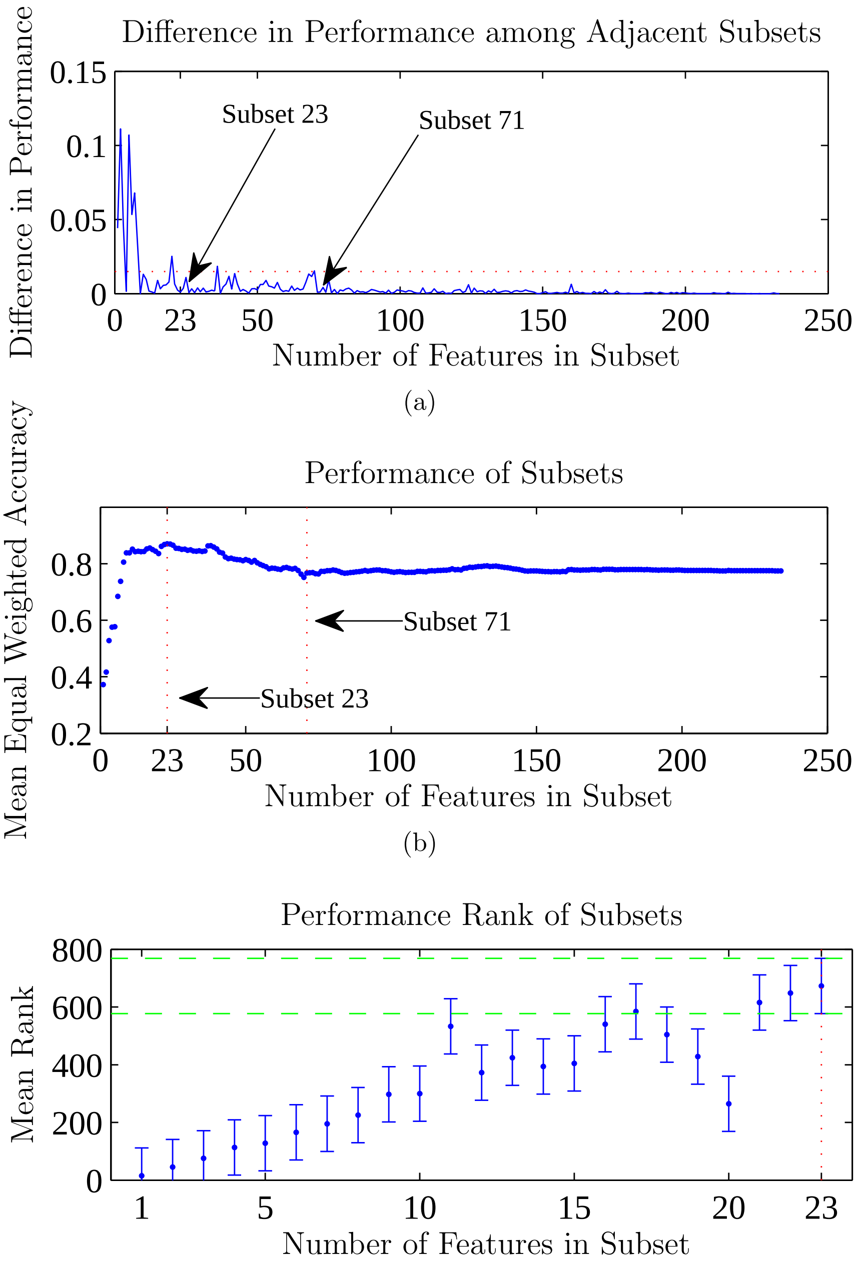


(a)

(b)

(c)

Figure A.6: Feature Selection: Generalized Relevance Learning Vector Quantization Improved (GRLVQI). Analysis of the plots determines a single subset for GRLVQI. Plot (a) illustrates where the performance plateaus. Plot (b) and (c) provide a multiple comparison of classification performance for determining statistically-significant differences for normally and non-normally distributed results, respectively

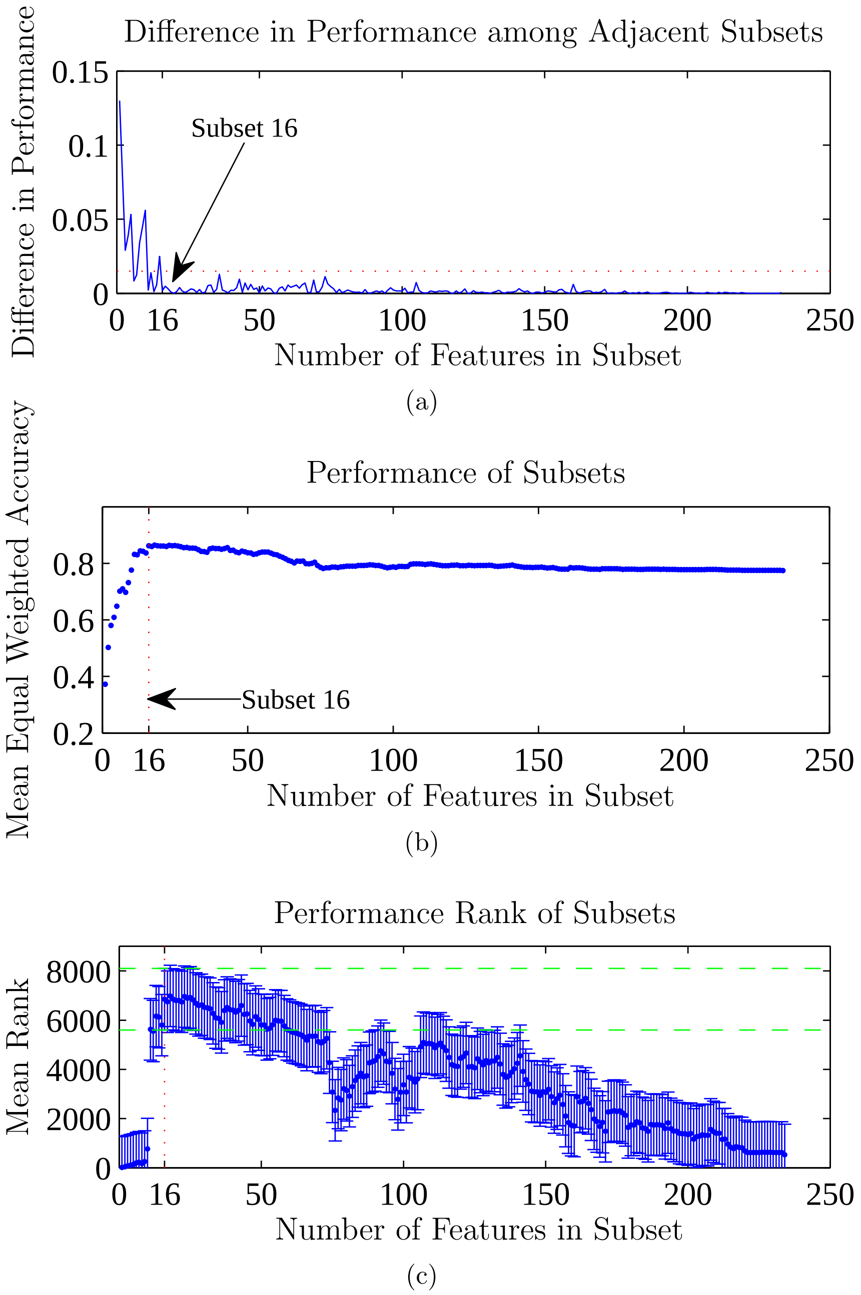


(a)

(b)

(c)

Figure A.7: Feature Selection: Median Bhattacharyya. Analysis of the plots determines a single subset for Median Bhattacharyya. Plot (a) illustrates where the performance plateaus. Plot (b) and (c) provide a multiple comparison of classification performance for determining statistically-significant differences for normally and non-normally distributed results, respectively

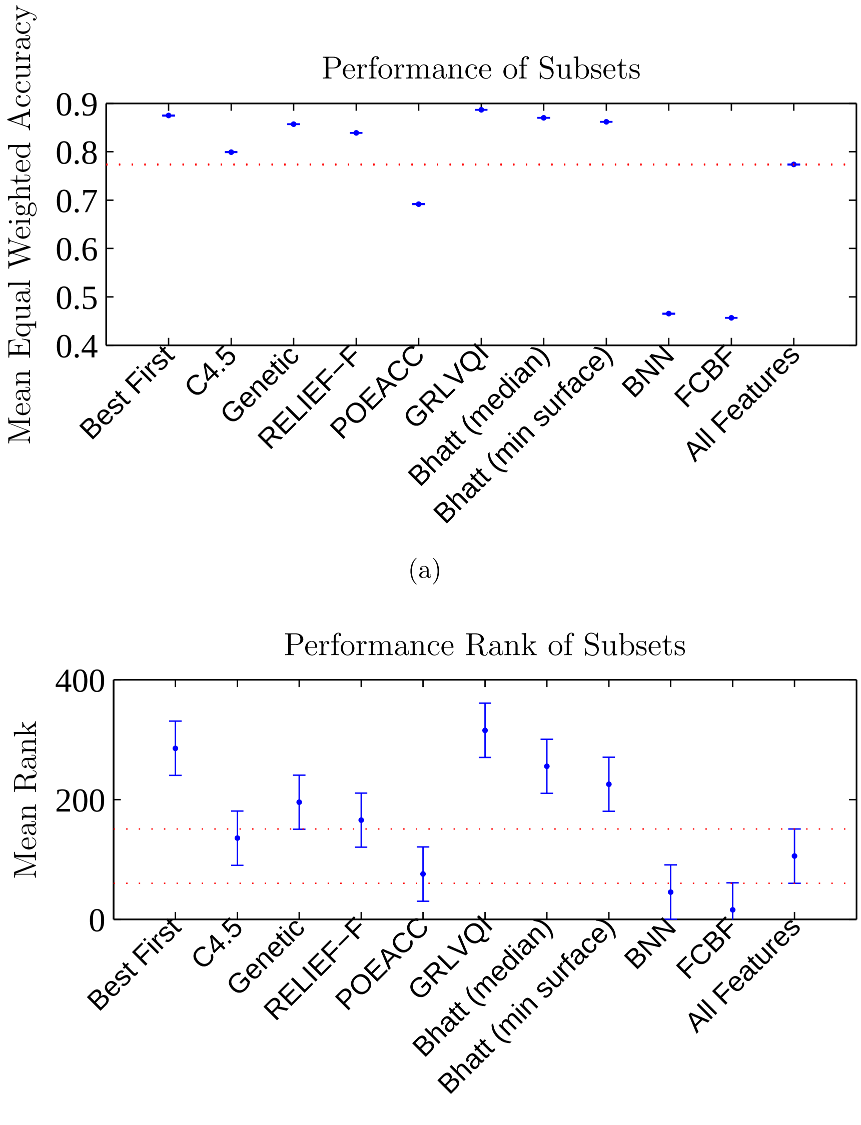


(a)

(b)

(c)

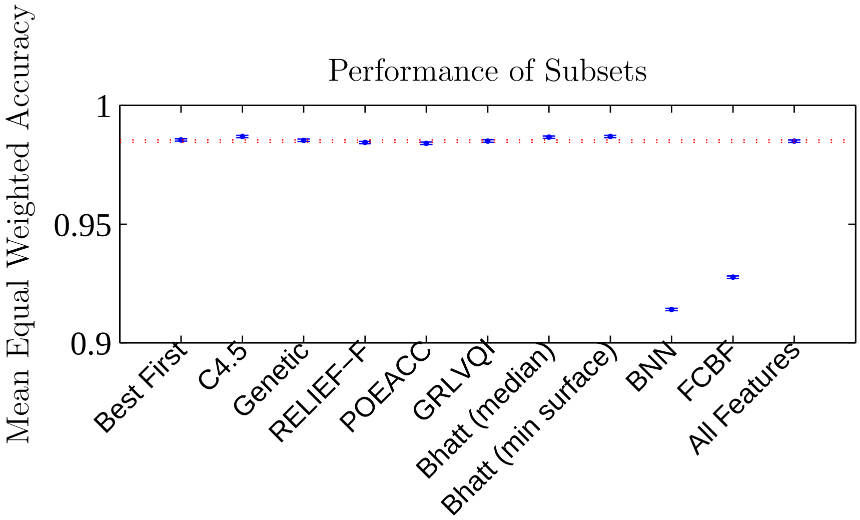
Figure A.8: Feature Selection: Minimum Surface Bhattacharyya. Analysis of the plots determines a single subset for Minimum Surface Bhattacharyya. Plot (a) illustrates where the performance plateaus. Plot (b) and (c) provide a multiple comparison of classification performance for determining statistically-significant differences for normally and non-normally distributed results, respectively



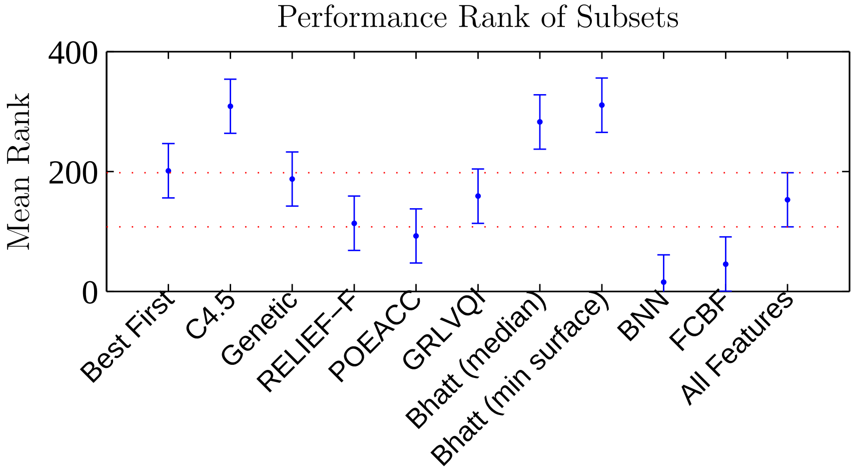
(a)

(b)

Figure A.9: Comparison of Selected Subsets: Näive Bayes Classifier. Compares Näive Bayes classifier performance of selected subsets with the entire feature set to determine, which subsets improve performance. Plot (a) and (b) provide a multiple comparison of classification performance for determining statistically-significant differences for normally and non-normally distributed results, respectively



(a)



(b)

Figure A.10: Comparison of Selected Subsets: C4.5 Classifier. Compares C4.5 classifier performance of selected subsets with the entire feature set to determine, which subsets improve performance. Plot (a) and (b) provide a multiple comparison of classification performance for determining statistically-significant differences for normally and non-normally distributed results, respectively

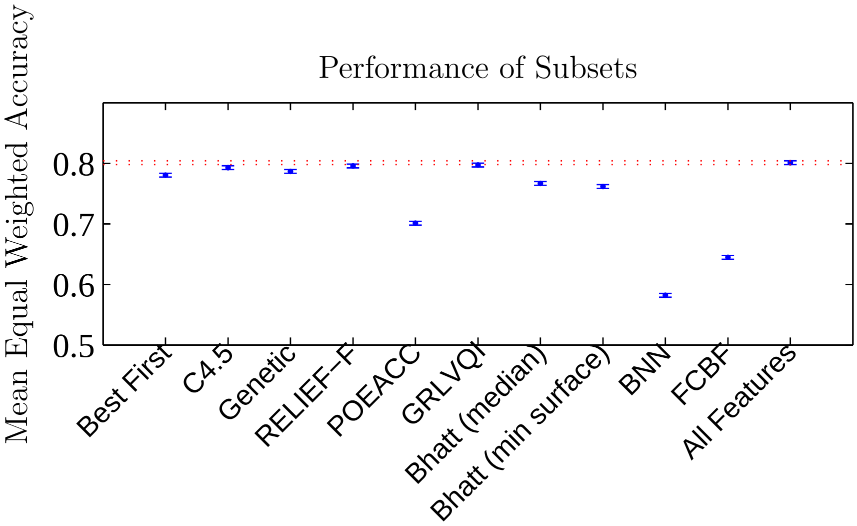
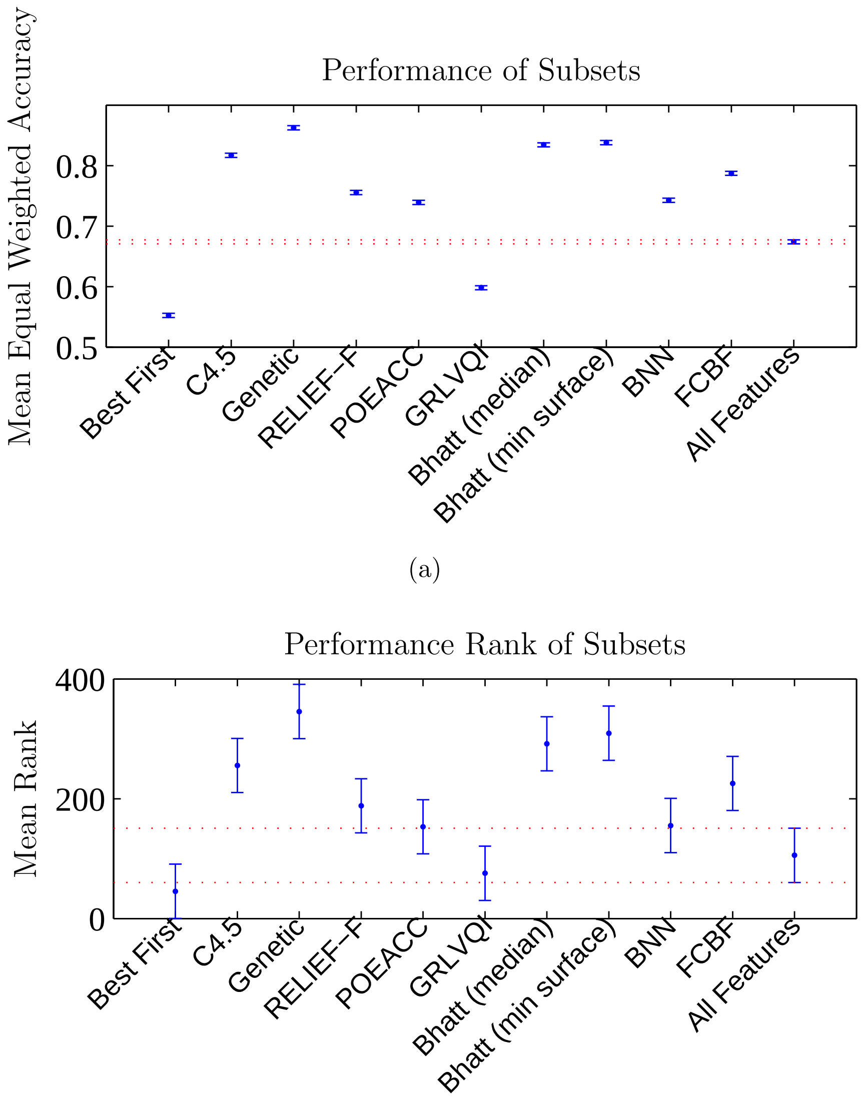


Figure A.11: Comparison of Selected Subsets: multilayer perceptron with backpropogation (BP) Classifier. Compares BP classifier performance of selected subsets with the entire feature set to determine, which subsets improve performance. Plot provides a multiple comparison of classification performance for determining statistically-significant differences for normally distributed results



(a)

(b)

Figure A.12: Comparison of Selected Subsets: Generalized Relevance Learning Vector Quantization Improved (GRLVQI) Classifier. Compares GRLVQI classifier performance of selected subsets with the entire feature set to determine, which subsets improve performance. Plot (a) and (b) provide a multiple comparison of classification performance for determining statistically-significant differences for normally and non-normally distributed results, respectively

Table A.1: Selected Subsets by Feature Selection Method

Feature Description	Index	BestFirst	C4.5	Genetic	RELIEF-F	POEACC	GRLVQI	Median Bhatt	Min Surf Bhatt	BNN [2]	FCBF [2, 60]	Index
Server Port	1	x	x	x	x	x	x	x	x			1
Client Port	2	x	x	x	x	x	x	x	x			2
min IAT	3					x						3
var IAT	9					x						9
q1 data wire	11		x			x				x		11
med data wire	12									x		12
mean data wire	13							x	x			13
max data wire	15	x	x		x		x					15
mean data ip	20							x	x			20
max data ip	22				x							22
min data control	24					x	x					24
med data control	26				x		x					26
max data control	29						x					29
var data control	30			x								30
ack pkts sent b a	34									x		34
pure acks sent a b	35					x						35
sack pkts sent a b	37					x				x		37
dsack pkts sent b a	40					x						40
unique bytes sent a b	43		x									43
actual data pkts a b	45										x	45
zwnd probe pkts a b	53					x						53
zwnd probe bytes b a	56					x						56
pushed data pkts a b	59									x	x	59
pushed data pkts b a	60									x	x	60
SYN pkts sent a b	61			x								61
req 1323 ts a b	66						x					66
req 1323 ws b a	67						x					67
req 1323 ts b a	68	x										68
req sack a b	71	x					x					71
req sack b a	72				x		x					72
mss requested a b	79		x									79
mss requested b a	80		x									80
max segm size a b	81				x		x	x				81
max segm size b a	82				x		x	x				82
min segm size a b	83		x	x	x		x	x	x		x	83
min segm size b a	84		x					x				84
avg segm size a b	85				x			x	x			85
avg segm size b a	86			x			x	x			x	86
max win adv a b	87		x							x		87
max win adv b a	88		x									88
min win adv b a	90		x					x	x			90
zero win adv a b	91		x	x								91
zero win adv b a	92	x										92
avg win adv a b	93							x	x			93
avg win adv b a	94							x	x			94
initial window-bytes a b	95	x				x	x	x	x		x	95
initial window-bytes b a	96						x	x	x		x	96
initial window-packets a b	97	x	x				x					97
truncated packets a b	105			x								105
data xmit time b a	108									x		108
idle time max a b	109						x			x		109
idle time max b a	110						x					110
throughput a b	111									x		111
throughput b a	112		x									112
RTT samples a b	113										x	113
RTT min a b	115		x									115
RTT max a b	117		x									117
RTT stdv a b	121		x									121
RTT from 3WHS b a	124		x									124
RTT full sz max a b	129			x								129
RTT full sz stdev a b	133									x		133
RTT full sz stdev b a	134									x		134
post-loss acks b a	136									x		136
segs cum acked a b	137		x									137
mean data wire a b	156							x	x			156
q3 data wire a b	157									x		157
max data wire a b	158				x		x	x		x		158
min data ip a b	160									x		160
med data ip a b	162										x	162
mean data ip a b	163		x					x	x			163
max data ip a b	165				x			x				165
q1 data control a b	168			x			x					168
med data control a b	169				x							169
mean data control a b	170				x							170
q3 data control a b	171				x		x					171
max data control a b	172						x					172
min data wire b a	174	x										174
mean data wire b a	177							x	x			177
q3 data wire b a	178						x					178
max data wire b a	179	x		x	x		x	x				179
var data wire b a	180										x	180
min data ip b a	181	x										181
med data ip b a	183						x					183
mean data ip b a	184							x	x			184
max data ip b a	186				x		x	x	x			186
q1 data control b a	189		x									189
var data control b a	194				x							194
q3 IAT b a	206									x		206
Time since last connection	209				x							209
No. transitions bulk/trans	210		x	x						x		210
Time spent idle	214									x		214
FFT Frequency#3 b a	241									x		241

Appendix B. List of Features from [61]

Number	**Short**	**Long**
1	Server Port	Port Number at server; we can establish server and client ports as we limit ourselves to flows for which we see the initial connection set-up.
2	Client Port	Port Number at client
3	min IAT	Minimum packet inter-arrival time for all packets of the flow (considering both directions).
4	q1 IAT	First quartile inter-arrival time
5	med IAT	Median inter-arrival time
6	mean IAT	Mean inter-arrival time
7	q3 IAT	Third quartile packet inter-arrival time
8	max IAT	Maximum packet inter-arrival time
9	var IAT	Variance in packet inter-arrival time
10	min data wire	Minimum of bytes in (Ethernet) packet, using the size of the packet on the wire.
11	q1 data wire	First quartile of bytes in (Ethernet) packet
12	med data wire	Median of bytes in (Ethernet) packet
13	mean data wire	Mean of bytes in (Ethernet) packet
14	q3 data wire	Third quartile of bytes in (Ethernet) packet
15	max data wire	Maximum of bytes in (Ethernet) packet
16	var data wire	Variance of bytes in (Ethernet) packet
17	min data ip	Minimum of total bytes in IP packet, using the size of payload declared by the IP packet
18	q1 data ip	First quartile of total bytes in IP packet
19	med data ip	Median of total bytes in IP packet
20	mean data ip	Mean of total bytes in IP packet
21	q3 data ip	Third quartile of total bytes in IP packet
22	max data ip	Maximum of total bytes in IP packet
23	var data ip	Variance of total bytes in IP packet
24	min data control	Minimum of control bytes in packet, size of the (IP/TCP) packet header
25	q1 data control	First quartile of control bytes in packet
26	med data control	Median of control bytes in packet
27	mean data control	Mean of control bytes in packet
28	q3 data control	Third quartile of control bytes in packet
29	max data control	Maximum of control bytes in packet
30	var data control	Variance of control bytes packet

Number	Short	Long
31	total packets a b	The total number of packets seen (client (a) to server (b)).
32	total packets b a	(server (b) to client (a))
33	ack pkts sent a b	The total number of ack packets seen (TCP segments seen with the ACK bit set) (client (a) to server (b)).
34	ack pkts sent b a	(server (b) to client (a))
35	pure acks sent a b	The total number of ack packets seen that were not piggy-backed with data (just the TCP header and no TCP data payload) and did not have any of the SYN/FIN/RST flags set (client (a) to server (b))
36	pure acks sent b a	(server (b) to client (a))
37	sack pkts sent a b	The total number of ack packets seen carrying TCP SACK [6] blocks (client (a) to server (b))
38	sack pkts sent b a	(server (b) to client (a))
39	dsack pkts sent a b	The total number of sack packets seen that carried duplicate SACK (D-SACK) [7] blocks. (client (a) to server (b))
40	dsack pkts sent b a	(server (b) to client (a))
41	max sack blks/ack a b	The maximum number of sack blocks seen in any sack packet. (client (a) to server (b))
42	max sack blks/ack b a	(server (b) to client (a))
43	unique bytes sent a b	The number of unique bytes sent, i.e., the total bytes of data sent excluding retransmitted bytes and any bytes sent doing window probing. (client (a) to server (b))
44	unique bytes sent b a	(server (b) to client (a))
45	actual data pkts a b	The count of all the packets with at least a byte of TCP data payload. (client (a) to server (b))
46	actual data pkts b a	(server (b) to client (a))
47	actual data bytes a b	The total bytes of data seen. Note that this includes bytes from retransmissions / window probe packets if any. (client (a) to server (b))
48	actual data bytes b a	(server (b) to client (a))
49	rexmt data pkts a b	The count of all the packets found to be retransmissions. (client (a) to server (b))
50	rexmt data pkts b a	(server (b) to client (a))
51	rexmt data bytes a b	The total bytes of data found in the retransmitted packets. (client (a) to server (b))
52	rexmt data bytes b a	(server (b) to client (a))
53	zwnd probe pkts a b	The count of all the window probe packets seen. (Window probe packets are typically sent by a sender when the receiver last advertised a zero receive window, to see if the window has opened up now). (client (a) to server (b))

Number	Short	Long
54	zwnd probe pkts b a	(server (b) to client (a))
55	zwnd probe bytes a b	The total bytes of data sent in the window probe packets. (client (a) to server (b))
56	zwnd probe bytes b a	(server (b) to client (a))
57	outoforder pkts a b	The count of all the packets that were seen to arrive out of order. (client (a) to server (b))
58	outoforder pkts b a	(server (b) to client (a))
59	pushed data pkts a b	The count of all the packets seen with the PUSH bit set in the TCP header. (client (a) to server (b))
60	pushed data pkts b a	(server (b) to client (a))
61	SYN pkts sent a b	The count of all the packets seen with the SYN bits set in the TCP header respectively (client (a) to server (b))
62	FIN pkts sent a b	The count of all the packets seen with the FIN bits set in the TCP header respectively (client (a) to server (b))
63	SYN pkts sent b a	The count of all the packets seen with the SYN bits set in the TCP header respectively (server (b) to client (a))
64	FIN pkts sent b a	The count of all the packets seen with the FIN bits set in the TCP header respectively (server (b) to client (a))
65	req 1323 ws a b	If the endpoint requested Window Scaling/Time Stamp options as specified in RFC 1323[8] a Y is printed on the respective field. If the option was not requested, an N is printed. For example, an N/Y in this field means that the window-scaling option was not specified, while the Time-stamp option was specified in the SYN segment. (client (a) to server (b))
66	req 1323 ts a b	. . .
67	req 1323 ws b a	If the endpoint requested Window Scaling/Time Stamp options as specified in RFC 1323[8] a Y is printed on the respective field. If the option was not requested, an N is printed. For example, an N/Y in this field means that the window-scaling option was not specified, while the Time-stamp option was specified in the SYN segment. (client (a) to server (b))
68	req 1323 ts b a	. . .
69	adv wind scale a b	The window scaling factor used. Again, this field is valid only if the connection was captured fully to include the SYN packets. Since the connection would use window scaling if and only if both sides requested window scaling [8], this field is reset to 0 (even if a window scale was requested in the SYN packet for this direction), if the SYN packet in the reverse direction did not carry the window scale option. (client (a) to server (b))

Number	Short	Long
70	adv wind scale b a	(server (b) to client (a))
71	req sack a b	If the end-point sent a SACK permitted option in the SYN packet opening the connection, a Y is printed; otherwise N is printed. (client (a) to server (b))
72	req sack b a	(server (b) to client (a))
73	sacks sent a b	The total number of ACK packets seen carrying SACK information. (client (a) to server (b))
74	sacks sent b a	(server (b) to client (a))
75	urgent data pkts a b	The total number of packets with the URG bit turned on in the TCP header. (client (a) to server (b))
76	urgent data pkts b a	(server (b) to client (a))
77	urgent data bytes a b	The total bytes of urgent data sent. This field is calculated by summing the urgent pointer offset values found in packets having the URG bit set in the TCP header. (client (a) to server (b))
78	urgent data bytes b a	(server (b) to client (a))
79	mss requested a b	The Maximum Segment Size (MSS) requested as a TCP option in the SYN packet opening the connection. (client (a) to server (b))
80	mss requested b a	(server (b) to client (a))
81	max segm size a b	The maximum segment size observed during the lifetime of the connection. (client (a) to server (b))
82	max segm size b a	(server (b) to client (a))
83	min segm size a b	The minimum segment size observed during the lifetime of the connection. (client (a) to server (b))
84	min segm size b a	(server (b) to client (a))
85	avg segm size a b	The average segment size observed during the lifetime of the connection calculated as the value reported in the actual data bytes field divided by the actual data pkts reported. (client (a) to server (b))
86	avg segm size b a	(server (b) to client (a))
87	max win adv a b	The maximum window advertisement seen. If the connection is using window scaling (both sides negotiated window scaling during the opening of the connection), this is the maximum window-scaled advertisement seen in the connection. For a connection using window scaling, both the SYN segments opening the connection have to be captured in the dumpfile for this and the following window statistics to be accurate. (client (a) to server (b))

Number	Short	Long
88	max win adv b a	(server (b) to client (a))
89	min win adv a b	The minimum window advertisement seen. This is the minimum window-scaled advertisement seen if both sides negotiated window scaling. (client (a) to server (b))
90	min win adv b a	(server (b) to client (a))
91	zero win adv a b	The number of times a zero receive window was advertised. (client (a) to server (b))
92	zero win adv b a	(server (b) to client (a))
93	avg win adv a b	The average window advertisement seen, calculated as the sum of all window advertisements divided by the total number of packets seen. If the connection endpoints negotiated window scaling, this average is calculated as the sum of all window-scaled advertisements divided by the number of window-scaled packets seen. Note that in the window-scaled case, the window advertisements in the SYN packets are excluded since the SYN packets themselves cannot have their window advertisements scaled, as per RFC 1323 [8]. (client (a) to server (b))
94	avg win adv b a	(server (b) to client (a))
95	initial window-bytes a b	The total number of bytes sent in the initial window i.e., the number of bytes seen in the initial flight of data before receiving the first ack packet from the other endpoint. Note that the ack packet from the other endpoint is the first ack acknowledging some data (the ACKs part of the 3-way handshake do not count), and any retransmitted packets in this stage are excluded. (client (a) to server (b))
96	initial window-bytes b a	(server (b) to client (a))
97	initial window-packets a b	The total number of segments (packets) sent in the initial window as explained above. (client (a) to server (b))
98	initial window-packets b a	(server (b) to client (a))
99	ttl stream length a b	The Theoretical Stream Length. This is calculated as the difference between the sequence numbers of the SYN and FIN packets, giving the length of the data stream seen. Note that this calculation is aware of sequence space wrap-arounds, and is printed only if the connection was complete (both the SYN and FIN packets were seen). (client (a) to server (b))
100	ttl stream length b a	(server (b) to client (a))
101	missed data a b	The missed data, calculated as the difference between the ttl stream length and unique bytes sent. If the connection was not complete, this calculation is invalid and an NA (Not Available) is printed. (client (a) to server (b))

Number	Short	Long
102	missed data b a	(server (b) to client (a))
103	truncated data a b	The truncated data, calculated as the total bytes of data truncated during packet capture. For example, with tcpdump, the snaplen option can be set to 64 (with NAME? option) so that just the headers of the packet (assuming there are no options) are captured, truncating most of the packet data. In an Ethernet with maximum segment size of 1500 bytes, this would amount to truncated data of 1500 64 = 1436bytes for a packet. (client (a) to server (b))
104	truncated data b a	(server (b) to client (a))
105	truncated packets a b	The total number of packets truncated as explained above. (client (a) to server (b))
106	truncated packets b a	(server (b) to client (a))
107	data xmit time a b	Total data transmit time, calculated as the difference between the times of capture of the first and last packets carrying non-zero TCP data payload. (client (a) to server (b))
108	data xmit time b a	(server (b) to client (a))
109	idletime max a b	Maximum idle time, calculated as the maximum time between consecutive packets seen in the direction. (client (a) to server (b))
110	idletime max b a	(server (b) to client (a))
111	throughput a b	The average throughput calculated as the unique bytes sent divided by the elapsed time i.e., the value reported in the unique bytes sent field divided by the elapsed time (the time difference between the capture of the first and last packets in the direction). (client (a) to server (b))
112	throughput b a	(server (b) to client (a))
113	RTT samples a b	The total number of Round-Trip Time (RTT) samples found. tcptrace is pretty smart about choosing only valid RTT samples. An RTT sample is found only if an ack packet is received from the other endpoint for a previously transmitted packet such that the acknowledgment value is 1 greater than the last sequence number of the packet. Further, it is required that the packet being acknowledged was not retransmitted, and that no packets that came before it in the sequence space were retransmitted after the packet was transmitted. Note : The former condition invalidates RTT samples due to the retransmission ambiguity problem, and the latter condition invalidates RTT samples since it could be the case that the ack packet could be cumulatively acknowledging the retransmitted packet, and not necessarily acking the packet in question. (client (a) to server (b))

Number	Short	Long
114	RTT samples b a	(server (b) to client (a))
115	RTT min a b	The minimum RTT sample seen. (client (a) to server (b))
116	RTT min b a	(server (b) to client (a))
117	RTT max a b	The maximum RTT sample seen. (client (a) to server (b))
118	RTT max b a	(server (b) to client (a))
119	RTT avg a b	The average value of RTT found, calculated straightforwardly as the sum of all the RTT values found divided by the total number of RTT samples. (client (a) to server (b))
120	RTT avg b a	(server (b) to client (a))
121	RTT stdv a b	The standard deviation of the RTT samples. (client (a) to server (b))
122	RTT stdv b a	(server (b) to client (a))
123	RTT from 3WHS a b	The RTT value calculated from the TCP 3-Way Hand-Shake (connection opening) [9], assuming that the SYN packets of the connection were captured. (client (a) to server (b))
124	RTT from 3WHS b a	(server (b) to client (a))
125	RTT full sz smpls a b	The total number of full-size RTT samples, calculated from the RTT samples of full-size segments. Full-size segments are defined to be the segments of the largest size seen in the connection. (client (a) to server (b))
126	RTT full sz smpls b a	(server (b) to client (a))
127	RTT full sz min a b	The minimum full-size RTT sample. (client (a) to server (b))
128	RTT full sz min b a	(server (b) to client (a))
129	RTT full sz max a b	The maximum full-size RTT sample. (client (a) to server (b))
130	RTT full sz max b a	(server (b) to client (a))
131	RTT full sz avg a b	The average full-size RTT sample. (client (a) to server (b))
132	RTT full sz avg b a	(server (b) to client (a))
133	RTT full sz stdev a b	The standard deviation of full-size RTT samples. (client (a) to server (b))
134	RTT full sz stdev b a	(server (b) to client (a))
135	post-loss acks a b	The total number of ack packets received after losses were detected and a retransmission occurred. More precisely, a post-loss ack is found to occur when an ack packet acknowledges a packet sent (acknowledgment value in the ack pkt is 1 greater than the packets last sequence number), and at least one packet occurring before the packet acknowledged, was retransmitted later. In other words, the ack packet is received after we observed a (perceived) loss event and are recovering from it. (client (a) to server (b))

Number	Short	Long
136	post-loss acks b a	(server (b) to client (a))
137	segs cum acked a b	The count of the number of segments that were cumulatively acknowledged and not directly acknowledged. (client (a) to server (b))
138	segs cum acked b a	(server (b) to client (a))
139	duplicate acks a b	The total number of duplicate acknowledgments received. (client (a) to server (b))
140	duplicate acks b a	(server (b) to client (a))
141	triple dupacks a b	The total number of triple duplicate acknowledgments received (three duplicate acknowledgments acknowledging the same segment), a condition commonly used to trigger the fast-retransmit/fast-recovery phase of TCP. (client (a) to server (b))
142	triple dupacks b a	(server (b) to client (a))
143	max # retrans a b	The maximum number of retransmissions seen for any segment during the lifetime of the connection. (client (a) to server (b))
144	max # retrans b a	(server (b) to client (a))
145	min retr time a b	The minimum time seen between any two (re)transmissions of a segment amongst all the retransmissions seen. (client (a) to server (b))
146	min retr time b a	(server (b) to client (a))
147	max retr time a b	The maximum time seen between any two (re)transmissions of a segment. (client (a) to server (b))
148	max retr time b a	(server (b) to client (a))
149	avg retr time a b	The average time seen between any two (re)transmissions of a segment calculated from all the retransmissions. (client (a) to server (b))
150	avg retr time b a	(server (b) to client (a))
151	sdv retr time a b	The standard deviation of the retransmission-time samples obtained from all the retransmissions. (client (a) to server (b))
152	sdv retr time b a	(server (b) to client (a))
153	min data wire a b	Minimum number of bytes in (Ethernet) packet (client (a) to server (b))
154	q1 data wire a b	First quartile of bytes in (Ethernet) packet
155	med data wire a b	Median of bytes in (Ethernet) packet
156	mean data wire a b	Mean of bytes in (Ethernet) packet
157	q3 data wire a b	Third quartile of bytes in (Ethernet) packet
158	max data wire a b	Maximum of bytes in (Ethernet) packet
159	var data wire a b	Variance of bytes in (Ethernet) packet
160	min data ip a b	Minimum number of total bytes in IP packet
161	q1 data ip a b	First quartile of total bytes in IP packet
162	med data ip a b	Median of total bytes in IP packet
163	mean data ip a b	Mean of total bytes in IP packet

Number	Short	Long
164	q3 data ip a b	Third quartile of total bytes in IP packet
165	max data ip a b	Maximum of total bytes in IP packet
166	var data ip a b	Variance of total bytes in IP packet
167	min data control a b	Minimum of control bytes in packet
168	q1 data control a b	First quartile of control bytes in packet
169	med data control a b	Median of control bytes in packet
170	mean data control a b	Mean of control bytes in packet
171	q3 data control a b	Third quartile of control bytes in packet
172	max data control a b	Maximum of control bytes in packet
173	var data control a b	Variance of control bytes packet
174	min data wire b a	Minimum number of bytes in (Ethernet) packet (server (b) to client (a))
175	q1 data wire b a	First quartile of bytes in (Ethernet) packet
176	med data wire b a	Median of bytes in (Ethernet) packet
177	mean data wire b a	Mean of bytes in (Ethernet) packet
178	q3 data wire b a	Third quartile of bytes in (Ethernet) packet
179	max data wire b a	Maximum of bytes in (Ethernet) packet
180	var data wire b a	Variance of bytes in (Ethernet) packet
181	min data ip b a	Minimum number of total bytes in IP packet
182	q1 data ip b a	First quartile of total bytes in IP packet
183	med data ip b a	Median of total bytes in IP packet
184	mean data ip b a	Mean of total bytes in IP packet
185	q3 data ip b a	Third quartile of total bytes in IP packet
186	max data ip b a	Maximum of total bytes in IP packet
187	var data ip b a	Variance of total bytes in IP packet
188	min data control b a	Minimum of control bytes in packet
189	q1 data control b a	First quartile of control bytes in packet
190	med data control b a	Median of control bytes in packet
191	mean data control b a	Mean of control bytes in packet
192	q3 data control b a	Third quartile of control bytes in packet
193	max data control b a	Maximum of control bytes in packet
194	var data control b a	Variance of control bytes packet
195	min IAT a b	Minimum of packet inter-arrival time (client (a) to server (b))
196	q1 IAT a b	First quartile of packet inter-arrival time
197	med IAT a b	Median of packet inter-arrival time
198	mean IAT a b	Mean of packet inter-arrival time

Number	Short	Long
199	q3 IAT a b	Third quartile of packet inter-arrival time
200	max IAT a b	Maximum of packet inter-arrival time
201	var IAT a b	Variance of packet inter-arrival time
202	min IAT b a	Minimum of packet inter-arrival time (server (b) to client (a))
203	q1 IAT b a	First quartile of packet inter-arrival time
204	med IAT b a	Median of packet inter-arrival time
205	mean IAT b a	Mean of packet inter-arrival time
206	q3 IAT b a	Third quartile of packet inter-arrival time
207	max IAT b a	Maximum of packet inter-arrival time
208	var IAT b a	Variance of packet inter-arrival time
209	Time since last connection	Time since the last connection between these hosts
210	No. transitions bulk/-trans	The number of transitions between transaction mode and bulk transfer mode, where bulk transfer mode is defined as the time when there are more than three successive packets in the same direction without any packets carrying data in the other direction
211	Time spent in bulk	Amount of time spent in bulk transfer mode
212	Duration	Connection duration
213	% bulk	Percent of time spent in bulk transfer
214	Time spent idle	The time spent idle (where idle time is the accumulation of all periods of 2 seconds or greater when no packet was seen in either direction)
215	% idle	Percent of time spent idle
216	Effective Bandwidth	Effective Bandwidth based upon entropy [10] (both directions)
217	Effective Bandwidth a b	(client (a) to server (b))
218	Effective Bandwidth b a	(server (b) to client (a))
219	FFT all	FFT of packet IAT (arctan of the top-ten frequencies ranked by the magnitude of their contribution) (all traffic) (Frequency #1)
220	FFT all	(Frequency #2)
221	FFT all	. . .
222	FFT all	. . .
223	FFT all	. . .
224	FFT all	. . .
225	FFT all	. . .
226	FFT all	. . .
227	FFT all	. . .
228	FFT all	(Frequency #10)
229	FFT a b	FFT of packet IAT (arctan of the top-ten frequencies ranked by the magnitude of their contribution) (client (a) to server (b)) (Frequency #1)

Number	**Short**	**Long**
230	FFT a b	(Frequency #2)
231	FFT a b	. . .
232	FFT a b	. . .
233	FFT a b	. . .
234	FFT a b	. . .
235	FFT a b	. . .
236	FFT a b	. . .
237	FFT a b	. . .
238	FFT b a	(Frequency #10)
239	FFT b a	FFT of packet IAT (arctan of the top-ten frequencies ranked by the magnitude of their contribution) (server (b) to client (a)) (Frequency #1)
240	FFT b a	(Frequency #2)
241	FFT b a	. . .
242	FFT b a	. . .
243	FFT b a	. . .
244	FFT b a	. . .
245	FFT b a	. . .
246	FFT b a	. . .
247	FFT b a	. . .
248	FFT b a	(Frequency # 10)
249	Classes	Application class, as assigned in [1]

Bibliography

1. Agosta, John M., Carlos Diuk-Wasser, Jaideep Chandrashekar, and Carl Livadas. "An Adaptive Anomaly Detector for Worm Detection". *Proceedings of Second Workshop on Tackling Computer Systems Problems with Machine Learning Techniques (SysML07).* 2007.

2. Auld, Tom, Andrew W. Moore, and Stephen F. Gull. "Bayesian Neural Networks for Internet Traffic Classification". *IEEE Transactions on Neural Networks*, 18(1):223–239, 2007.

3. Axelson, Stefan. *Intrusion Detection Systems: A Survey and Taxonomy.* Technical Report 99-15, Chalmers University of Technology Department of Computer Engineering, Göteborg, Sweden, March 2000.

4. Benediktsson, Jon A., Philip H. Swain, and Okan K. Ersoy. "Neural Network Approaches Versus Statistical Methods In Classification Of Multisource Remote Sensing Data". *Geoscience and Remote Sensing, IEEE Transactions on*, 28(4):540–552, Jul 1990. ISSN 0196-2892.

5. Bernaille, Laurent, Renata Teixeira, and Kavé Salamatian. "Early application identification". Christophe Diot and Mostafa H. Ammar (editors), *Proceedings of the ACM Conference on Emerging Network Experiment and Technology (NEXT)*, 6. ACM, 2006. ISBN 1-59593-456-1.

6. Bishop, Christopher M. *Pattern Recignition and Machine Learning.* Springer, 2006.

7. Blum, Avrim L. and Pat Langley. "Selection of relevant features and examples in machine learning". *Artificial Intelligence*, 97(1-2):245–271, 1997.

8. Boz, Olcay. "Feature Subset Selection by Using Sorted Feature Relevance". M. Arif Wani, Hamid R. Arabnia, Krzysztof J. Cios, Khalid Hafeez, and Graham Kendall (editors), *Proceedings of the International Conference on Machine Learning and Applications (ICMLA)*, 147–153. CSREA Press, 2002. ISBN 1-892512-29-7.

9. Buja, Andreas and Yung-Seop Lee. "Data Mining Criteria for Tree-Based Regression and Classification". *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 27–36. ACM, San Francisco, CA, 2001.

10. University of California, Irvine (UCI) Knowledge Discovery in Data (KDD) Archive. "KDD Cup 1999 Data". [Online], 1999. URL `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`.

11. Cannady, James. "Artificial Neural Networks for Misuse Detection". *Proceedings of the National Information Systens Security Conference (NISSC'98)*. Arlington, VA, 1998.

12. Cardie, Claire. "Using Decision Trees to Improve Case-Based Learning". *Proceedings of the Tenth International Conference on Machine Learning*, 25–32. Morgan Kaufmann, 1993.

13. Chebrolu, Srilatha, Ajith Abraham, and Johnson P. Thomas. "Feature deduction and ensemble design of intrusion detection systems". *Computers & Security*, 24(4):295–307, 2005.

14. Chen, Yuehui, Ajith Abraham, and Ju Yang. "Feature Selection and Intrusion Detection Using Hybrid Flexible Neural Tree". Jun Wang, Xiaofeng Liao, and Zhang Yi (editors), *Proceedings of the Second International Symposium on Neural Networks (ISNN)*, volume 3498 of *Lecture Notes in Computer Science*, 439–444. Springer, 2005. ISBN 3-540-25914-7.

15. Claffy, Kimberly C., Hans-Werner Braun, and George C. Polyzos. "A Parameterizable Methodology for Internet Traffic Flow Profiling". *IEEE Journal on Selected Areas in Communications*, 13(8):1481–1494, 1995.

16. D'Agostino, Ralph B. and Michael A. Stephens (editors). *Goodness-of-Fit Techniques*, volume 68 of *STATISTICS: Textbooks and Monographs*. Marcel Dekker, New York, NY, 1986.

17. Dash, Manoranjan and Huan Liu. "Feature Selection for Classification". *Intelligent Data Analysis 1*, volume 2, 131–156. 1997.

18. Debar, Hervé, Monique Becker, and Didier Siboni. "A Neural Network Component for an Intrusion Detection System". *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, 240–250. Oakland, CA, May 1992.

19. DeSieno, Duane. "Adding a Conscience to Competitive Learning". *Proceedings of the IEEE International Conference on Neural Networks*, volume 1, 117–124. IEEE, New York, 1988.

20. Erman, Jeffrey, Martin F. Arlitt, and Anirban Mahanti. "Traffic classification using clustering algorithms". *Proceedings of the ACM Special Interest Group on Data Communications (SIGCOMM) Conference on Mining Network Data Workshop (MineNet)*, 281–286. ACM, 2006. ISBN 1-59593-569-X.

21. Gharibian, Farnaz and Ali A. Ghorbani. "Comparative Study of Supervised Machine Learning Techniques for Intrusion Detection". *Communication Networks and Services Research, 2007. CNSR '07. Fifth Annual Conference on*, 350–358, May 2007.

22. Ghosh, Anup K., Aaron Schwartzbard, and Michael Schatz. "Learning Program Behavior Profiles for Intrusion Detection". *Proceedings of the Workshop on Intru-*

sion Detection and Network Monitoring, 51–62. USENIX Association, Berkeley, CA, April 9-12 1999.

23. Giacinto, Giorgio and Fabio Roli. “Intrusion detection in computer networks by multiple classifier systems”. *Proceedings of 16th International Conference on Pattern Recognition*, volume 2, 390–393 vol.2. 2002. ISSN 1051-4651.

24. Gibbons, Jean Dickinson and Subhabrata Chakraborti. *Nonparametric Statistical Inference*, volume 168 of *STATISTICS: Textbooks and Monographs.* Marcel Dekker, New York, NY, 2003.

25. Golovko, Vladimir and Pavel Kochurko. “Intrusion Recognition Using Neural Networks”. *Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, 108–111, Sept. 2005.

26. Guyon, Isabelle and André Elisseeff. “An Introduction to Variable and Feature Selection”. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

27. Haines, Joshua W., Richard P. Lippmann, David J. Fried, Eushiuan Tran, Steve Boswell, and Marc A. Zissman. *1999 DARPA Intrusion Detection System Evaluation: Design and Procedures.* Technical Report TR-1062, Massachusetts Institute of Technology Lincoln Laboratory, August 2001.

28. Hall, Mark A. “Correlation-based feature selection for discrete and numeric class machine learning”. *Proceedings of the Seventeenth International Conference on Machine Learning*, 359–366. 2000.

29. Hammer, Barbara, Marc Strickert, and Thomas Villmann. “Learning vector quantization for multimodal data”. *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, 370–376. 2002.

30. Hammer, Barbara and Thomas Villmann. “Generalized relevance learning vector quantization”. *Neural Networks*, 15(8-9):1059–1068, 2002.

31. Haykin, Simon. *Neural Networks: A Comprehensive Foundation.* IEEE Press, Macmillan College, New York, 1994.

32. Holmstrom, Lasse, Petri Koistinen, Jorma Laaksonen, and Erkki Oja. “Neural and Statistical Classifiers: Taxonomy and Two Case-Studies”. *IEEE Trans. Neural Networks*, 8(1):5–17, January 1997.

33. John, George H. and Pat Langley. “Estimating continuous distributions in Bayesian classifiers”. *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, 338–345. Morgan Kaufmann, 1995.

34. Karagiannis, Thomas, Andre Broido, Nevil Brownlee, Kimberly Claffy, and Michalis Faloutsos. *File-sharing in the Internet: A characterization of P2P traffic in the backbone.* Technical Report Technical Report, Queen Mary, University of London, 2003.

35. Kira, Kenji and Larry A. Rendell. "The Feature Selection Problem: Traditional Methods and a New Algorithm". *Proceedings of the Tenth National Conference on Artificial Intelligence AAAI*, 129–134. 1992.

36. Kira, Kenji and Larry A. Rendell. "A Practical Approach to Feature Selection". *Assorted Conferences and Workshops*, 249–256, 1992.

37. Kohonen, Teuvo. *Self-Organizing Maps*. Springer, 2001.

38. Kononenko, Igor. "Estimating Attributes: Analysis and Extensions of Relief". *Proceedings of the European Conference on Machine Learning*, 171–182. 1994.

39. Kou, Yufeng, Chang-Tien Lu, Sirirat Sirwongwattana, and Yo-Ping Huang. "Survey of Fraud Detection Techniques". *Proceedings of the International Conference on Networking, Sensing, and Control*, 749–754. 2004.

40. Kumar, Vipin, Jaideep Srivastava, and Aleksandar Lazarevic. *Managing Cyber Threats*. Springer, 2005.

41. Lakhina, Anukool, Mark Crovella, and Christophe Diot. "Characterization of network-wide anomalies in traffic flows". Alfio Lombardo and James F. Kurose (editors), *Proceedings of the 4th ACM Special Interest Group on Data Communications (SIGCOMM) Conference on Internet Measurement*, 201–206. ACM, 2004. ISBN 1-58113-821-0.

42. Laskov, Pavel, Patrick Dussel, Christin Schafer, , and Konrad Rieck. "Learning intrusion detection: Supervised or unsupervised?" *Lecture Notes in Computer Science*, 3617:50–57, 2005.

43. Lee, Joong-Hee, Jong-Hyouk Lee, Seon-Gyoung Sohn, Jong-Ho Ryu, and Tai-Myoung Chung. "Effective Value of Decision Tree with KDD 99 Intrusion Detection Datasets for Intrusion Detection System". *Advanced Communication Technology, 2008. ICACT 2008. 10th International Conference on*, 2:1170–1175, Feb. 2008. ISSN 1738-9445.

44. Li, Jian, Guo-Yin Zhang, and Guo-Chang Gu. "The research and implementation of intelligent intrusion detection system based on artificial neural network". *Proceedings of the International Conference on Machine Learning and Cybernetics*, 3178–3182 vol.5. 2004.

45. Lippmann, Richard P., David J. Fried, Isaac Graf, Joshua W. Haines, Kristopher R. Kendall, David McClung, Dan Weber, Seth E. Webster, Dan Wyschogrod, Robert K. Cunningham, and Marc A. Zissman. "Evaluating Intrusion Detection Systems: the 1998 DARPA Off-Line Intrusion Detection Evaluation". *Proceedings of the 2000 DARPA Information Survivability Conference and Exposition*, volume 2. 2000.

46. Liu, Huan and Hiroshi Motoda. *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic Publishers, 1998.

47. Liu, Huan and Lei Yu. "Toward Integrating Feature Selection Algorithms for Classification and Clustering". *IEEE Trans. Knowl. Data Eng*, 17(4):491–502, 2005.

48. Lunt, Teresa. "IDES: An intelligent System for Detecting Intruders". *Proceedings of the Symposium: Computer Security, Threat and Countermeasures.* November 1990.

49. Bonifacio, Jr., Jose M., Adriano M. Cansian, Andre C. P. L. F. De Carvalho, and Edson dos S. Moreira. "Neural networks applied in intrusion detection systems". volume 1, 205–210 vol.1. May 1998.

50. Dr. Seuss. *One Fish, Two Fish, Red Fish, Blue Fish.* Random House, Inc., 1960.

51. Internet Assigned Numbers Authority (IANA). "Port Numbers". [Online], 2009. URL `http://www.iana.org/assignments/port-numbers`.

52. United States Department of Defense. "The Implementation of Network-Centric Warfare", January 2005.

53. United States Government. "The National Strategy to Secure Cyberspace", February 2003.

54. McHugh, John. "Testing Intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory". *ACM Trans. Inf. Syst. Secur*, 3(4):262–294, 2000.

55. Mendenhall, Michael J. "personal communication (verbal)", December 2008. AFIT/ENG.

56. Mendenhall, Michael J. and Erzsbet Mernyi. "Relevance-Based Feature Extraction for Hyperspectral Images". *IEEE Transactions on Neural Networks*, 19, April 2008.

57. Milton, J. Susan and Jesse C. Arnold. *Introduction to Probability and Statistics.* McGraw Hill, 2003.

58. Moore, Andrew W., James Hall, Euan Harris, Christian Kreibech, and Ian Pratt. "Architecture of a Network Monitor". *Proceedings of the Fourth Passive and Active Measurement Workshop (PAM 2003).* April 2003.

59. Moore, Andrew W. and Konstantina Papagiannaki. "Toward the Accurate Identification of Network Applications". Constantinos Dovrolis (editor), *Proceedings of the Sixth Passive and Active Network Measurement Workshop, (PAM 2005)*, volume 3431 of *Lecture Notes in Computer Science*, 41–54. Springer, 2005. ISBN 3-540-25520-6.

60. Moore, Andrew W. and Denis Zuev. "Internet traffic classification using bayesian analysis techniques". Derek L. Eager, Carey L. Williamson, Sem C. Borst, and John C. S. Lui (editors), *Proceedings of the International Conference on Measurements and Modeling of Computer Systems, SIGMETRICS*, 50–60. ACM, 2005. ISBN 1-59593-022-1.

61. Moore, Andrew W., Denis Zuev, and Michael Crogan. *Discriminators for Use in Flow-based Classification.* Technical Report RR-05-13, Queen Mary, University of London, August 2005.

62. Moradi, Mehdi and Mohammad Zulkernine. "A Neural Network Based System for Intrusion Detection and Classification of Attacks". *Proceedings of the IEEE International Conference on Advances in Intelligent Systems.* 2004.

63. Mucciardi, Anthony N. and Earl E. Gose. "A Comparison of Seven Techniques for Choosing Subsets of Pattern Recognition Properties". *IEEE Transactions on Computers*, c-20(9):1023–1031, 1971.

64. Mukkamala, Srinivas, Guadalupe Janoski, and Andrew Sung. "Intrusion Detection using Neural Networks and Support Vector Machines". *Proceedings of the 2002 International Joint Conference on Neural Networks (IJCNN).* Honolulu, HI, 2002.

65. Nazario, Jose. "DDoS attack evolution". *Network Security*, 7:7–10, July 2008.

66. Nguyen, Thuy T. T. and Grenville Armitage. "Training on multiple sub-flows to optimise the use of machine learning classifiers in real-world ip networks". *Proceedings of the 31st IEEE Conference on Local Computer Networks.* November 2006.

67. Paxson, Vern. *Empirically-Derived Analytic Models of Wide-Area TCP Connections: Extended Report.* Technical Report TR LBL-34986, Lawrence Berkeley Labs, June 15, 1993.

68. Quinlan, J. Ross. "Induction of Decision Trees". *Machine Learning*, 1:81–106, 1986.

69. Quinlan, J. Ross. "Simplifying Decision Trees,", December 1986.

70. Quinlan, J. Ross. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

71. Ramsey, Fred L. and Daniel W. Schafer. *The Statistical Sleuth: A Course in Methods of Data Analysis.* Duxbury, 2002.

72. Ren, Pin, Yan Gao, Zhichun Li, Yan Chen, and Benjamin Watson. "IDGraphs: Intrusion Detection and Analysis Using Histographs". Kwan-Liu Ma, Stephen C. North, and William Yurcik (editors), *Proceedings of the IEEE Workshop on Visualization for Computer Security (VizSEC)*, 5. IEEE Computer Society, 2005. ISBN 0-7803-9477-1.

73. Rosenblatt, Frank. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms.* Spartan Books, Washington DC, 1961.

74. Rossey, Lee M., Robert K. Cunningham, David J. Fried, Jesse C. Rabek, Richard P. Lippmann, Joshua W. Haines, and Marc A. Zissman. "LARIAT:

Lincoln Adaptable Real-Time Information Assurance Testbed". *IEEE Aerospace Conference*. Big Sky, Montana, USA, March 9-16 2002.

75. Rumelhart, David E., Geoffrey E. Hinton, and R. J. Williams. "Learning Internal Representations by Error Propagation". David E. Rumelhart, James L. McClelland, and the PDP research group. (editors), *Parallel distributed processing: Explorations in the microstructure of cognition, Volume 1: Foundations*. MIT Press, 1986.

76. Russel, Stuart and Peter Novig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.

77. Sabhnani, Maheshkumar and Gürsel Serpen. "Why machine learning algorithms fail in misuse detection on KDD intrusion detection data set". *Intelligent Data Analysis*, 8(4):403–415, 2004.

78. Snedecor, George W. and William G. Cochran. *Statistical Methods*. Blackwell Publishing, eighth edition edition, 1991.

79. Somayaji, Anil. "How to Win and Evolutionary Arms Race". *IEEE Security & Privacy*, 2(6):70–72, 2004.

80. Sung, Andrew H. and Srinivas Mukkamala. "Identifying Important Features for Intrusion Detection Using Support Vector Machines and Neural Networks". *Proceedings of the Symposium on Applications and the Internet (SAINT)*, 209–217. IEEE Computer Society, 2003. ISBN 0-7695-1872-9.

81. Tan, Kymie M. C. and Blair R. Collie. "Detection and Classification of TCP/IP Network Services". *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, 99–107. IEEE Computer Society, 1997. ISBN 0-8186-8274-4.

82. Thacker, Neil A., Frank J. Aherne, and Peter I. Rockett. "The Bhattacharyya Metric as an Absolute Similarity Measure for Frequency Coded Data". *Kybernetika*, 34(4):363–368, 1998.

83. Thode, Henry C. *Testing for Normality*. CRC Press, 2002.

84. Utschick, Wolfgang, P. Nachbar, C. Knobloch, A. Schuler, and Josef A. Nossek. "The evaluation of feature extraction criteria applied to neural network classifiers". *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, 315–318. 1995.

85. Williams, Nigel, Sebastian Zander, and Grenville Armitage. *Evaluating Machine Learning Algorithms for Automated Network Application Identification*. Technical Report Technical Report 060410B, Centre for Advanced Internet Architectures (CAIA), April 2006.

86. Wilson, Edwin B. "Probable inference, the law of succession, and statistical inference". *Journal of the American Statistical Association*, 22:209–212, 1927.

87. Witten, Ian H. and Eibe Frank. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, San Francisco, 2005.

88. Yegneswaran, Vinod, Paul Barford, and Johannes Ullrich. "Internet intrusions: global characteristics and prevalence". *Proceedings of the ACM Special Interest Group on Measurement and Evaluation (SIGMETRICS)*, 138–147. ACM, 2003.

89. Zainal, Anazida, Mohd A. Maarof, and Siti M. Shamsuddin. "Feature Selection Using Rough Set in Intrusion Detection". *Proceedings of the IEEE Region 10 Conference (TENCON)*, 1–4. 2006. ISBN 1-4244-0548-3.

90. Zander, Sebastian, Thuy Nguyen, and Grenville Armitage. "Automated Traffic Classification and Application Identification using Machine Learning". *Proceedings of the 30th IEEE Conference on Local Computer Networks (LCN)*, 250–257. IEEE Computer Society, 2005. ISBN 0-7695-2421-4.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)* 26–03–2009	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From — To)* Sept 2007 — Mar 2009

4. TITLE AND SUBTITLE

Numerical Analysis for Relevant Features
in Intrusion Detection
(NARFid)

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

José Andrés González, Capt, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GCE/ENG/09-02

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Nicholas Fraser
Air Force Information Operations Center (AFIOC)
Information Operations Technology Division
102 Hall Blvd, Suite 345, San Antonio, TX 78243-7038
210-977-6445; nicholas.fraser@lackland.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

AFIOC

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approval for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This thesis evaluates the usefulness of good feature subsets for the general classification task of identifying cyber attacks and network services. The generality of the selected features elucidates the relevance or irrelevance for the classification task of intrusion detection. Additionally, the work provides an extension to assessing features by inter-class separability (Bhattacharyya Coefficient) for multiple class problems, which intends to select the best-performing features for all of the classes.

15. SUBJECT TERMS

Artificial intelligence; machine learning; feature selection; intrusion detection; classification; Bhattacharyya.

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Maj Michael J. Mendenhall
U	U	U	UU	126	**19b. TELEPHONE NUMBER** *(include area code)* (937) 255–3636, ext 4614 Michael.Mendenhall@afit.edu