DETECTOR DESIGN CONSIDERATIONS IN HIGH-DIMENSIONAL ARTIFICIAL IMMUNE SYSTEMS

THESIS

Jason M. Bindewald , Captain, USAF

AFIT/GCO/ENG/12-02

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCO/ENG/12-02

DETECTOR DESIGN CONSIDERATIONS IN HIGH-DIMENSIONAL ARTIFICIAL IMMUNE SYSTEMS

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Insitute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science

Jason M. Bindewald , M.S.I.T.M., B.S.C.S

Captain, USAF

March 2012

AFIT/GCO/ENG/12-02

DETECTOR DESIGN CONSIDERATIONS IN HIGH-DIMENSIONAL ARTIFICIAL IMMUNE SYSTEMS

Jason M. Bindewald , M.S.I.T.M., B.S.C.S
Captain, USAF

Approved:

Angela A. Sodemann, PhD (Co-Chairman)	1 Mar 2012 Date
FOR Lt Col Brett J. Borghetti, PhD (Co-Chairman)	1 MAR 2012 Date
Gilbert L. Peterson, PhD (Committee Member)	28 FEB 2012 Date
Gary B. Lamont, PhD (Committee Member)	1 MAR 2012 Date
Lt Col Joshua J. Corner (Committee Member)	1 MAR 2012 Date

Abstract

Network intrusion detection systems (NIDS) provide an area of expanding research for cyber defense interests. This research aims to lay the groundwork for a system that can operate with only knowledge of normal network traffic, using a process known as anomaly detection. One method for detecting anomalous data is that of Artificial Immune Systems (AIS). Real-valued negative selection (RNS) is a specific AIS algorithm that can be used to perform two-class classification when only one class is available for training. Researchers have shown fundamental problems with the geometry of the most common detector shape, hyperspheres, in high-dimensional space. Additionally, the research contained herein shows that the second most common detector type, hypercubes, can cause problems due to biasing certain features over others in high-dimensional space. To address these problems, a new detector shape known as the hypersteinmetz solid has been proposed, the goal of which is to provide a tradeoff between the geometrical problems of hyperspheres and hypercubes in high-dimensional spaces. In order to investigate the potential benefits of the hypersteinmetz solid, an effective RNS detector size range is determined. Then the relationship between content coverage of the dataset and classification accuracy is investigated. Once these issues are addressed, this research shows the tradeoffs that take place in high-dimensional data when hypersteinmetzes are chosen over hyperspheres or hypercubes. The final results of experiments show that detector shape is the dominant factor in high-dimensional detection, contributing 86% of variance in the classification accuracy results in 11 dimensions of the chosen dataset as compared to 0% in 2 dimensions. This verifies that detector shape becomes an increasingly important factor in classification accuracy within a real-valued negative selection system as dimensionality increases.

To my wife and children. I love you and thank you.

Acknowledgments

I would like to express my sincere appreciation to my research advisors, Dr. Angela Sodemann and Lt Col Brett Borghetti, for their insightful support, teaching, and mentorship throughout the course of this research. Their experience and knowledge helped so much that I cannot thank them enough.

I would, also, like to thank my thesis committee members, Dr. Gary Lamont, Dr. Gilbert Peterson, and Lt Col Josh Corner, for their invaluable insight into these research efforts.

Jason M. Bindewald

Table of Contents

List of Figures

List of Tables

List of Symbols

List of Abbreviations

Detector Design Considerations in High-Dimensional Artificial Immune Systems

1 Introduction

The United States is currently operating in a networked world. Enemies and allies alike are increasingly dependent upon cyberspace and, as such, the lines of information warfare are being drawn. New threats to the U.S.'s information resources are emerging almost daily. The current research, discussed herein, aims to address these problems by creating the framework for a network intrusion detection system operating that addresses the problems presented to artificial immune systems in high-dimensionality by using the hypersteinmetz solid.

This introductory chapter aims to lay the foundation for the research that proceeds it, in the following format. First, the vision and policy motivating for the current research are explored. Next, the problem that this research addresses is proposed. Then, a discussion of how we attempt to address this problem takes place. Finally, a discussion of the thesis of this research, and the results obtained follows.

1.1 Motivation

The importance of this research to the United States Air Force (USAF) can be shown through a discussion of the motivating policies. A few of the pertinent documents include: the National Military Strategy for Cyberspace Operations, United States Air Force Blueprint for Cyberspace, Comprehensive National Cybersecurity Initiative, Cyberspace Operations: Air Force Doctrine Document 3-12, and DoD Strategy for Operating in Cyberspace. The main points of these documents and how they work together to

implement the information management initiatives previously outlined are discussed in the following section.

The first major military policy document regarding the operation of DoD forces within the cyberspace domain was the National Military Strategy for Cyberspace Operations (NMSCO), published in December 2006. The purpose of the document was to provide a working framework of how the DoD operates in cyberspace with regard to military, intelligence, and business operations. The strategy outlines the contexts in which military operations would be needed to defend the global information grid (GIG). Although the document focuses largely on the emerging landscape of using cyber as a weapon, there are also specific focuses on the securing of cyberspace assets. Specifically, one of the strategic priorities outlined in NMSCO is to

> "Maintain continuous active layered defenses using existing information assurance guidance to protect confidentiality, integrity, availability, authentication, and non-repudiation of information as it is processed, created, and manipulated at rest and in-motion [50]."

This policy clearly reflects the move toward fusing the previously discussed IRM documents within an emerging threat environment.

The United States Air Force Blueprint for Cyberspace (USAFBC), released in November 2009, was an initial push made by the USAF to integrate the service's current cyberspace operations activities with a long-range cyberspace plan. One of the main purposes of the document was to provide a culture change to the Air Force's cyber personnel, specifically to "shift paradigms from network-focus to mission-focus [49]." The problem of aligning the USAF mission with NMSCO was that there needed to be less focus on protecting and ensuring the network "for the network's sake," and more focus on ensuring information resources in support of the broader mission. Within the document

several strategies for accomplishing this task are outlined, with 11 specific objectives identified [49].

Soon after the USAFBC was released, National Cybersecurity Coordinator Howard Schulz from the office of President Barack Obama released the Comprehensive National Cybersecurity Initiative (CNCI). In a similar vein with USAFBC, the initiative highlighted 12 areas of focus that the federal government would take to help secure its cyberspace assets. Of the twelve areas identified, two of them pertain to the current discussion: "Initiative 2. Deploy an intrusion detection system of sensors across the Federal enterprise" and "Initiative 9. Define and develop enduring 'leap-ahead' technology, strategies, and programs [47]." While there are specific programs outlined in the document that are beginning to achieve the goal of intrusion detection, Initiative 9 provides the groundwork for developing new technologies that can help to defend the federal government's information resources and the technologies and infrastructure upon which they rely [47].

Released in July 2010, Cyberspace Operations: Air Force Doctrine Document 3-12 was released by the USAF LeMay Center with a goal of codifying the principles laid out previously in USAFBC. The document's specific purpose is outlined as

> "[T]he Air Force's foundational doctrine publication for Air Force operations in, through, and from the cyberspace domain. [It] represents known sanctioned ideas and practices . . . to provide insight for Airmen to follow. This document speaks to Air Force support of maintaining Cybersapce Superiority." [48]

With the creation of this document, the USAF had, for the first time, created a set of operational ideas and practices to not only support and defend operations in cyberspace, but also to execute offensive operations. A principal problem acknowledged within the document, however, is that it was created prior to a joint operational cyberspace strategy.

Therefore, when a joint doctrine document would be established, it would be significantly harder to ensure that they were in alignment [48].

With this void of joint policy in mind, the DoD created the Department of Defense Strategy for Operating in Cyberspace (DODSOC) in July 2011. The document created five strategic initiatives that were aimed at shaping how the military "leverages the opportunities of cyberspace, while managing inherent uncertainties and reducing vulnerabilities [51]." The document provides a broad overview of how the DoD is to operate in cyberspace, and does not provide many details on specific doctrine decisions. However, it does establish cyberspace as a domain of military operations, decisively delineating a long-held discussion. It additionally continues to implore organizations to pursue new defense operating concepts within the context of cyberspace [51].

As the preceding documents have shown, the United States military, and the Air Force specifically, has-in recent years-taken a keen interest in defining how military operations are evolving in cyberspace as well as how the federal government's cyber assets can be defended. By taking this doctrine, it becomes clear that defense of information resources relies heavily on the safeguarding of the infrastructure upon which it resides. Innovation into the realm of network and information infrastructure is essential to the proper execution of cyberspace operations. The motivational theme of this research, therefore, is to place the groundwork for a network intrusion detection system in order to further facilitate protection of United States Air Force technology assets.

1.2 Problem statement

The first question to be answered is "what types of network intrusion detection systems can be created?" This research investigates anomaly-based network intrusion detection methods. The specific type of anomaly-based system chosen finds its roots in the artificial immune systems (AIS) line of research. The specific AIS model chosen is that of real-valued negative selection. Network intrusion datasets have many features, and

translate into high-dimensional spaces when translated into real-valued negative selection problems. However, there are difficulties with real-valued negative selection systems in high-dimensional data spaces. Two of these problems are: the content covered and biasing of features by detectors high-dimensional spaces. In order to create a real-valued negative selection-based NIDS, how do we overcome the content and feature bias problems in high-dimensional real-valued negative selection detectors?

1.3 How problem statement is addressed

In order to address the problems inherent in high-dimensional real-valued negative selection detectors, this research focuses on detector shape as the main component of a successful system design. The hypersteinmetz solid is presented as an alternative to the most common shapes currently found in the literature: hyperspheres and hypercubes. This analyzes how the hypersteinmetz performs in high-dimensions, specifically as it pertains to content coverage and feature bias, when compared against hyperspheres and hypercubes. The research then shifts toward designing the constraints needed for an experimental comparison of the three detector shapes. First, the detector radius size is determined through a set of experiments. Then, the relationship between the content coverage of a set of detectors, or coverage factor, and real-valued negative selection classification accuracy is investigated. These experiments are then brought together to form the basis for an experimental comparison of the effects of detector shape within a real-valued negative selection system as dimensionality increases.

1.3.1 Thesis statement. This research aims to show that "Detector shape is an extremely important factor in the effectiveness of a real-valued negative selection system as the number of dimensions of data increases, especially in comparison to other factors such as radius size and coverage factor."

1.4 Overview of results

The results of the analysis and experiments outlined above confirm the thesis. A successful detector radius bounding range is found. Then, it is determined that increasing coverage factor does improve classification accuracy, but at a rate of diminishing returns. Using these preliminary experiments, it is shown that the classification accuracy of a real-valued negative selection system is dependent on the shape of the detector chosen when moving into higher dimensions, and that the hypersteinmetz, specifically, provides the expected improvements over against the hypersphere detector.

2 Background

In order to better demonstrate the objective of the current research, a background of research within applicable fields of study is important. The following chapter aims to lay the essential framework of knowledge required to sufficiently support the thesis statement involving detector shapes within a high-dimension artificial immune system.

First, network intrusion detection systems are defined. Then, considerations involving the selection of a network intrusion dataset are discussed. Finally, an introduction to the field of artificial immune systems is provided, along with a presentation of research explaining the problems specific to artificial immune systems in high dimensions.

2.1 Network Intrusion Detection Systems

Before delving into how a network intrusion detection system can aid the information resource management (IRM) process, an understanding of what network intrusion detection systems are and how they work is essential. The following sections discuss intrusion detection systems, distinguish between intrusion detection and intrusion prevention, explain differences between network and host-based systems, differentiate anomaly and signature-based methods, and touch on a few of the computational methods currently used as the backbones of these systems.

According to the National Institute of Standards and Technology (NIST), intrusion detection is

> "The process of monitoring the events occurring in a computer system or network and analyzing them for signs of possible incidents, which are violations or imminent threats of violations of computer security policies, acceptable use policies, or standard security practices." [39]

Therefore, An intrusion detection system (IDS) is a system designed to perform the process of intrusion detection. Namely, an IDS monitors network or computer system information to determine if incidents are occurring and then acts upon or alerts someone to act upon that information. Intrusion detection is simply the process of determining that an incident has occurred, is occurring, or will occur.

An additional layer often added to the intrusion detection process is that of the intrusion prevention system (IPS). The purpose of an IPS is to not only act as a detector, but also as a preventer of intrusion. An IPS can act either after detection of an intrusion–blocking it from continuing–or before an event has taken place–shoring up the network or computer system from a potential attack [31]. Going forward, the rest of this discussion assumes an IDS, rather than an IPS, is the subject.

The next delineation that must be drawn regards network and host-based detection systems. Network-based IDSs (NIDS) monitor network traffic on a specific segment within a larger network system; they then classify that traffic, identifying many different types of events that may interest network management personnel. Contrarily, host-based IDSs (HIDS) reside on a single host system (e.g. a personal computer, a server, etc.) and monitor characteristics of the host, and events that occur therein, for suspicious activity. In addition to monitoring the network traffic for that specific host, an anomaly-based IDS can also look at system logs, processes, system files, access permissions, and application permissions in order to detect abnormal behavior [41]. A network-based IDS is the subject of this research.

The last distinction of different IDS methods is signature-based versus anomaly-based detection. Signature-based detection "is a technique for intrusion detection that relies on a predefined set of attack signatures [37]." In this scheme, the IDS attempts to match current network packets or traffic patterns to predefined attack or anomaly signatures in a database. The burden therefore is in keeping up with the ever increasing

number of intrusion signatures [40]. In contrast, anomaly-based detection relies on a pre-defined baseline of normal network traffic. The system compares incoming network traffic to the baseline. If the incoming traffic is sufficiently different than the baseline, it is classified as anomalous. The key benefit to an anomaly-based IDS over against a signature-based system is that it has the potential to detect new and emerging threats that have not already been defined in a signature database. However, it can be hard to create a sufficient baseline of normal traffic, thus causing the system to suffer from a plethora of false alarms [37]. Despite the drawbacks, though, anomaly-based systems currently provide the greatest area for research, as signature-based systems such as SNORT cannot detect newly emerging threats [31].

Within anomaly-based NIDS, there are several methods used to detect anomalous network traffic. The full extent of these methods is not reviewed and a good discussion of them can be found in [53]. There are several areas of artificial intelligence research which attempt to solve the intrusion detection problem with varying degrees of success. One particular method which has shown initial promise takes the approach of modeling the human immune system in order to detect anomalous network traffic.

2.2 Dataset Selection

To choose a dataset, an investigation of currently available datasets is in order. This discussion begins with a look into the problems of data collection, and then delves into an examination of a number of currently available datasets. There are three types of problems with using real-world datasets: privacy and anonymization of data, unavailability of data, and the issue of moving targets [19].

One of the most important topics in any real-world network traffic capture is that of the legal issues surrounding privacy and anonymization of the data. Ohm *et al* [36] demonstrate the tremendous legal concerns surrounding the use and sharing of private organizations' network traffic. There are regulations at all levels of government

surrounding the collection of network data, and–in almost all cases–the anonymization of addresses is essential. This is important to note, as some of the most essential data features for network intrusion detection can come from specific internet protocol (IP) and media access control (MAC) address data. Anonymizing or scrubbing this data can cripple the operation of a network intrusion detection system.

Legal issues can also create a second problem: unavailable data. Because of the privacy regulations, not only must data be anonymized, but in most cases it cannot be released to outside organizations at all. For this reason, a large portion of the novel data sets used in emerging research are home-grown. Lastly, it is important to realize that the definition of network traffic is a moving target. Some datasets that are currently used for benchmarking are more than ten years old and can no longer accurately demonstrate real-world network traffic [19].

Because of the problems with real-world network traffic, it might seem a logical conclusion to create synthetic traffic. However, there are additional problems with creating synthetic datasets, namely background traffic generation and attack traffic generation. Creating normal traffic requires the use of either a large user-base or a traffic generation tool. One of the problems with using a tool is that it can be difficult to represent the distributed nature of real networks. Real networks exist not only in disparate geographic locations, but also in different network locations. Moreover, truly representing the cacophony of actual user traffic can be difficult. On the other hand, relying on real users to create fake network traffic is obviously constrained by the availability of the given user set for this purpose.

Additionally, creating representative attack data is an extremely difficult problem. There are an infinite number of potential attack vectors available to malicious actors. From distributed denial of service attacks to structured query language (SQL) injection attacks, there is no way to predict and/or guard against all potential attack vectors.

Therefore, choosing an effective representative subset of these potential vectors is an extremely difficult task [3].

2.2.1 MIT-DARPA Sets. Two datasets that pertain to this discussion form the basis for a large portion of the intrusion detection dataset body of knowledge. Both datasets were created by the Massachusetts Institute of Technology (MIT) Lincoln Labs in conjunction with the Defense Advanced Research Projects Agency (DARPA). The first was created in 1998 and the second in 1999. Consequently, the datasets are commonly referred to as the MIT-DARPA '98 and MIT-DARPA '99 datasets respectively, and both can be found online at http://www.ll.mit.edu/mission/communications/ist/CST/index.html. As stated in [28], the main goal of creating these datasets was "to drive iterative performance improvements in participating systems by revealing strengths and weaknesses and helping researchers focus on eliminating weaknesses." In order to meet this goal, the research team created two successive datasets to set a baseline for testing different intrusion detection systems.

Both datasets were created using similar methodology, with the MIT-DARPA '99 data collection accounting for some problems that were present in the MIT-DARPA '98 dataset. The datasets were built to model the standard network traffic of a United States Air Force (USAF) base, and traffic was simulated in order to account for privacy concerns. The MIT-DARPA '98 dataset consists of 32 attack vectors spread out over a seven week period. The attacks are interspersed with normal network traffic and network packet data was captured using tcpdump, with approximately 4 Gigabytes (Gb) of data in total [27, 28, 46]. The MIT-DARPA '99 dataset consists of 200 instances of 58 attack vectors launched over a five-week period of time. The packet data was captured using tcpdump and is divided into a three-week training data subset and a two-week testing data subset [29, 32].

A third dataset was created by combining network packet based features of the MIT-DARPA ’98 dataset with traffic and content based features derived from the original data [46]. This new dataset was used for the The Third International Knowledge Discovery and Data (KDD) Mining Tools Competition. It is known as the KDD Cup ’99 dataset and can be found at the University of California at Irvine Machine Learning Repository: http://archive.ics.uci.edu/ml/databases/kddcup99/kddcup99.html. The purpose of the competition was to find the best method for intrusion detection by distinguishing “bad” network traffic from “good.” Specifically, the KDD Cup ’99 data contains about five million “connection vectors.” Each vector contains 41 features and is labeled not only as normal or attack, but also with the specific type of attack.

Although the KDD Cup and MIT-DARPA datasets are the most widely used within the network intrusion detection community, there have been several problems noted with the data. Three underlying problems with the data generation were proposed in [32]. First, it was suggested that the data did not truly represent real network traffic, and that the methods used for modeling real-world traffic patterns were never fully explained. Secondly, there is no evaluation of the effectiveness of the tcpdump feature used to capture the packets, as it is known to drop packets during intervals of high traffic. Lastly, and most importantly, there was never a solid definition put forth for each of the different types of attack [32, 46]. Additionally, the traffic data for both MIT-DARPA datasets was created pre-2000. Therefore, the data is over 10 years old. As a result it does not properly convey modern real-world traffic patterns and makeup.

Further problems were noted specific to the KDD Cup ’99 dataset. Two specific areas of problem include the presence of redundant records which can bias learning algorithms and the relative ease of classification that results from the default locations of attacks within the training and test data [46]. This analysis was used to form a new dataset from the KDD Cup ’99 data. The NSL-KDD dataset, currently housed at

http://iscx.ca/NSL-KDD/, aims to remove these issues. It still relies on the same 41-feature vectors of the original KDD Cup '99 data, but removes redundant records and recreates the training and test sets in a more challenging order. It is important to note that the NSL-KDD dataset does not solve any of the dataset creation problems that McHugh outlined in his analysis of the MIT-DARPA '98 and '99 datasets [32]. Mainly due to the fact that it is the most commonly cited dataset and ease of access, the KDD Cup '99 dataset is used as the primary network intrusion detection dataset for the purposes of this research. Further information involving the pre-processing of the dataset for use in the specific system created is discussed in the following chapter.

2.2.2 High-dimensionality of network intrusion datasets. Network intrusion datasets tend to have many features. These features represent many different measurable characteristics of both individual packets and network traffic subsets. For example, while the KDD Cup '99 dataset has 41 features, other datasets contain as many as 249 features [34]. The problem with datasets that contain many dimensions is the "curse of dimensionality." First proposed in [4], the curse of dimensionality is tied to the problem of exponential growth. As dimensions are added, the number of computations needed to perform basic comparisons while working with the data increases exponentially and computational power and memory are consumed quickly.

In most cases, the dimensionality problem is addressed by reducing dimensionality by selecting an ideal subset of features [11]. Due to current computing constraints this is essential–and will be for the foreseeable future. However, less research has addressed solutions for how to work more effectively with higher-dimensional data. One benefit of higher dimensions is specific to anomaly detection. Since anomaly detection systems are trained on normal data, anomalous patterns are not known. Therefore, it is difficult to determine which features best separate the anomalous traffic from the normal traffic. Since the scope of the current research aims to avoid an assumption of prior knowledge of attack

network traffic, it is important to use as many features as possible. The current research attempts to address some of the problems inherent with high-dimensional data, especially as it pertains to both network intrusion detection and artificial immune systems.

2.3 Artificial Immune Systems

The following sections provides a background of artificial immune systems and begin to explain the challenges in implementing these systems in high-dimensional spaces. First, a brief overview of artificial immune systems, in general, and real-valued negative selection, in particular, is performed. Then, the discussion investigates problems presented by high-dimensional real-valued negative selection systems. Finally, a few competing approaches to solving the problems presented in high-dimensional real-valued negative selection systems are presented.

2.3.1 AIS and negative selection. The biological immune system(BIS) is one of nature's anomaly detection systems. The basic role of the biological immune system(BIS) is to recognize all cells within the body and classify them as self or non-self [9]; the cells used to perform the recognition task are known as *antibodies*. Many organs, cells, and processes interact with these antibodies in order to create a robust immune system.

Many BIS processes have been modeled artificially, but the most common is that of negative selection. Negative selection is a biological process through which antibodies are "trained." The process, which takes place in the thymus, begins with a newly-created, *naive*, antibody. The naive antibody is presented protein strings that represent self cells that are commonly found in the host body. If the antibody detectors any of the self strings, it is "negatively selected" (discarded). This process is repeated continually in order to create a set of antibodies that does not recognize and thereby destroy healthy self cells.

The negative selection process was first adapted to a computational model by by Forrest *et al* [13]. The first phase of the algorithm creates a set of nonself detectors by

creating detectors and negatively selecting those that detect self point. A large set of these detectors is created to model the complement of the set of self points within a dataset. In the next phase, the detector set classifies a set of test points as self and nonself. The negative selection algorithm was created to operate on sets of binary strings, where each binary string represented a data point within the dataset.

2.3.2 Real-Valued negative selection. The first attempt to computationally model the BIS into a real-valued space was that of Perelson and Oster [38]. They presented the concept of *shape-space* as an *n*-dimensional Euclidean vector space. In shape-space, the Euclidean distance between two points represents the *affinity*, or similarity, between those points. Each feature in a dataset can be mapped to a dimension in shape-space. Shape-space is similar to the concept of feature-space in pattern recognition, but the range of shape-space is typically limited to the region of feature-space within which values for each feature can feasibly fall.

By combining the ideas of real-valued shape space along with the negative selection algorithm, Gonzalez *et al* [14] created the real-valued negative selection algorithm (RNS). RNS uses *hyper-spheres* to define detectors. A random vector is selected to represent the center of a hypersphere, and a radius is defined to represent an affinity threshold. The detector generated then defines all points within this hypersphere as non-self. A randomly-generated detector is compared against all self points. If the detector matches self, the detector is iteratively moved away from self points toward a location of the most separation possible from other detectors.

A subsequent approach to randomizing detector generation is to use a method known as *randomized RNS* [15]. In this approach, Monte Carlo integration is used to determine the size of self and non-self within the given feature space, then a number of randomly placed detectors are chosen according to Monte Carlo integration calculations. Simulated annealing is then applied to the random detectors in order to space them out as evenly as

possible over the non-self space. One problem with this method is that it requires a set of fixed-sized detectors, and this fixed size can cause problems with ensuring that all space is accurately covered.

In order to solve fixed size detector problem, the *V-detector algorithm* [22] was proposed. A V-detector is a variable-sized detector that is used to achieve the largest possible coverage of non-self space using the smallest number of detectors. In addition to moving detectors around the non-self space to create a greater spread–as in randomized RNS–the radii of detectors can be lengthened or shortened to provide a more accurate coverage. This approach yielded mixed results in practice. It provided for more accurate non-self/self differentiation, however, two problems arose. First, overlapping detectors can occur when two detectors' radii are increased. This overlap causes redundancy in the detector set and thus wasted computation. Additionally, outliers in the self set can cause the detectors to shrink and replicate in order to fill a space that would be better represented by a large detector.

2.3.3 Problem with RNS in high dimensions. Several authors have pointed out the problem of the curse of dimensionality within real-valued negative selection as it applies to the growth of shape-space [15, 23, 24]. In order to improve computational efficiency and feasibility of the ensuing algorithms, several approaches have been used to try to reduce the size of real-valued shape-space in high-dimensions. The most common approach is to reduce the number of dimensions through feature selection [11]. Other approaches include scaling of dimensions in order to reduce unoccupied space [30], and reducing the complexity of the any ensuing algorithm by reducing the number of self points present in shape-space [54].

Stibor *et al* [45] further honed the analysis of how the curse of dimensionality affects real-valued negative selection a step further by comparing the growth of shape-space to that of the detectors placed therein. Assuming the length of the radius remains constant, it

is shown that the *content*, the *n*-dimensional extension of the concepts of area in two dimensions and volume in three dimensions, of a hypersphere approaches zero as the number of dimensions approaches infinity. Since most algorithms used in real-valued negative selection use hyperspherical detectors, it is demonstrated through a ROC analysis that the classification results of traditional RNS-based algorithms suffer due to the diminishing content of hyperspheres.

2.3.4 Approaches to combat problems with RNS in high dimensions. There have not been many attempts made to address the problem of hypersphere growth in comparison to shape-space as dimensionality increases. Stibor *et al* [44] chose to create a real-valued positive selection system. In this system, each self point is treated as the center of a detector of self. Although this approach removes the problem of covering the nonself portion of shape-space, it fails to address the fundamental problem with hyperspheres. Rather than eliminating the problem of hyperspherical content approaching zero, it has just been transferred from nonself space to self space.

The most direct attempt made to combat the hyperspherical growth problem is that of applying the concept of distance norms to detector shapes [6]. Chmielewski and Wierzchon first define a real-valued detector as a point centered at a real-valued vector with an affinity threshold that is not directly tied to Euclidean distance, but rather to a distance based on either the *Minkowski norm distance*. The distance between two *n*-dimensional points $x = [x_1, x_2, ..., x_n]$ and $y = [y_1, y_2, ..., y_n]$ is determined using the The Minkowski norm of order m (L_m-norm distance) in shape-space, which is defined in Equation 2.1.

$$L_m(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^m \right)^{\frac{1}{m}} \tag{2.1}$$

The Minkowski norm distance is equivalent to Manhattan distance when $m = 1$ and Euclidean distance when $m = 2$. Further, the shape of a detector depends on the value of m.

The properties of detector shapes with $m \geq 1$ were addressed by Ji and Dasgupta [24]. However, their analysis of the Minkowski norm distance was done for the purposes of addressing the speed of their algorithm. Therefore, an analysis of Minkowski norm detectors with $m \geq 1$ was not applied to classification results.

The properties of Minkowski norm detectors with $0 \leq m < 1$ have been discussed by Aggarwal *et al* [1] and is applied to real-valued negative selection by Chmielewski and Wierzchon [6]. It is shown that reducing the value of m directly results in higher classification accuracy in high-dimensional spaces. However, there is a tradeoff that occurs between effectiveness and time complexity, as the efficiency of the underlying algorithms decreases as m decreases.

3 Approach and Methodology

The following chapter builds upon the framework set in the preceding chapters to explain the methods used to verify the thesis, “Detector shape is fan extremely important factor in the effectiveness of a real-valued negative selection system as the number of dimensions of data increases, especially in comparison to other factors such as radius size and coverage factor.” The methods described herein build upon the foundational background overviewed in Chapter 2.

First, the real-valued negative selection algorithm used in this research is presented, followed by a discussion of the properties of the detector shapes in the real-valued negative selection system in high-dimensional space. Then, a detector radius sizing method is discussed. Next, the concept of coverage factor is introduced. Limitations of the radius sizing methods as the system moves into higher-dimensional spaces are then explained. Finally, an experiment is laid out to compare the different detector shapes presented, in order to determine how detector shape influences classification accuracy in high-dimensional spaces.

3.1 Negative selection system overview

In order to perform the task of network intrusion detection, an artificial immune systems approach was chosen, specifically a naive real-valued negative selection system. This real-valued representation of space is known as shape-space, which is a subset of feature-space where bounds are placed on each feature in order to create an n-dimensional orthotope [38]. The system is modeled after the real-valued negative selection algorithm (RNS) [15], where the system randomly creates a set number of hyperspherical detectors and places them in the regions of shape-space not occupied by self points. Then, any time that a point is detected by one of the hyperspherical detectors, the point is classified as nonself.

Additionally, the created system does not attempt to address the problem of detector overlap, addressed in [15] and [25]. The current research aims to address specific problems that arise from implementing an AIS in high dimensional spaces. It is not necessary to solve the overlap problem in order to address the geometrical problems with AIS in high-dimensional spaces. The real-valued negative selection algorithm used for all of the proceeding experiments is shown in Algorithm 1, where p is a data point in the shape-space, P is a set of data points in the shape-space, P_r is the set of training points, P_s is the set of test points, η is a pre-determined number of detectors, c represents a point in shape-space, det_c is the detector centered at c, and D is a set of detectors.

3.2 Detector shapes in high-dimensional space

Stemming from the problems of high-dimensional data discussed in Section 2.3.3, previous research has shown poor results when scaling real-valued negative selection algorithms to higher-dimensional spaces. It is proposed that, the goals in selecting a detector shape are threefold. First, it should be computationally easy to determine if a point lies within the detector. Second, the detector should not be biased toward points in any given dimension or set of dimensions. Third, the content of the detector, as defined in Section 2.3.3, should grow proportionally to the content of shape-space as the dimensionality of the shape-space increases. The following sections define the terms computational complexity, feature bias, and content ratio as they pertain to detector shape.

3.2.1 Computational complexity. A limiting factor within AISs, and negative selection in particular, is the number of detectors that can be maintained simultaneously. Due to limited computing power, a maximum number of detectors is imposed on any negative selection system. In order to determine the maximum number of detectors allowed, the computational complexity of each detector must be known. The *computational complexity* of a detector shape is defined herein as the number of

Algorithm 1 Negative selection algorithm

Create training dataset P_r consisting of only self points and testing dataset P_s consisting of both self and nonself points need to be classified

repeat

Randomly select a center point c within the confines of the shape-space.

while det_c detects any points in P_r **do**

Select a new random value for c

end while

Add det_c to the set of detectors D

until η detectors have been created

for each point p in P_s **do**

for each detector $det \in D$ **do**

Determine if p is detected by det

end for

if p was detected by any det **then**

Label p as nonself

else

Label p as self

end if

end for

computations needed in order to determine if a point falls within the given detector. The number of computations needed is then used to create an upper bound on the growth rate of the complexity as a function of the number of dimensions, which is reported in *O*-notation [8].

3.2.2 Feature bias. Based on the works of Clark and Evans [7] and Perelson and Oster [38], *affinity* in shape-space is based on the assumption that points in real-valued shape-space that lie near each other are more closely related than those that are farther. For this reason, the distance between two points is relevant for AIS self/nonself determination. Thus, the shape of a detector can influence whether a point is classified as self or nonself, and *feature bias* is therefore defined to account for this dilemma. A detector is biased toward a certain feature if it has a large Euclidean distance between points in one feature (i.e. dimension) and a small distance in others. If a detector is biased, the orientation of the detector or of the points around a fixed detector influence the ability of a detector to recognize an antigen. Figure 3.1 illustrates this point by showing a case where the orientation of two points around a square detector, rather than distance between the points alone, determines whether or not the detector survives negative selection.

Feature bias, b, is defined as the ratio of maximum sensitivity to minimum sensitivity, where sensitivity, δ, is the distance between the center point of a detector and a given point on the surface of the detector. Equation 3.1 defines the bias ratio. An ideally biased shape would have a bias of $b = 1$, which would mean that it has an equal sensitivity in all dimensions.

$$b = \frac{\delta_{max}}{\delta_{min}} \tag{3.1}$$

3.2.3 Content ratio. Content, C, is the n-dimensional generalization of the concept of area in two dimensions and volume in three dimensions. As stated earlier,

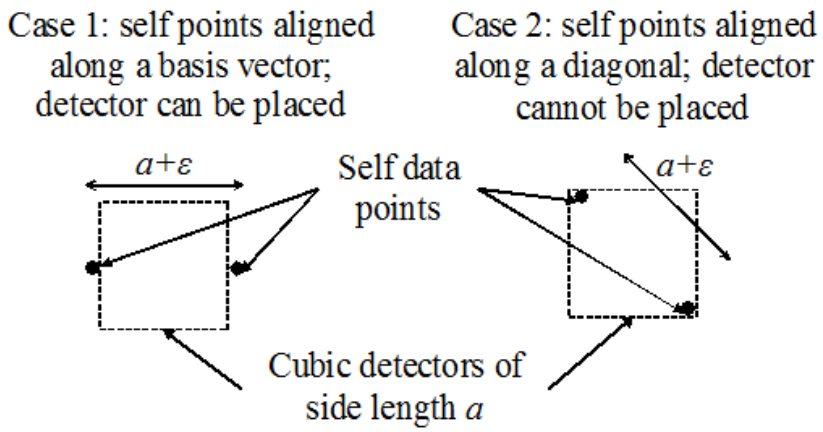


Figure 3.1: Illustration of the implications of feature bias in a square detector during negative selection.

shape-space is a subset of feature-space where bounds are placed on each feature in order to create an *n*-dimensional orthotope. The content of shape-space, C_{ss}, is shown in Equation 3.2, where p_i represents a point in the *i*-th dimension, β_i is the side length of the *i*-th dimension, and *n* is the total number of dimensions in shape-space.

$$C_{ss} = \Pi_{i=1}^{n} (max(p_i) - min(p_i)) = \Pi_{i=1}^{n} \beta_i \tag{3.2}$$

A shape-space element is defined as an orthotopic subset of shape-space with side-length that is a fraction of the side-length of shape space. Thus, the side-length of a shape-space element can be represented as $\frac{\beta}{\lambda}$; where λ, the number of shape-space elements contained in one dimension of shape-space, is greater than one. The content of a shape-space element, C_{el}, is defined in Equation 3.3, where λ_i is chosen such that $\frac{\beta_i}{\lambda_i}$ is a constant, thus creating a shape-space element that is cubic in *n* dimensions.

$$C_{el} = \Pi_{i=1}^{n} \frac{\beta_i}{\lambda_i} \tag{3.3}$$

A single detector is sized, in the following equations, relative to a shape-space element such that the shape-space element is the smallest n-dimensional hypercube that

completely contains the given detector. Figure 3.2 demonstrates a spherical detector contained within a shape-space element.

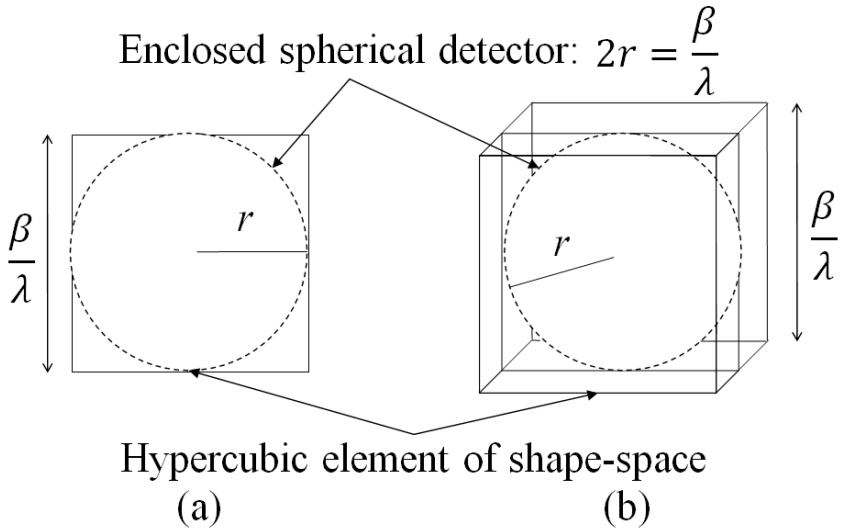


Figure 3.2: Illustration of a spherical detector within a shape-space element in (a) two dimensions and (b) three dimensions.

The number of shape-space elements contained in *n*-dimensional shape-space, $E(n)$ is a function of the number of dimensions, as is shown in Equation 3.4.

$$E(n) = \lambda^n \tag{3.4}$$

This reveals that the number of shape-space elements required to completely fill the shape-space grows exponentially with regard to the number of dimensions, completely independently of the detector shape.

Content ratio, γ is defined as the portion of the content of a shape-space element that is contained within a single detector, and is shown in Equation 3.5, where C_d is the content of a detector, and C_{el} is the content of an element of shape-space.

$$\gamma = \frac{C_d}{C_{el}} \tag{3.5}$$

Content ratio is an indicator of the number of detectors required to completely cover shape space when compared to the number of shape-space elements, $E(n)$, required to do the same. The largest possible content ratio is one, which would require $E(n)$ non-overlapping detectors to cover shape-space. A content ratio less than one would require a number of non-overlapping detectors greater than $E(n)$ to cover the entire shape-space. It is proposed that larger values of γ are preferred, because a higher γ implies that each individual detector has a higher likelihood of anomaly detection within the bounding shape-space element.

The use of content ratio allows the ability to asses the number of detectors needed as dimensionality increases. If a detector's content ratio decreases as dimensionality increases, then the number of detectors would need to be increased faster than the growth of $E(n)$ just to keep the same likelihood of anomaly detection.

3.2.4 A comparison of three detector shapes. Using the concepts of computational complexity, feature bias, and content ratio, the following section compares the two most common detector shapes: hyperspheres and hypercubes. Then a third shape, the hypersteinmetz, is proposed to potentially balance the tradeoffs between hyperspheres and hypercubes.

An n-dimensional hypersphere is a generalization of the two-dimensional circle and three-dimensional sphere to $n \geq 4$ dimensions. It is therefore defined by equation 3.6, where $[c_1, c_2, ..., c_n]$ defines the hypersphere's center point c in n-dimensional space, $[p_1, p_2, ..., p_n]$ is a point p that is within the hypersphere, and r represents the radius of the hyper-sphere [52].

$$r^2 = (p_1 - c_1)^2 + (p_2 - c_2)^2 + \cdots + (p_n - c_n)^2 \tag{3.6}$$

Euclidean distance between points p and c, d_{pc}, for an n-dimensional space is defined by equation 3.7, where d_i is the Euclidean distance from point p to c in dimension i.

$$d_{pc} = \sqrt{{d_1}^2 + {d_2}^2 + \cdots + {d_n}^2} \tag{3.7}$$

A point p falls within a given hypersphere detector if its Euclidean distance from the detector's center c is $\leq r$. This calculation requires three operations for every dimension or $3n = O(n)$ operations. This is a linear upper bound and is therefore considered computationally feasible.

Since the surface of a hypersphere is one radius distance from the center point in all directions, one benefit of using hyperspheres is that they have a feature bias of $b = 1$; this is demonstrated in Equation 3.8.

$$b_{hs} = \frac{\delta_{max}}{\delta_{min}} = \frac{r}{r} = 1 \tag{3.8}$$

Since $b_{hs} = 1$, we know that the hypersphere is not biased toward any specific dimension in shape-space.

The primary drawback to using hyperspheres is that as the number of dimensions of shape-space increases, the content ratio of a hypersphere goes to zero. In [45] it is shown that for each length of r there exists a dimension n for which the content of the n-dimensional hypersphere, C_{hs}, is maximized. After this point is reached, C_{hs} goes to zero as n approaches infinity. This is shown in Equation 3.9.

$$n \overset{lim}{\rightarrow} \infty \; C_{hs} = 0 \tag{3.9}$$

The content ratio of a hypersphere in comparison to a shape-space element, therefore, decreases exponentially with respect to the number of dimensions n. The content ratio of a hypersphere, γ_{hs}, is as shown in Equation 3.11 and demonstrated in Figure 3.3. $\Gamma(\alpha)$ is the mathematical *gamma function*, which has factorial growth with respect to α, as shown in

Equation 3.10 [33].

$$\Gamma(\alpha) = \begin{cases} 1 & \text{if } \alpha = 1 \\ (\alpha - 1)\Gamma(\alpha - 1) & \text{if } \alpha > 1 \end{cases} \tag{3.10}$$

$$\gamma_{hs} = \frac{C_{hs}}{C_{el}} = \frac{\pi^{0.5n}}{n(2^{n-1})\Gamma(0.5n)} \tag{3.11}$$

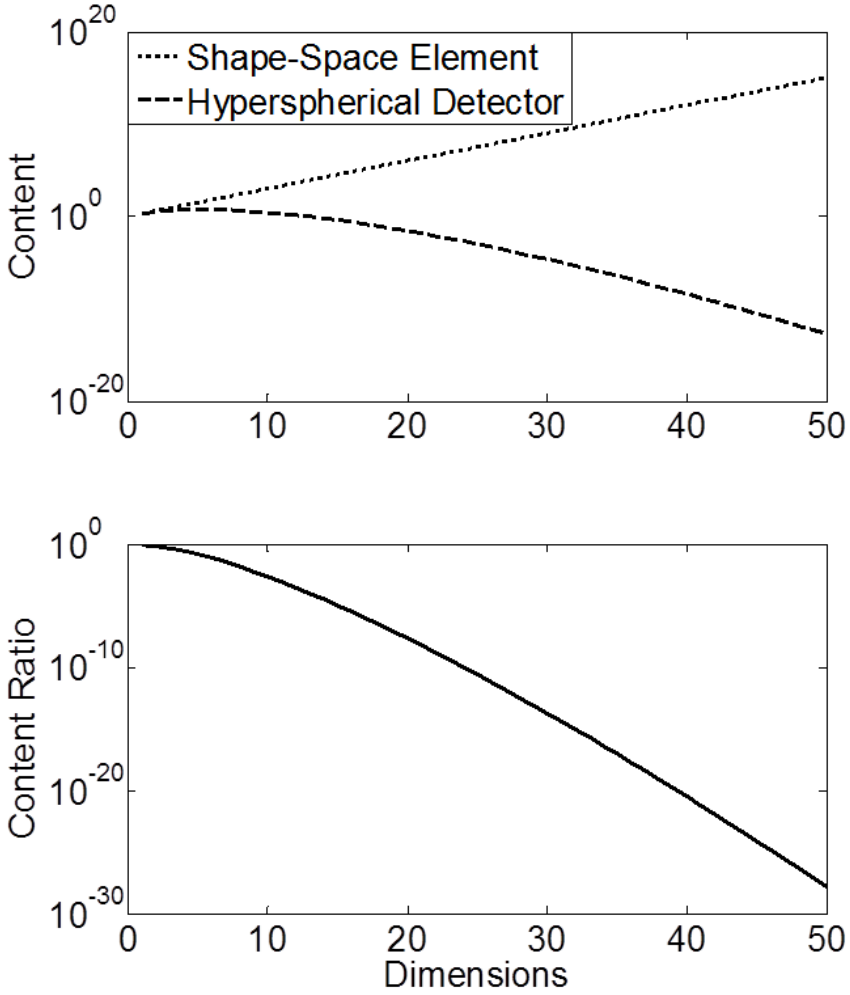


Figure 3.3: Content and content ratio of a hypersphere with $r = 1$ as the number of dimensions increases.

An n-dimensional hypercube is a generalization of the two-dimensional square and three-dimensional cube to $n \geq 4$ dimensions. For the purpose of this discussion, a hypercube is assumed to be aligned with the axes of shape space. As such, a hypercube can be projected onto each dimension as a line segment. A given point y, therefore, falls within a given hypercube detector if for every dimension it falls within the projected line segment representing the cube. Each point must be compared to a lower and upper bound in each dimension. So, the number of comparisons needed is twice the number of dimensions. This provides a computational complexity bounded by a growth rate of $3n = O(n)$, making it computationally equivalent to the hypersphere.

The feature bias of a hypercube can be determined by applying the method shown in Equation 3.1. The shortest distance from a hypercube's center to the surface, δ_{min}, is equal to half of the side length, a. The longest distance from a hypercube's center to the surface, δ_{max}, is from the center to the corner. The feature bias of a hypercube, b_{hc}, is shown in Equation 3.12.

$$b_{hc} = \frac{\delta_{max}}{\delta_{min}} = \frac{\sqrt{n\left(\frac{a}{2}\right)^2}}{\frac{a}{2}} = \sqrt{n} \tag{3.12}$$

Therefore, as the number of dimensions increases, the hypercubic detector's feature bias increases.

Because a hypercube is a specific instance of orthotope where all side lengths a_i are equal, it does not exhibit the decreasing content ratio of a hypersphere as dimensionality increases. We can see that a is a subset of shape-space such that $a_{hc} = \frac{\beta}{\lambda}$. Therefore, there must exist a shape-space element that contains the hypercube, such that $a_{el} = \frac{\beta}{\lambda}$. Based on Equation 3.3, we can substitute this value and see that a content ratio, γ_{hc}, of one is achieved in Equation 3.13

$$\gamma_{hc} = \frac{C_{hc}}{C_{el}} = \frac{a^n}{\left(\frac{\beta}{\lambda}\right)^n} = 1 \tag{3.13}$$

An n-dimensional hypersteinmetz solid is the orthogonal intersection of N cylinders, where N is equal to the ceiling of half of the number of dimensions, as shown in

Equation 3.14.

$$N = \left\lceil \frac{n}{2} \right\rceil \tag{3.14}$$

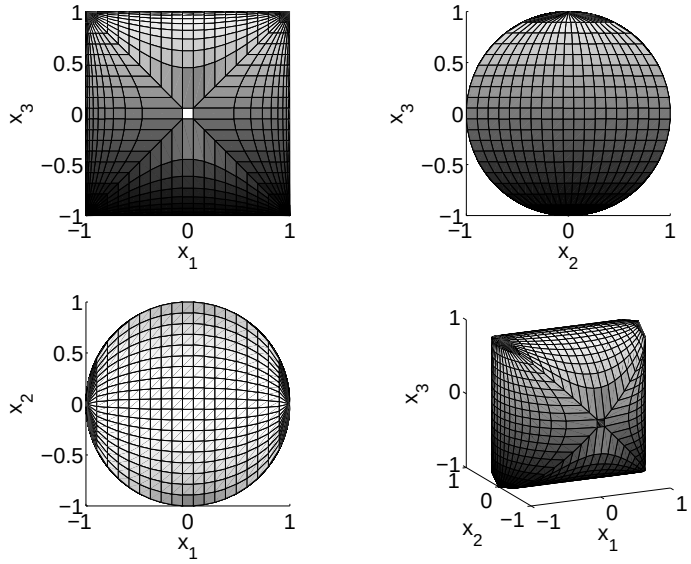


Figure 3.4: Hypersteinmetz solid shown from different angles.

An example of a hypersteinmetz is shown in Figure 3.4. For the current discussion, a unit hypersteinmetz aligned along the shape space dimensions is assumed. The *n*-dimensional hypersteinmetz is defined by the set of equations 3.15, where x is a point in the hypersteinmetz and x_i is the i-th dimensional value of x.

$$\begin{aligned} {x_1}^2 + {x_2}^2 &\leq r^2 \\ {x_3}^2 + {x_4}^2 &\leq r^2 \\ &\vdots \\ {x_{n-1}}^2 + {x_n}^2 &\leq r^2 \end{aligned} \tag{3.15}$$

In order to determine whether a point y resides within a given hypersteinmetz x, each of the $\left\lceil \frac{n}{2} \right\rceil$ cylinders is projected down to two dimensions. The point y is projected onto each dimensional pair. If all projections lie within the corresponding circles, then the point y is contained by the hypersteinmetz. Since determining if a point lies within a hypersphere requires $3n$ operations and a hypersteinmetz can be projected down to a set of two-dimensional hyperspheres, a hypersteinmetz detection computation requires $3 \cdot 2 = 6$ operations for each cylinder. In the worst case scenario, $6N$ operations would need to be performed. Therefore, the computational complexity of the hypersteinmetz is $6N = 6 \cdot \frac{n}{2} = 3n$, resulting in a worst case growth rate of $O(n)$. The growth rate of determining if a point lies within a hypersteinmetz is, therefore, computationally equivalent to determining the same for the hypercube and hypersphere.

The distance between the center point of a hypersteinmetz and the closest point on the surface is equal to the radius length chosen for all cylinders. Since we are assuming a unit hypersteinmetz, Equation 3.16 represents the minimum distance.

$$\delta_{min} = r = 1 \tag{3.16}$$

As Equation 3.17 demonstrates, the values for δ_{max} differ in odd and even dimensions.

$$\delta_{max} = \left\{ \begin{array}{ll} \sqrt{\frac{n}{2}} & \text{if } n \text{ is even} \\ \sqrt{\frac{n+1}{2}} & \text{if } n \text{ is odd} \end{array} \right\} = \sqrt{\left\lceil \frac{n}{2} \right\rceil} \tag{3.17}$$

The feature bias of a hypersteinmetz, b_{st}, is shown in Equation 3.18.

$$b_{st} = \frac{\delta_{max}}{\delta_{min}} = \frac{\sqrt{\left\lceil \frac{n}{2} \right\rceil}}{1} = \sqrt{\left\lceil \frac{n}{2} \right\rceil} \tag{3.18}$$

The following equations are presented in order to determine the content ratio of a hypersteinmetz. First, the content of a hypersteinmetz, C_{st}, can be found through iterative integration shown in Equation 3.19, where $y_i = \sqrt{r^2 - x_i^2}$.

$$C_{st} = \left\{ \begin{array}{ll} \int_{-r^2}^{r^2} \int_{-y_n}^{y_n} \cdots \int_{-r^2}^{r^2} \int_{-y_2}^{y_2} dx_1 dx_2 \cdots dx_n & \text{if } n \text{ is even} \\ \int_{-r^2}^{r^2} \int_{-y_n}^{y_n} \int_{-y_{n+1}}^{y_{n+1}} \cdots \int_{-r^2}^{r^2} \int_{-y_2}^{y_2} dx_1 dx_2 \cdots dx_n & \text{if } n \text{ is odd} \end{array} \right. \tag{3.19}$$

Evaluation of the integrals in Equation 3.19 results in the content expression in Equation 3.20.

$$C_{st} = \begin{cases} \left(\pi r^2\right)^{\frac{n}{2}} & \text{if } n \text{ is even} \\ \frac{16}{3} r^n \pi^{\frac{n-3}{2}} & \text{if } n \text{ is odd} \end{cases} \tag{3.20}$$

Equation 3.21 shows the content ratio for a hypersteinmetz, γ_{st}, obtained by substituting Equation 3.20 into Equation 3.5.

$$\gamma_{st} = \begin{cases} \frac{\left(\pi r^2\right)^{\frac{n}{2}}}{2^n} & \text{if } n \text{ is even} \\ \frac{16}{3} \frac{r^n \pi^{\frac{n-3}{2}}}{2^n} & \text{if } n \text{ is odd} \end{cases} \tag{3.21}$$

The optimal number of orthogonal cylinders to use in a hypersteinmetz solid detector is $\left\lceil \frac{n}{2} \right\rceil$, where n is the number of dimensions in the shape space. This is due to the fact that the feature bias does not change as cylinders are added, while at the same time the number of operations required increases and the content ratio decreases. Therefore, hypersteinmetz detectors for the purpose of the current research consists of $\left\lceil \frac{n}{2} \right\rceil$ orthogonally intersecting cylinders.

Comparing the three shapes, the hypersteinmetz balances the problems between the hypersphere and hypercube. Table 3.1 shows that the hyperspheres content ratio decreases factorially due to the presence of the gamma function in the denominator, while that of the hypersteinmetz only decreases exponentially. Additionally, the feature bias of the hypercube is reduced in the hypersteinmetz by a factor of $\sqrt{2}$. Although the hypersteinmetz does better in both regards, it does not approach the ideal content ratio of the hypercube or the ideal feature bias of the hypersphere.

3.3 Detector radius sizing experimental design

The discussion presented in Section 3.2 proposed that content ratio and feature bias have an adverse effect on detector shape as dimensionality increases. However, we investigate whether or not content ratio and feature bias translate directly into

Table 3.1: Table comparing the content ratio and feature bias of the hypersphere, hypersteinmetz, and hypercube.

	Hypersphere	Hypersteinmetz	Hypercube
Content ratio	$\frac{\pi^{0.5n}}{n(2^{n-1})\Gamma(0.5n)}$	$\frac{\pi^{0.5n}}{2^{n-1}}$	1
Feature bias	1	$\sqrt{\left\lceil \frac{n}{2} \right\rceil}$	$\sqrt{n}$

classification accuracy within a real-valued negative selection AIS, and the following sections describe a set of experiments designed to test this.

The first experiment performed to investigate the relationship between different detectors is to constrain the range of detector radii that can be used for further experiments. In order to provide a baseline for the minimum radius size for a detector, it is important to take into account two things. First, a radius size that is too large does not allow the detector to fall between data points that are separable. The tradeoff, however, is that a radius that is too small "overfits" the data by falling between data points that are similar. Overfitting causes the RNS system to classify points that are self as nonself. The following sections propose the nearest neighbor method for determining a minimum detector radius, and a maximum radius sizing method, wherein the largest possible detector is placed. Once these methods are explained, an experiment is designed to show that the range produced by the proposed minimum and maximum methods is a feasible sizing method for the purpose of classification in a real-valued negative selection system.

3.3.1 Minimum radius sizing using nearest neighbor (nn) method. Clark and Evans [7] noted that the most similarity to any data point is gained by looking at the data point closest to it. This phenomenon, known as the *nearest neighbor principle*, is also a fundamental assumption required in the real-valued negative selection approach to AIS, expressed by equating the concept of 'affinity' with Euclidean distance. Since real-valued

negative selection fills up negative space with detectors, we need to ensure that groups of similar points are not broken up by detectors. In order to do this, a nearest neighbor distance is computed to determine the minimum radius size allowed.

In order to determine how close together similar points are distributed, first a subset of points P is selected randomly from the set of self training points within the dataset. For each point $p \in P$, p's nearest neighbor q is found. The distance, d_{nn}, from p to q is calculated. Once all values of d_{nn} have been calculated, the mean distance is determined. This mean nearest neighbor distance then becomes r_{min}, the minimum radius size. See Algorithm 2 for further details.

Algorithm 2 Minimum radius sizing using nearest neighbors, where p and q are self data points, d_{nn} is the minimum distance from p to any q, and r_{min} is the minimum radius size for a detector.

repeat

Randomly select a data point p from the training dataset

for each other training data point q **do**

Determine the distance from p to q

end for

Determine the minimum pq distance, d_{nn}, and record it

until n nearest neighbor distances have been recorded

Set r_{min} to the mean of all d_{nn} values found

3.3.2 Maximum radius sizing using largest radius placement method. In order to determine the maximum for the radius range, the largest possible detector is placed. A detector with radius $r = \sqrt{n}$, where n is the number of dimensions of shape-space, is placed randomly in shape space. If the detector detects any self training points, its center point is moved to a new random location. If the detector cannot be placed in fewer than

10,000 attempts, the radius is reduced by $0.05 \cdot \sqrt{n}$ and placement is attempted again. This is repeated until a detector is successfully placed within 10,000 attempts. The radius length at this point is recorded as the maximum radius size, r_{max}. Whereas r_{min} is the same value regardless of detector shape, each detector shape has a different value for r_{max}. A further explanation of the method can be found in Algorithm 3.

Algorithm 3 Maximum radius sizing algorithm, where r_c is the radius of a given detector d_c centered at point c, n is the number of dimensions of shape-space, a is the number of center points tested so far, and r_{max} is the maximum detector radius length.

$r_c = \sqrt{n}$
while d_c has not been successfully placed **do**
  while $a < 10,000$ AND d_c has not been successfully placed **do**
    Randomly choose a new center point c
    if d_c does not detect any self training points **then**
      d_c has been successfully placed
      set $r_{max} = r_c$
    else
      $a = a + 1$
    end if
  end while
  if d_c has not been successfully placed **then**
    $r_c = r_c - .05\sqrt{n}$
    $a = 1$
  end if
end while

3.3.3 Detector radius sizing experiment. Current research addresses the need to search for best radius sizes within a real-valued negative selection system. One solution is to use the variable sized detectors presented by Ji and Dasgupta [25]. However, when using fixed-size detectors, the only method shown in the research is that of trial and error [15]. Additionally, within a variable-sized detector algorithm, all detectors must have an initial size. Therefore, constraining the range of values in which to search provides the first steps toward finding the best detector size for a given dataset. The following experiment is designed to verify that the proposed min and max detector sizing methods provide a reasonable range of radius lengths within which this search can be performed.

3.3.3.1 Experimental question. Do the nearest neighbor method for sizing the minimum radius length r_{min} and the largest detector placement method for sizing the maximum radius length r_{max} provide a good lower and upper bound respectively on the radii to test for the naive real-valued negative selection algorithm described in Algorithm 1?

3.3.3.2 Testable hypothesis. The optimal point, that point closest to the 0% false positive/100% true positive point via Manhattan distance, on the Receiver Operating Characteristic (ROC) curve comparing false positives and true positives for a given classification run falls somewhere between r_{min} and r_{max}.

3.3.3.3 Dataset. Datasets are chosen from the University of Memphis negative selection 2-D synthetic datasets [10]. These datasets have been used by several researchers in order to baseline and compare effectiveness of real-valued negative selection algorithms [25][55]. Each dataset involves a training dataset of only self points and a testing dataset including both self and nonself points. Self points are defined by some given shape. The datasets used for this specific research include the *Pentagram-big*, *Pentagram-bigneg*, *Comb*, *Combneg*, *Intersection-thick*, *Intersection-thickneg*, *Ring-thick*,

and *Ring-thickneg* datasets, see Figure 3.5. These specific shapes were chosen as they provide a good variance of different distributions of data points within shape-space. Each training dataset includes 1000 2-d self points and each testing set includes 1000 2-d points, 500 self and 500 nonself.

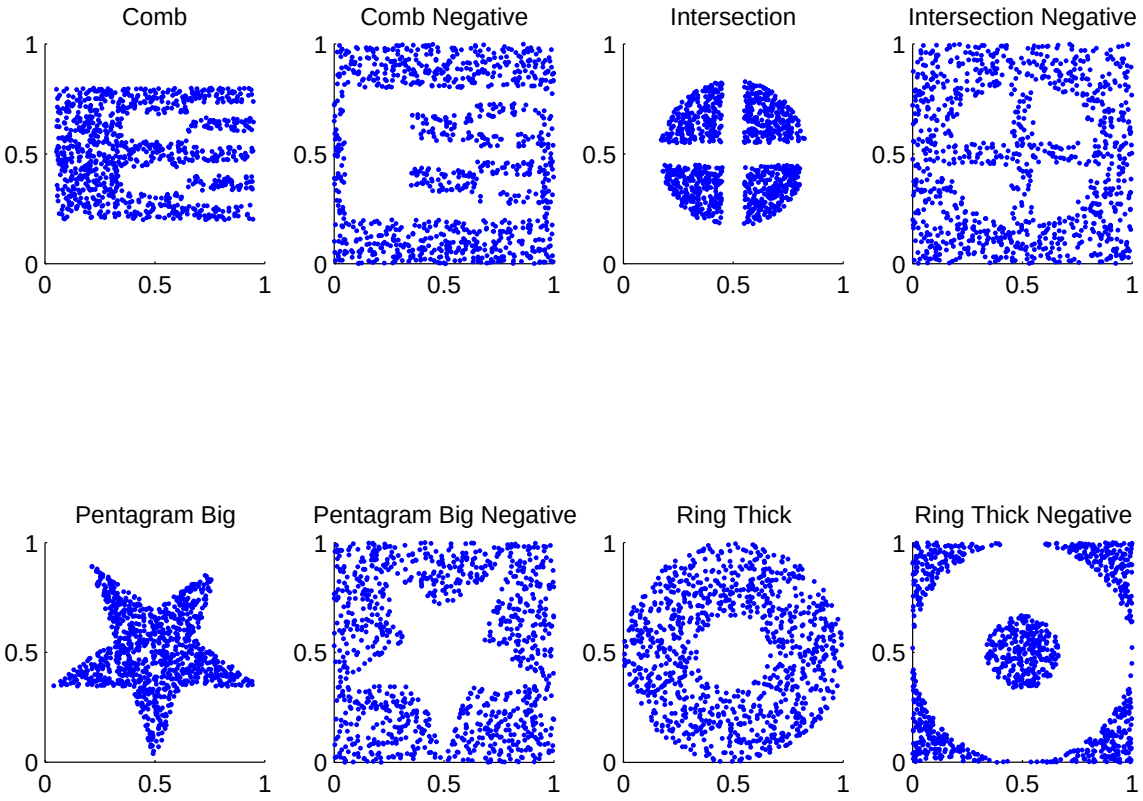


Figure 3.5: Datasets used for training, points represent self data.

3.3.3.4 Outline of experiment. For this experiment, the effects of detector shape is not investigated. Only a two-dimensional hypersphere detector is considered. The control variables are the percentage of the content of shape-space covered by the detector set (ignoring detector overlap), size of shape-space, and number of dimensions. The only independent variable in this experiment is detector size. In order to form a baseline for

comparison, a successively increasing set of radius sizes are also used. In addition to r_{min} and r_{max}, radius sizes from .01 up to and including r_{max}, incremented by .01, are tested.

The side-length of shape-space is set at 1 for all dimensions, thus the content of shape-space is one. The number of detectors created is equal to the number of detectors needed to create a detector set that holds exactly 200% of the content of shape-space, ignoring detector overlap. Therefore, the number of detectors, η, is equal to the content of shape-space divided by the content of a detector, multiplied by the coverage factor f. The coverage factor is equal to the percentage of content covered, c_{pct}, divided by 100 (see Equation3.22).

$$\eta = \frac{c_{pct}}{100}\frac{1}{2\pi r^2} = \frac{f}{2\pi r^2} \tag{3.22}$$

Additionally, for each radius size threshold, the experiment runs 10 times on each dataset.

Each run of the experiment executes Algorithm 1 recording the numbers of: (I) True positives - A data point was “NONSELF” and was classified as “NONSELF”, (II) True negatives - A data point was “SELF” and was classified as an “SELF”, (III) False positives - A data point was “SELF” and was classified as “NONSELF”, and (IV) False negatives - A data point was “NONSELF” and was classified as “SELF”. See Algorithm 4 for pseudo-code representation of this outline and Table 3.2 for tabular specification of experiment parameters.

3.3.3.5 Representation of results. For each dataset there are 10 runs; the number of true positives, true negatives, false positives, and false negatives are averaged over these 10 runs. The mean and variance for each set of runs are recorded. Thus, there are a mean number and variance of true positives, true negatives, false positives, and false negatives associated with each radius size of detectors tested. All of these statistics are collected for each of the datasets and are reported as follows.

Algorithm 4 Pseudocode of detector radius sizing experiment.

for each dataset **do**

for $r = 0.01 \rightarrow r_{max}$, by steps of .01 **do**

Set the number of detectors n such that the total content of all detectors is equal to 200% of the content of the shape-space

repeat

Execute Algorithm 1 and record the true positives, true negatives, false positives, and false negatives

until 10 tests have been run on each subset

end for

end for

Results are displayed in a series of two-dimensional receiver operating characteristic (ROC) curve plots. Each plot represents the results for one dataset (e.g. Comb, Ring Thick Negative, etc.). The x-axis shows the percentage of false positives–the percentage of points that were SELF and were classified as NONSELF (i.e. false alarms). The y-axis shows the percentage of true positives–the percentage of points that were NONSELF and were classified as NONSELF (i.e. detections). Each point on the plot represents one radius threshold.

3.4 Coverage factor experimental design

Where the first experiment aims to find a range of radius sizes to use in the real-valued negative selection system, the next experiment aims to determine how many detectors to produce once a radius size is chosen. Coverage factor was briefly introduced in the previous experiment. However, a full explanation is warranted. The content of a set of detectors C_D is the cumulative content of all detectors, disregarding detector overlap. Equation 3.23 shows C_d as a function of the number of detectors η and the content of an

Table 3.2: Detector radius sizing experiment design parameters.

Parameter	Values
Number of Dimensions n	2
Dataset	*Comb*, *Combneg*, *Intersection-thick*, *Intersection-thickneg*, *Pentagram-big*, *Pentagram-bigneg*, *Ring-thick*, *Ring-thickneg*
Detector Radius r	$0.01 \rightarrow r_{max}$, by steps of 0.01,
Shape-Space Content C_{ss}	1
Coverage factor f	2
Number of Detectors η	$\frac{f}{2\pi r^2}$
10 iterations are performed for each test. Measured outputs for each test are true positives, true negatives, false positives, and false negatives.	

individual detector C_d.

$$C_D = \eta C_d \tag{3.23}$$

The content of a set of detectors is also equal to the content of shape space C_{ss} multiplied by the coverage factor f, as shown in Equation 3.24.

$$C_D = f \cdot C_{ss} \tag{3.24}$$

For this reason, a coverage factor of 2 would represent a C_D that contains 200% of the content of shape-space, disregarding detector overlap.

3.4.1 Experimental question. How do we approximate the relationship between coverage factor and classification accuracy?

3.4.2 Testable hypothesis. Increasing coverage factor does not necessarily improve accuracy, as the addition of further detectors may cause the system to overfit the

self data points and also cause increasing detector overlap. However, the difference in accuracy between runs should decrease as detectors are added. Therefore, as the coverage factor is increased from 2 to 30, the standard deviation of the classification accuracy decreases. It reaches a point at which adding further coverage no longer reduces the standard deviation. Additionally, the mean of the true positives should increase, while the mean of false positives should decrease up to the point at which adding additional detectors does not provide additional benefit. Thus, the ROC curves of higher coverage factors should approach points closer to 0% false positives and 100% true positives.

3.4.3 Dataset. Fisher's iris dataset [12] is used. The dataset consists of 150 data points, 50 of each of three classes (Setosa, Virginica, Versicolour). The dataset is broken into a set of training and testing datasets as follows. Since real-valued negative selection acts on only two classes (self and nonself) each dataset is set aside one type of iris as nonself and combine the other two iris types into the self data points. For example, Setosa and Virginica are combined to create self in order to test classification of Versicolour as nonself. The larger dataset is broken up in this way to create three separate sets. This dataset was chosen for two reasons. First, it provides a distribution of data that is not synthetic, as were the datasets used in radius sizing experiment. Secondly, the dataset has been used numerous times and results can be compared with those of others.

Additionally, 90/10 cross-validation is used. Each experimental run trains on 90% of the data and test on 10%. In order to achieve this, the data sets are first each be broken into ten subsets. Each subset consists of a training subset and a testing subset. The training subset consists of 90 points (45 from each of the two self classes of iris) and each testing subset consists of 15 points (5 from each of the two self and one nonself iris classes). The ten datasets are created such that different points are chosen as test points for each subset. Therefore, there are 30 total subsets created.

3.4.4 Outline of experiment. The experiment runs with a four-dimensional hypersphere detector. The control variables are the size of shape-space, detector shape, and number of dimensions. The length of shape-space is set at 1 for all dimensions, thus the content of shape-space is one. Ten successive radii are tested, and for each radius 11 coverage factors are tested. For each set of parameters, the experiment runs 10 times on each dataset. Each run of the experiment executes Algorithm 1 recording the numbers of true positives, true negatives, false positives, and false negatives.

The algorithm is executed with η detectors, where η is calculated in the following manner. The content of a hypersphere in n dimensions is shown in Equation 3.25.

$$C_{hs} = \frac{2r^n \pi^{n/2}}{n\Gamma(n/2)} \tag{3.25}$$

The content of a detector set C_D is equal to the content of an individual detector C_d multiplied by the number of detectors in the set η, as shown previously in equation 3.23.

The number of detectors needed in an individual run, is calculated using Equations 3.24, 3.25, and 3.23. This results in Equation 3.26.

$$\eta = \frac{f \cdot C_{ss}}{C_d} = \frac{f \cdot n \cdot C_{ss} \cdot \Gamma(n/2)}{2 \cdot r^n \cdot \pi^{n/2}} \tag{3.26}$$

See Algorithm 5 for pseudo-code representation of this outline and Table 3.3 for tabular specification of experiment parameters.

3.4.5 Representation of results. Results are represented in two forms. A ROC curve of false positives versus true positives is created. The tunable statistic used to create each ROC curve is the size of the radius.

A low variance of classification accuracy between classification runs is desirable. Low variance means that regardless of the training points selected and the exact placement of the detectors, the same results are obtained every time. Therefore, the variance of the classification runs is important to record. In order to capture this, the following procedure

Algorithm 5 Pseudocode design of coverage factor experiment. r is the radius of the detector, f is the coverage factor, and η is the number of detectors.

Set the detector shape to hypersphere

Set the number of dimensions to 4

for $r = r_{min} \rightarrow r_{max}$, by steps of $\frac{r_{max} - r_{min}}{9}$ **do**

  for $f = 2, 4, 6, 8, 10, 12, 14, 15, 20, 25, 30$ **do**

    for each dataset: Setosa, Versicolor, Virginica **do**

      for each of the 90/10 cross-validation subsets **do**

        repeat

          Set η using equation 3.26

          Execute Algorithm 1 using η detectors and record the true positives, true negatives, false positives, and false negatives

        until 10 tests have been run on each subset

      end for

    end for

  end for

end for

is completed: The percentage of true positives and false positives are averaged across each radius/coverage factor pairing; a percentage is computed for each of the ten test runs for each of the ten test sets for each of the three iris types. The standard deviation across each radius/coverage factor pairing is recorded. Then, the mean standard deviation across all radii for a given coverage factor is reported. By comparing the standard deviations of both true and false positives across different coverage factors, a good coverage factor can be obtained for future tests.

Table 3.3: Coverage factor experiment design parameters.

Parameter	Values
Dataset	Setosa, Versicolor, Virginica
Detector Shape	Hypersphere
Number of Dimensions n	4
Detector Radius r	$r_{min} \rightarrow r_{max}$, by steps of $\frac{r_{max}-r_{min}}{9}$
Detector Content C_d	$\frac{r^4\pi^2}{2\Gamma(2)}$
Shape-Space Content C_{ss}	1
Coverage Factor f	2, 4, 6, 8, 10, 12, 14, 15, 20, 25, 30
Number of Detectors η	$\frac{fC_{ss}}{C_d}$
10 iterations are performed for each test. Measured outputs for each test are true positives, true negatives, false positives, and false negatives.	

3.5 Detector size and coverage factor in higher-dimensional space

Network intrusion datasets are inherently high-dimensional, as explained in Section 2.3.3. As AIS systems are applied in higher dimensional shape-space, there are considerations in addition to those inherent in different detector shapes that arise. First, memory constraints of the implementation system constrains the number of detectors that can be implemented, thus limiting the range of radius sizes that can be used to achieve a given coverage factor. Secondly, it is shown here that there is a relationship between the number of dimensions and effective coverage factor within an AIS.

3.5.1 Memory limitations on minimum radius size. Equation 3.26 reports the method for calculating the number of detectors needed within a run of Algorithm 1 given coverage factor, number of dimensions, and detector size. However, this number cannot always be implemented due to memory limitations. In order to use Algorithm 1, all

detector/test point pairs must be in memory simultaneously. Therefore, the maximum number of detectors allowed, is dependent on both the number of test points and the number of dimensions (i.e. how many comparisons must be made). The constraining factor in this regard is dependent on the memory of the computing and programming platforms being used. By taking into account the specific system capabilities, a maximum number of elements allowed in the arrays arises. Using this maximum, Equation 3.27 shows the exact relation between the maximum number of elements in an array E_{max} and the maximum number of detectors allowed η_{max}, where n is the number of dimensions and P_s is the number of test points. The number of test points is important even in a realtime system, due to the fact that the speed of network traffic determines how many points need to be calculated at once in order to avoid significant network slowdown.

$$\eta_{max} = \frac{E_{max}}{n \cdot P_s} \tag{3.27}$$

From the maximum number of detectors allowed, a minimum allowable radius size follows. Solving Equation 3.26 for r and substituting values from Equation 3.27, Equation 3.28 is derived.

$$r_{min} = \sqrt[n]{\frac{f \cdot n \cdot \Gamma(n/2)}{2 \cdot \eta_{max} \cdot \pi^{n/2}}} = \sqrt[n]{\frac{f \cdot n^2 \cdot P_s \cdot \Gamma(n/2)}{2 \cdot E_{max} \cdot \pi^{n/2}}} \tag{3.28}$$

Therefore, the minimum radius allowed due to memory constraints is a function of the number of dimensions n, coverage factor f, number of simultaneously processed test points P_s, and the maxim number of elements allowed in an array E_{max}.

3.5.2 Coverage factor limitations on maximum radius size. In addition to the problem of limited memory in higher-dimensional space, there is also a maximum constraint on radius size due to coverage factor in high-dimensions. To define these high-dimensional coverage effects, here we introduce five terms: *potential content*, *actual content*, *redundant content*, *effective content*, and *lost content*.

The potential content C_p of a detector set is the total sum of the content of all detectors in the set. The equation for potential content is shown in Equation 3.23 (it is repeated here in Equation 3.29). This value is termed ‘potential content’ because it is the amount of content that could be covered by the entire set of detectors if they were placed within an infinite shape space with no overlap.

$$C_p = C_D = \eta C_d \tag{3.29}$$

Actual content C_a is the total content covered by the detector set (i.e. content covered by more than one detector is only counted once). This value is termed ‘actual content’ because it does not including overlapping content, but only that content that is actually applied toward coverage. Contrarily, the redundant content C_r is the content contained in overlapping detectors. Thus, redundant content is calculated as the difference between the potential and actual content. The relationship between potential, actual, and redundant content is shown in Equation 3.30.

$$C_p = C_r + C_a \tag{3.30}$$

The effective content C_e is the total content covered by the detector set that lies within shape-space. Thus, effective content is that part of the actual content that falls within shape-space. This value is termed ‘effective content’ due to the fact that it is only the portion of the actual content that has an effect on anomaly detection. Lost content C_l is that portion of the total content covered by the detector set that is not a part of shape-space. This value is termed ‘lost content’ because it is that content that is using computing power, since some portion of each detector must fall within shape-space, but provides no value toward anomaly detection. The relationship between actual, effective, and lost content is shown in Equation 3.31.

$$C_a = C_e + C_l \tag{3.31}$$

Whereas the potential content is dependent only on the size of the detector; the actual, redundant, effective, and lost content are also dependent upon where the detector has been placed in shape space. In order to maximize the effectiveness of a real-valued negative selection system, it is important to maximize the effective content, while reducing the redundant and lost content. By reducing redundant and lost content, computing power and memory is not used on detectors that are providing no further benefit to the system. Additionally, maximizing the effective content ensures that the system is using as much of shape-space as possible to detect anomalies. The relationship between potential, redundant, effective, and lost content is derived in Equation 3.32, by substituting Equation 3.31 into Equation 3.30.

$$C_p = C_r + C_e + C_l \tag{3.32}$$

The naive real-valued selection system determines the number and size of detectors based on the *principle of potential content*, which is introduced here. A coverage factor of x means that the potential content of the detector set created contains x times the content of shape space. Since detection is dependent upon effective coverage, not potential coverage, it is important to ensure that lost content and redundant content are reduced (see Equation 3.32). Determining the redundant coverage C_r of a set of detectors has been investigated in the literature and shown to be a computationally difficult problem [42, 43]. Thus, this research considers the problem of reducing lost content. In order to reduce C_l, this research looks at two cases: the scenario where a detector falls in the extreme corner of shape space, and that where a detector falls in the center of shape space. These two locations provide bounds on lost content, since a detector placed at the center has minimal lost content, and a detector in the corner has maximal lost content.

In the case where a detector falls in the corner of shape-space, there is always content lost. The percent of content lost depends on the radius length and the number of dimensions. As the number of dimensions increases, the percent of content lost grows at a

rate greater than or equal to that shown in Equation 3.33, where the side-length of shape-space is 1 and n is the number of dimensions.

$$C_l = C_d - \frac{C_d}{2^n} \tag{3.33}$$

Content loss grows at that rate until the radius of a detector exceeds the side-length of shape space. After this point, it loses content at a greater rate. For example, two dimensional content loss is shown in Equation 3.34, where l is the side-length of shape-space, r is the radius length, and n is the number of dimensions. The first half of Equation 3.34 is found by inserting 2 as the value of n in Equation 3.33. The second half of equation 3.34 is derived by using the first half coupled with the derivation of the area of the content of a circular segment as shown in [18]

$$C_l = \begin{cases} C_d - \frac{C_d}{4}, & \text{for } 0 \le r \le l \\ C_d - \frac{C_d}{4} + \frac{r^2}{2} \cdot \left(\arcsin\left(\frac{\sqrt{r^2-l^2}}{r}\right) - \sin\left(\frac{\sqrt{r^2-l^2}}{r}\right)\right), & \text{for } l < r \le \sqrt{n} \end{cases} \tag{3.34}$$

This effect is demonstrated in Figures 3.6 and 3.7. Figure 3.6 shows the detector content lost from a two-dimensional hypersphere detector placed in the corner of shape-space as a function of radius length, while Figure 3.7 shows the percentage of detector content lost by the same detector. As additional dimensions beyond two are added, the effect of a

radius length exceeding the side-length of shape space becomes more pronounced. Therefore, a large percentage of content is always lost when a hyperspherical detector is placed in the corner of shape-space.

However, when a detector is placed in the center of shape-space there is a different effect seen. The detector has a $C_l = 0$ until the radius length is greater than half the side-length of shape-space. After this point, content is lost at increasing rates as radius length is increased, which is shown in Figure 3.8.

Since content loss is unavoidable, the proper maximum radius size based on content loss could be an area for further study. Stibor *et al* also cited observed effects in a study of

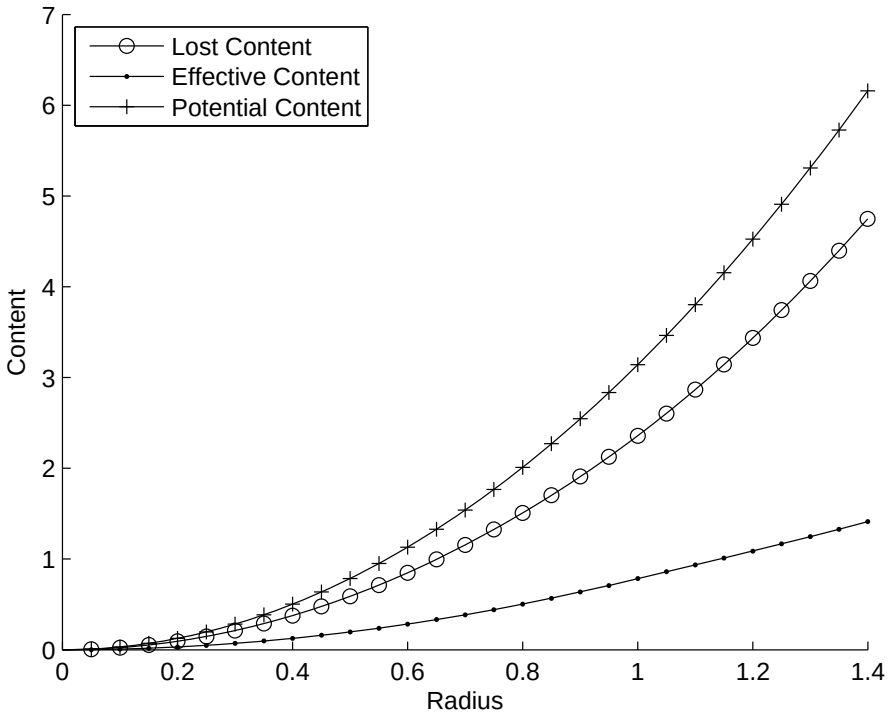


Figure 3.6: Detector content lost from a two-dimensional hypersphere detector placed in the corner of shape-space as a function of radius length.

V-detectors stemming from a disjoint between potential and effective content, which may be attributed to this phenomenon [44]. A maximum radius size of half of the side-length of shape-space is imposed in order to limit the effects of lost content on the experimental results of the detector shape comparison.

3.6 Detector shape comparison experimental design

With an understanding of high-dimensional considerations in place, an experiment is designed to compare the differences between detector shapes in high-dimensional spaces. First, the goal of the experiment is shown. Next, the pre-processing techniques used on the chosen dataset–KDD Cup ’99 10% dataset–is explained. Finally, the high-dimensional

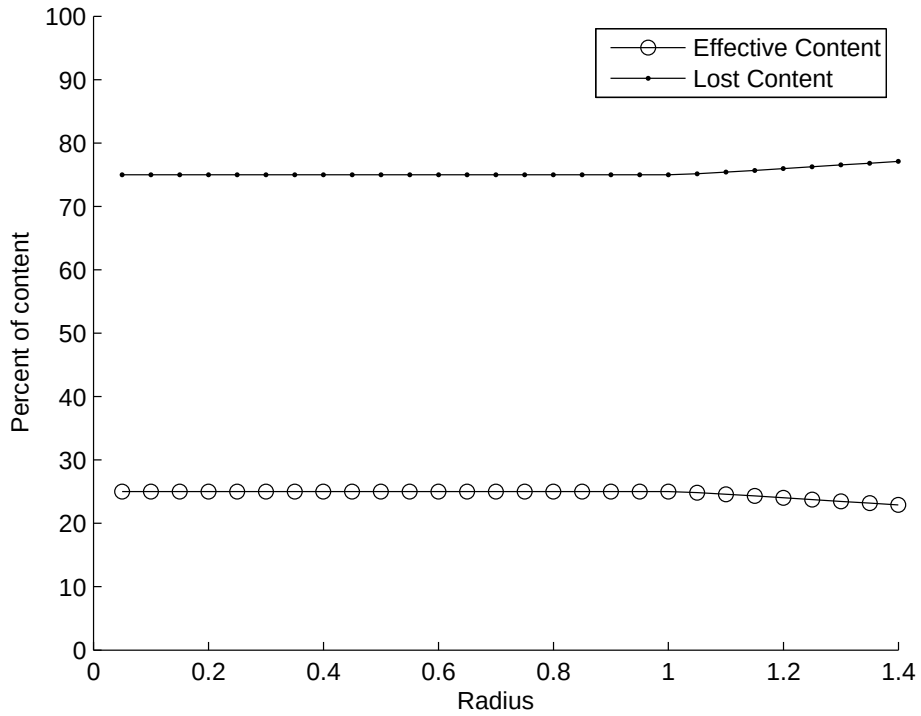


Figure 3.7: Percentage of detector content lost from a two-dimensional hypersphere detector placed in the corner of shape-space as a function of radius length.

detector sizing considerations explained in Section 3.5 is applied in order to find a bounding for detector radius sizes. This then leads to an experimental design.

3.6.1 Experimental question. Does the hypersteinmetz solid either provide better classification results or reduce variance within a real-valued negative selection system as dimensionality increases when compared to the hypersphere or hypercube?

3.6.2 Testable hypothesis. Holding the number of detectors created for a given negative selection run constant, the hypersteinmetz solid provides noticeably different results than the hypersphere, while showing somewhat different results than the hypercube as dimensionality increases. For the purposes of this experiment, the definition of *good*

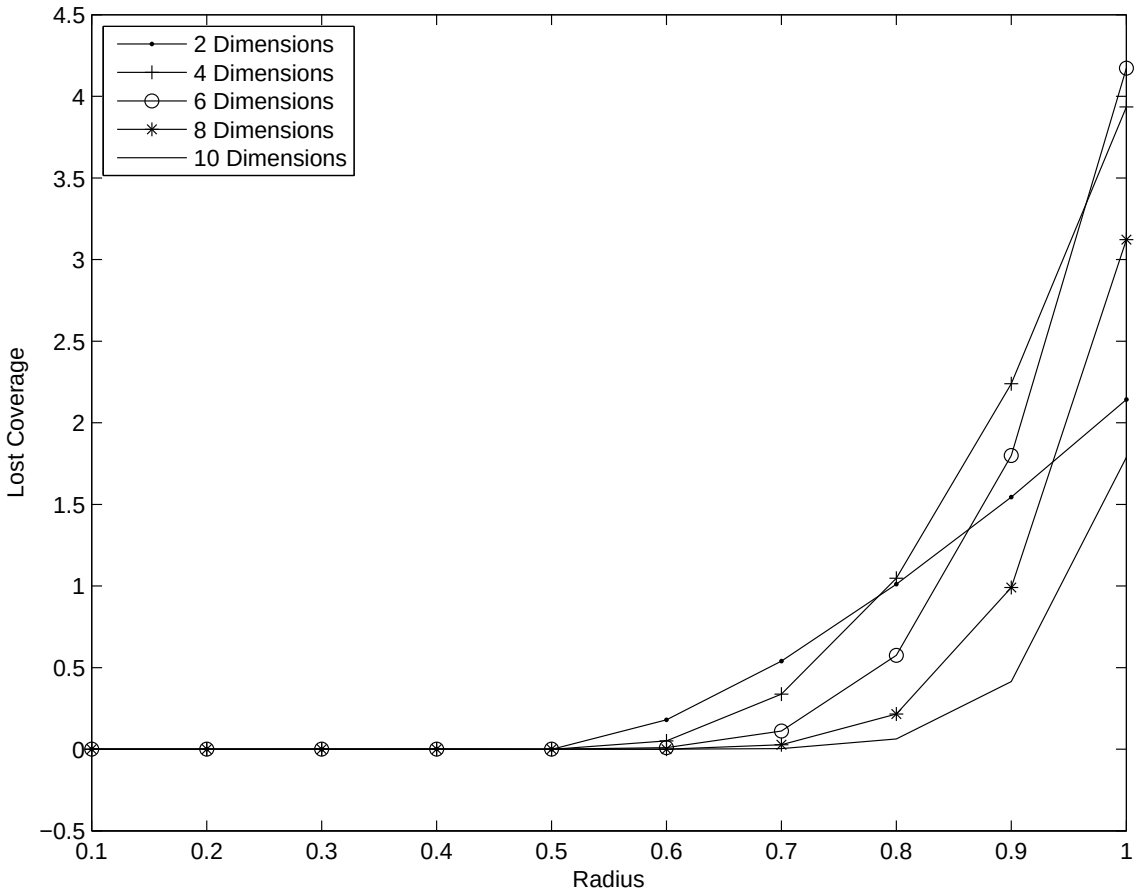


Figure 3.8: Detector content lost from a hypersphere detector placed in the center of shape-space as a function of radius length.

relies on the ROC curve for the detection results, comparing false positives to true positives. For a detector type to provide ”better” results, its classification results, when reported in ROC curve format, falls closer 0% false positive/100% true positive position than that of the other detectors; this implies that it can, overall, achieve higher true positive rates with lower false positives. It is expected that the hypersteinmetz and hypercube detectors outperforms the hypersphere, by garnering lower mean false positive percentages and higher mean true positive percentages. This effect becomes more pronounced as dimensionality increases.

Additionally, another definition of good for the purpose of this experiment is the variance that occurs between test runs. If there is a high variance from one run to the next, the classifications are not repeatable. However, if classification runs are consistently providing similar results, then they can be relied upon. For this reason, if a shape provides lower variance–all else remaining equal–than another, it can be considered better in that regard. It is expected that the hypercube and hypersteinmetz provides classification results with lower variance than that of the hypersphere.

3.6.3 KDD dataset pre-processing. The KDD Cup ‘99 dataset was chosen for two reasons. First, it provides an example of a network intrusion dataset with many features. Secondly, it has been used by many previous researchers, and therefore provides a useful dataset for comparison purposes. Here, we have used the KDD Cup ’99 10% dataset, rather than the entirety of the dataset. In the original, there are hundreds of thousands of data points, and as such an exhaustive investigation of the dataset is not computationally feasible with accessible resources. Therefore, the 10% dataset provides a subset of the points that has been used by previous researchers [56]. The dataset is set up in such a way that all different attack types are fused to form an attack class. This provides for a two-class classification method, and has been shown previously [5] [35].

In order to use this dataset, however, some pre-processing must first take place. The KDD Cup ’99 originally contains 41 dimensions of data. However, seven features of the data are non-real-valued features. The use of the non-real-valued features is an area of possible future research, but for the current experiment these dimensions are excluded. Additionally, feature 20 (number of outbound commands) is removed, as every entry in this field is the same. Thus, the dataset is pared down to 33 dimensions. Similar methods have been used previously [56].

Three methods were considered for dimensional ordering. The dimension orderings found in both [26] and [2] were investigated. The method employed in [26] was to rank

the features from most to least discriminating using information gain, by calculating how well each feature discriminates a specific class. The method for ranking features proposed by [2] incorporates the information gain technique along with K-means learning. Classification results were extremely poor using these two methods. The reason for this is most likely that both methods were developed in order to determine those features that would distinguish specific types of attacks, rather than doing a binary–attack versus normal–classification.

The dimensions were therefore reordered according to the process described in Algorithm 6, where P_{self} and $P_{nonself}$ represent the sets of self and nonself points, p and q are points within the set of all points, n is the number of dimensions, and u_i is the number of unique nonself points for a given dimension i. The reason the data are arranged in this manner is that the data are presented to the system in such a way that those features most likely to show difference between self and nonself points are processed when fewer dimensions are used. This choice is made, understanding that the objective of the negative selection algorithm is to train only on self data. However, the objective of this experiment is not to show the validity of the training process, but rather the effectiveness of the detector shapes.

From a training perspective, there is no added value in training on duplicate data points. Previous studies using the KDD data, such as [46], have also recognized duplicate data as a problem causing the failure of methods using the dataset. In order to eliminate duplicate points, the dataset is first pared down to only include the features used in the current iteration. Once this is done, the data is divided into normal and attack traffic sets. All duplicates are removed from each individual set. However, for testing purposes, duplicate points may still exist between the normal and attack traffic. This process is demonstrated in Algorithm 7.

Algorithm 6 Pseudocode of algorithm to reorder features in KDD dataset

Divide data into self P_{self} and nonself $P_{nonself}$ points

for $i = 1 \rightarrow n$ **do**

  for all points $p \in P_{self}$ **do**

    Remove all points p with duplicate values in dimension i

  end for

  for all points $q \in P_{nonself}$ **do**

    Remove all points q with duplicate values in dimension i

  end for

  for all points $p \in P_{self}$ **do**

    for all points $q \in P_{nonself}$ **do**

      if $p = q$ in dimension i **then**

        Remove q from the set of nonself points

      end if

    end for

  end for

  Record the number of remaining unique nonself points u_i for dimension i

end for

Reorder the dimensions according to u_i value, from highest to lowest

Algorithm 7 Pseudocode of algorithm to remove all duplicate data points

Remove all features not used in this run

Divide data into self P_{self} and nonself $P_{nonself}$ points

for all points $p \in P_{self}$ **do**

Remove all points p with duplicate values in all dimensions

end for

for all points $q \in P_{nonself}$ **do**

Remove all points$q$ with duplicate values in all dimensions

end for

Once the pre-processing is complete, training and testing datasets must be selected from the data. A random sampling method, similar to that of [56] is used. As the system must be trained only on self data, only that network traffic considered normal is included in the training set. 5,000 data points are randomly selected from the entirety of the set of points classified as normal. Additionally a test dataset is selected; this dataset contains 500 points selected in the same manner. However, these points include not only normal data, but also data points from the attack traffic (of which there are four types). Attack and normal traffic points are selected at the same ratio as they exist in the larger dataset, in order to form the smaller test dataset. The numbers of data points chosen for the experiment were specifically chosen due to time constraints for completion of the experiments. More or fewer data points could be chosen in future experiments.

3.6.4 KDD Dataset radius sizing constraints. Detector radius size is a parameter that must be set in the shape experiment. The methods for minimum and maximum sizing laid out previously are used to provide a range of testable radii. For every iteration of the experiment, new r_{min} and r_{max} values are determined. Unless memory constraints are the

constraining factor, the default method for sizing is the nearest neighbor method for minimum sizing and the largest detector placement method for maximum sizing.

3.6.5 Outline of experiment. This experiment is a full factorial experiment, testing all three variables: radius size, coverage factor, and detector shape. There are a series of three different shapes used (hypersphere, hypersteinmetz, and hypercube). For the purposes of this discussion, the radius of a hypersphere is the distance from the center of the hypersphere to any point on the surface, the radius of a hypersteinmetz is the minimum distance from the center of the hypersteinmetz to any point on the surface, and the term radius is used in conjunction with hypercubes to mean half of the side-length of the given hypercube. Each detector shape is tested 10 times, using varying coverage factors, numbers of dimensions, and radius sizes ($r_{min} \rightarrow r_{max}$, by steps of $\frac{r_{max}-r_{min}}{2}$). The results are recorded as true positives, false positives, true negatives, and false negatives. See Algorithm 8 for pseudo-code representation of this outline and Table 3.4 for a tabular representation of the variables being tested.

Choosing the dimensional pairs for each of the $\left\lceil \frac{n}{2} \right\rceil$ cylinders within a hypersteinmetz, it is important to ensure that each dimension is chosen at least once, ensuring that the detectors are bounded in each dimension. If computational complexity were not an issue, it would be logical to choose a random dimensional pairing for each cylinder within each detector. This would provide an even distribution of possible detector "orientations" throughout the space. However, in order to keep track of which detectors have which set of pairings, a matrix of detector dimensional pairings must be kept. This doubles the space and time complexity needed. For this reason, all detectors are kept at the same pairing. An area for future research, therefore, is to determine which is the optimal pairing of dimensions for the $\left\lceil \frac{n}{2} \right\rceil$ cylinders within a given dataset. Since an optimal pairing of

Algorithm 8 Pseudocode design of detector shape comparison experiment. r is the radius of the detector, f is the coverage factor, η is the number of detectors, n is the number of dimensions.

for each detector shape: Hypersphere, Hypersteinmetz, Hypercube **do**
  for $n = 2, 8, 16$ **do**
    Determine r_{min} and r_{max} using methods described in Section 3.6.4
    for $r = r_{min} \rightarrow r_{max}$, by steps of $\frac{r_{max} - r_{min}}{2}$ **do**
      for $f = 2, 8, 16$ **do**
        repeat
          Set η according the number of detectors used for a hypersphere using equation 3.26, and use the same η for equivalent hypersteinmetz and hypercube
          Execute Algorithm 1 using η detectors and record the true positives, true negatives, false positives, and false negatives
        until 10 tests have been run on each subset
      end for
    end for
  end for
end for

dimensions is not an objective of this research, the pairs are determined by order within the dataset, with the last dimension being paired with itself.

3.6.6 Representation of results. Results are shown in both graphical and tabular form. A series of graphs shows a set of ROC curves comparing false positives against true positives. The series shows the results of runs using different numbers of dimensions, demonstrating the difference between the detection results of the different shapes as

Table 3.4: Design parameters for detector shape comparison experiment

Parameter	Values
Dataset	KDD Cup '99 10% Dataset
Detector Shape	Hypersphere, Hypersteinmetz, Hypercube
Number of Dimensions n	2, 5, 8, 11
Detector Radius r	$r_{min} \rightarrow r_{max}$, by steps of $\frac{r_{max}-r_{min}}{2}$
Shape-Space Content C_{ss}	1
Coverage Factor f	2, 8, 16
Number of Training Points P_r	5000
Number of Test Points P_s	500
10 iterations are performed for each test. Measured outputs for each test are true positives, true negatives, false positives, and false negatives.	

dimensions are increased. Additionally, a second series of graphs demonstrates how each individual shape progresses as dimensions are added, with a set of three ROC curves–one for each shape. Then a set of tables shows the results displayed in the ROC curve, along with the standard deviations of the results.

3.7 Summary of experiments

All of the experiments contained in this chapter are summarized in Table 3.5.

Table 3.5: Design parameters for detector shape comparison experiment

Experiment	Variables	Experimental Question
Detector radius sizing	Dataset, Detector radius r	Do the nearest neighbor method for sizing the minimum radius length r_{min} and the largest detector placement method for sizing the maximum radius length r_{max} provide a good lower and upper bound on the radii to test for the RNS system?
Coverage factor	Dataset, Detector radius r, Coverage factor f	How do we approximate the relationship between coverage factor and classification accuracy?
Detector shape comparison	Detector shape, Number of dimensions n, Detector radius r, Coverage factor f	Does the hypersteinmetz solid either provide better classification results or reduce variance within a RNS system as dimensionality increases when compared to the hypersphere or hypercube?

4 Results and Analysis

This chapter present the results of the experiments outlined in Chapter 3. Upon stating the significant results of each experiment, these results are then analyzed. The causes of the results are suggested and the significance of each result is discussed.

First, the detector radius sizing experiment results are discussed. Then the results of the coverage factor experiment are presented. Finally, we present the results of the detector shape comparison experiment.

4.1 Detector radius sizing experiment results

The detector sizing experiments described in Section 3.3.3 were performed on a Dell Precision M6500 with an Intel i7 920 processor. This processor has a clock speed of 2.67 Gigahertz (GHz) = $2.67 \cdot 10^9$ cycles/second. The following number of runs were needed: 8 datasets $\times$ 20 radius size thresholds per dataset $\times$ 10 repetitions per dataset $\approx$ 1600 runs. Each run for these thresholds took on average approximately 1.5 minutes. Therefore, the tests took approximately: 1600 runs $\times$ 1.5 minutes/run = 2400 minutes = 30 hours.

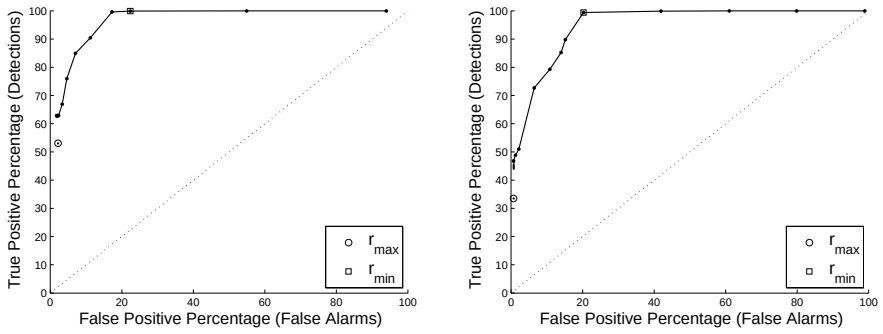
Figures 4.1 and 4.2 show the receiver operating characteristic curves of the results of the experiment. Each data point represents the mean true positive percentage and mean false positive percentage value attained in ten classification runs with a given radius. The circled data points represent the results obtained using the r_{max} value found using the largest detector placement method for the given dataset. The squared data points represent the results obtained using the r_{min} value found using the nearest neighbor method for the given dataset. Tables 4.1 and 4.2 summarize the results shown in the figures, showing the true and false positive percentages along with the standard deviation of each percentage. These results are reported only for those runs completed using the r_{min}, r_{max}, and best radii. The results marked ‘n/a’ in Table 4.1 are due to the fact that the minimum radius size was

not tested, due to the r_{min} and r_{max} falling between the 0.01 radius size step of the experiment. Complete results from this experiment are shown in Appendix A.

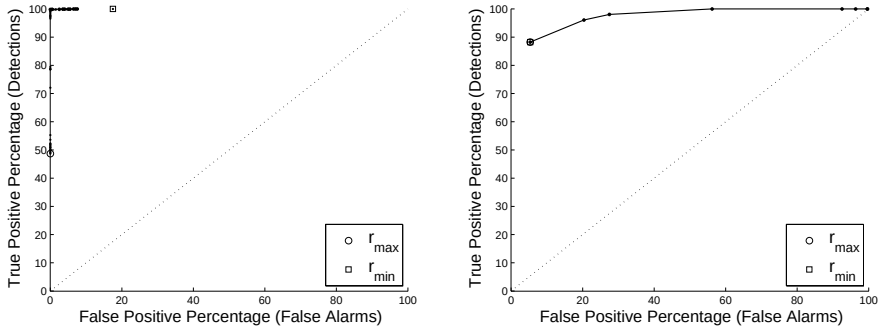
For the purposes of this experiment, a false negative percentage and a false positive percentage is considered to carry equal weight. For example, increasing false negative percentage by 1% is considered equally as undesirable as increasing the false positive percentage by 1%. The ideal point on the ROC curve would be at 0% false positives and 100% true positives. For comparison, given this assumption, the Manhattan distance between the ideal 0%/100% point and a given radius size point provides the level of 'goodness' of the radius size point. If point *a* has a smaller Manhattan distance to the ideal point than point *b*, then *a*'s false positive percentage plus false negative percentage is less than that of *b*. The *best* radius for a given dataset is the one that produces mean classification results with the smallest Manhattan distance to the ideal point.

Table 4.1: Condensed results of detector size range experiments for the first four datasets comparing results obtained using r_{min}, r_{max}, and *best* radius sizes. Dist to 0/100 represents the Manhattan distance of the point to the 0% false positive/100% true positive point.

Dataset	Comb			Comb Neg			Int Thick			Int Thick Neg		
Point	r_{min}	*best*	r_{max}	r_{min}	*best*	r_{max}	r_{min}	*best*	r_{max}	r_{min}	*best*	r_{max}
Radius	.032	.040	.207	.041	.041	.160	.009	.130	.353	N/A	.050	.050
Dist to 0/100	22.4	17.6	49.1	20.8	20.8	67.2	17.5	0.2	51.3	N/A	17.1	17.1
True Pos %	99.9	99.6	53.1	99.4	99.4	33.5	100.0	99.8	48.7	N/A	88.2	88.2
TP% Std Dev	0.09	0.14	0.14	0.12	0.12	0.16	0.03	0.08	0.18	N/A	0.00	0.00
False Pos %	22.4	17.2	2.2	20.2	20.2	0.7	17.5	0.0	0.0	N/A	5.3	5.3
FP% Std Dev	0.87	0.50	0.00	0.70	0.70	0.00	3.54	0.00	0.00	N/A	0.05	0.05



(a) Comb (b) Comb Negative

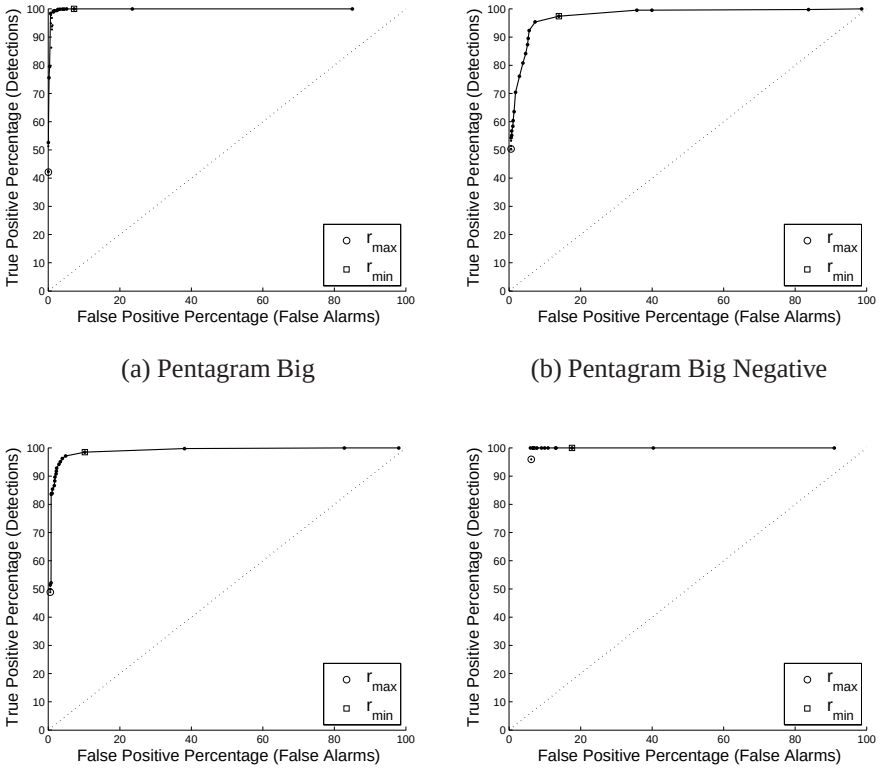


(c) Intersection Thick (d) Intersection Thick Negative

Figure 4.1: Receiver operating characteristic curves for datasets (a) Comb, (b) Comb Negative, (c) Intersection Thick, and (d) Intersection Thick Negative

Table 4.2: Condensed results of detector size range experiments for the second four datasets comparing results obtained using r_{min}, r_{max}, and *best* radius sizes.

Dataset	Pent Big			Pent Big Neg			Ring Thick			Ring Thick Neg		
Point	r_{min}	*best*	r_{max}	r_{min}	*best*	r_{max}	r_{min}	*best*	r_{max}	r_{min}	*best*	r_{max}
Radius	.025	.130	.340	.041	.050	.193	.040	.060	.200	.030	.140	.172
Dist to 0/100	23.5	2.2	57.9	16.5	11.9	50.2	11.7	7.7	51.7	17.6	6.0	10.2
True Pos %	100.0	99.2	42.2	97.4	95.4	50.3	98.5	96.2	48.9	100.0	100.0	95.9
TP% Std Dev	0.00	0.15	0.06	0.15	0.15	106.14	0.00	0.11	0.10	0.00	0.06	0.18
False Pos %	7.2	1.4	0.0	13.9	7.3	0.5	10.2	3.9	0.6	17.6	6.0	6.2
FP% Std Dev	0.55	0.59	0.00	0.80	0.21	1.11	0.99	0.12	0.00	0.49	0.91	0.28



(a) Pentagram Big

(b) Pentagram Big Negative

(c) Ring Thick

(d) Ring Thick Negative

Figure 4.2: Receiver operating characteristic curves for datasets (a) Pentagram Big, (b) Pentagram Big Negative, (c) Ring Thick, and (d) Ring Thick Negative

The trend that emerges from these results is that the *best* radius size consistently falls between the minimum and maximum radii. Thus, the r_{min} and r_{max} values provide a reasonable constraint on radius size. The one dataset where the best point does not fall between the two extremes, Intersection Thick Negative (Figure 4.1d), is because of the fact that the minimum and maximum values were closer to each other than were the increments used between radii for the experiment. Therefore, the minimum radius size was not actually tested. This experiment confirms that the nearest neighbor and largest detector placement methods are viable constraints for placing a detector radius range in two-dimensional datasets, given the assumptions of the experiment: two dimensions, hypersphere detector, and the specified datasets. Therefore, we use this method to bound detector radius size in future experiments.

4.2 Coverage factor experiment results

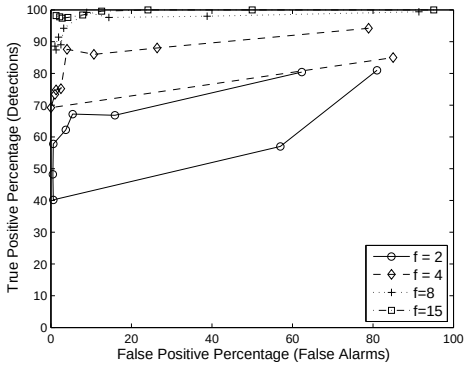
The coverage factor experiment described in Section 3.4 was run on a computer with 2 Intel Xeon processors X5472, 3.00 GHz, and 32.0 Gb of random access memory (RAM). The experiment required the following number of runs: 1 detector shape * 1 dimension size * 10 radius sizes * 11 coverage factors * 30 datasets * 10 runs/dataset = 33,000 runs. An average runtime of 5 seconds created a total runtime of 2,750 minutes (2 days). This was further reduced by running multiple processes, and completion took approximately 22 hours.

Figure 4.3 contains the ROC curves obtained from the classification results using the three different types of Iris as nonself. Each line represents all results for one coverage factor, with the points on the line representing different detector radii. Points are connected in the order of smallest radius to largest radius, with the smallest usually falling in the upper-right corner and each consecutive point representing a larger radius until the final r_{max} point is reached for the given coverage factor. An individual point represents the mean classification values for a set of ten runs. Table 4.3 summarizes the results shown in

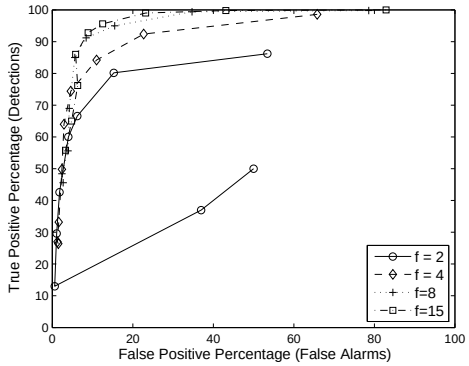
figure 4.3 for the *best* radius for each coverage factor. Complete results are shown in Appendix B.

The same definition of a *best* radius is used as that for the previous experiment. Those points on the ROC curve that provide results with shortest Manhattan distance to the 0% false positive 100% true positive point are considered to be the best. In that regard, it can be seen from Figure 4.3 and Table 4.3 that as the coverage factor is increased, the ROC curves move closer to the 0% false positive 100% true positive point. However, the aggregate gain in classification accuracy from each successive coverage factor becomes less. For example, in Figure 4.3a a coverage factor of four provides significantly better results than a coverage factor of two, but a coverage factor of 15 barely provides better results than that of coverage factor eight. Specifically, Table 4.3 shows that the best point produced using coverage factor of two with the Setosa dataset provides a true positive percentage of 67.2% and a false positive percentage of 5.4%, a coverage factor of four provides a true positive percentage of 87.6% and a false positive percentage of 4.0%, while a coverage factor of eight provides only a true positive percentage of 94.2% and a false positive percentage of 3.2%. Additionally, Table 4.3 shows that each iris type reaches a

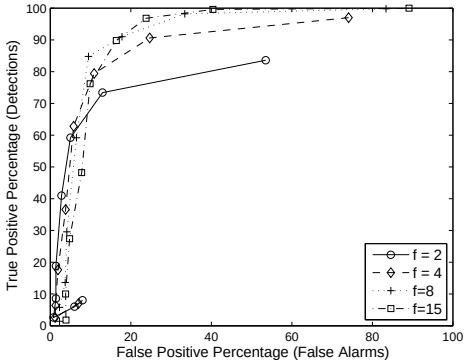
point at which the best radius size for the coverage factor is the same for all successive coverage factors. Setosa reaches this point at a coverage factor of 14, Versicolor reaches it at $f = 12$, and Virginica achieves it at $f = 8$. The reason for this, is that there is that as coverage factor is increased, the area of nonself space covered by detectors changes less by the added detectors. As such, the results remain more closely aligned.



(a) Setosa as nonself



(b) Versicolor as nonself



(c) Virginica as nonself

Figure 4.3: Receiver operating characteristic curves as coverage factor increases for increasing detector sizes for Iris datasets.

Table 4.3: Condensed results of coverage factor experiments for *best* radius for each coverage factor.

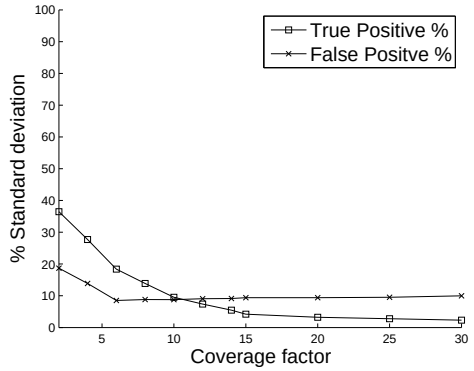
Setosa											
Cov Factor	2	4	6	8	10	12	14	15	20	25	30
Radius	.265	.357	.726	.449	.818	.726	.818	.818	.818	.818	.818
True Pos %	67.2	87.6	89.8	94.2	94.6	97.0	97.2	98.2	98.6	99.0	99.2
False Pos %	5.4	4.0	1.4	3.2	1.4	2.0	1.6	1.3	2.0	2.6	3.1
Versicolor											
Cov Factor	2	4	6	8	10	12	14	15	20	25	30
Radius	.164	.254	.343	.343	.343	.433	.433	.433	.433	.433	.433
True Pos %	80.2	84.2	85.2	91.2	92.6	93.8	92.0	92.8	94.8	97.4	98.4
False Pos %	15.3	11.0	6.5	8.4	10.2	8.3	9.0	8.9	9.6	13.2	12.3
Virginica											
Cov Factor	2	4	6	8	10	12	14	15	20	25	30
Radius	.159	.250	.250	.342	.342	.342	.342	.342	.342	.342	.342
True Pos %	73.4	79.4	85.2	84.8	87.6	86.8	90.4	89.8	94.4	95.6	95.0
False Pos %	13.0	10.9	13.7	9.5	10.5	12.8	15.2	16.4	16.5	19.7	21.5

Another aspect that increasing coverage factor provides is that of reduced variance. Figure 4.4 reports the mean standard deviation of the percentage of false positive and true positives over all detector radii for a given coverage factor. Each point represents the mean percentage obtained by all runs completed with a given coverage factor. Table 4.4 shows the complete results shown in Figure 4.4 in tabular form.

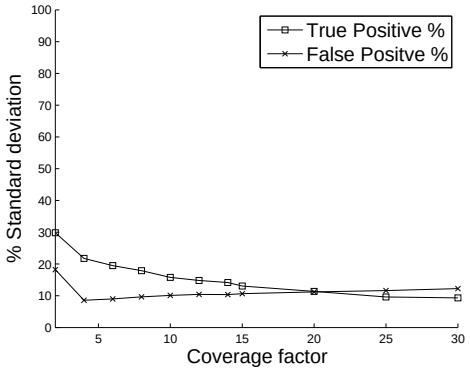
As can be seen in Figure 4.4 and Table 4.4, as the coverage factor increases, the standard deviation of the true positive and false positive percentages approaches a minimum value. This minimum value comes as a result of the fact that by using more detectors, there is less likelihood of change in the space covered by detectors between two consecutive runs of the same algorithm. It can also be seen from Table 4.4 that there are fewer benefits the higher the coverage factor increases.

For example, in the Table 4.4 Setosa section there is a large benefit in moving from a coverage factor of two up to a coverage factor of 10, with true positive percentage standard deviation going from 36.5% down to 9.5% and false positive percentage standard deviation reducing from 18.7% to 8.8%. However, there is little to no benefit once the coverage factor moves from 10 up to 30, with true positive percentage standard deviation reducing from 9.5% to 2.3% while the false positive percentage standard deviation actually increases from 8.8% to 10.0%. The other datasets show a similar effect.

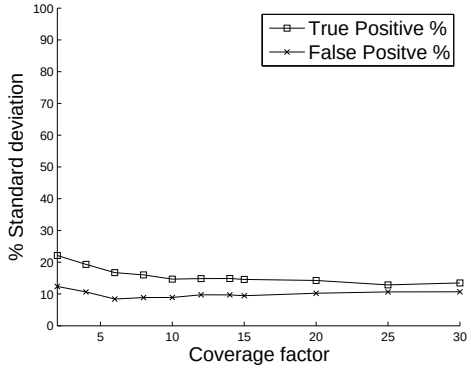
The significance of this finding is that we can use these results to potentially set a coverage factor range for future experiments. A low coverage factor provides more variability in the results and a higher coverage factor provides less. It also gives a potential bound on the highest coverage factor needed to provide a given level of variance. However, it is important to note that the specific effect of coverage factor is determined by the dataset in question.



(a) Setosa



(b) Versicolor



(c) Virginica

Figure 4.4: Mean percent standard deviation of the false positives and true positives as the coverage factor is increased on the iris datasets.

Table 4.4: Mean percent standard deviations of the true and false positive percentages on the iris dataset.

Setosa											
Cov Factor	2	4	6	8	10	12	14	15	20	25	30
TP% Std Dev	36.5	27.7	18.4	13.9	9.5	7.4	5.5	4.2	3.2	2.8	2.3
FP% Std Dev	18.7	13.9	8.5	8.8	8.8	9.1	9.1	9.4	9.4	9.5	10.0
Versicolor											
Cov Factor	2	4	6	8	10	12	14	15	20	25	30
TP% Std Dev	29.8	21.8	19.5	17.9	15.8	14.8	14.2	13.1	11.4	9.6	9.3
FP% Std Dev	18.2	8.6	9.0	9.7	10.1	10.4	10.4	10.6	11.2	11.6	12.3
Virginica											
Cov Factor	2	4	6	8	10	12	14	15	20	25	30
TP% Std Dev	22.1	19.3	16.8	16.0	14.7	14.8	14.9	14.6	14.3	12.9	13.5
FP% Std Dev	12.4	10.7	8.4	8.9	8.9	9.8	9.7	9.4	10.3	10.7	10.7

4.3 Detector shape comparison experiment results

The following section is divided into two sub-sections. First, radius sizing constraints specifically applied to the KDD Cup ’99 dataset are addressed. Then, the results of the experiments using these radius sizing constraints are discussed.

4.3.1 Minimum radius sizing. Both the nearest-neighbor and memory limitations constrain minimum detector size. Using the nearest neighbor method described in section 3.3.1, the minimum radius constraints for the KDD Cup ’99 dataset were found. Figure 4.5 shows the r_{min} values found using the nearest neighbor method for increasing numbers of dimensions, it shows that as dimensions are added, the radius of the nearest neighbor distance increases.

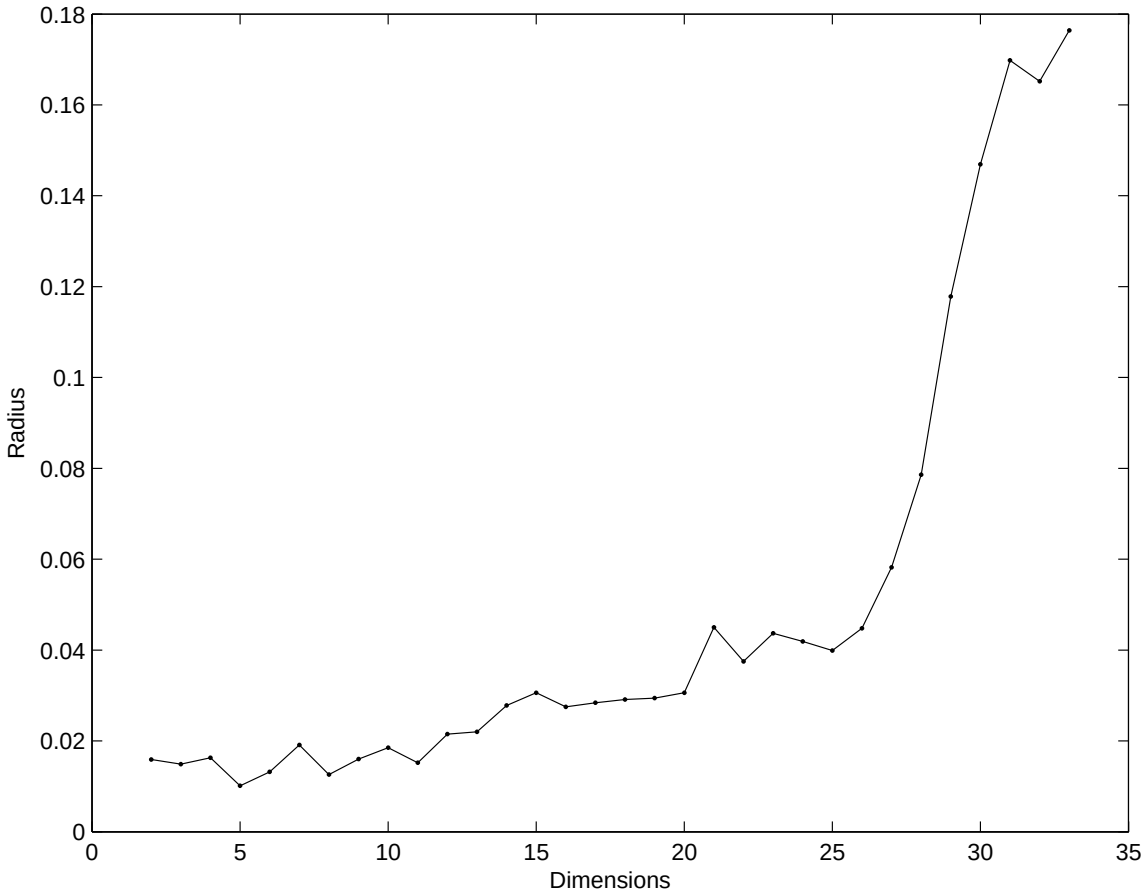


Figure 4.5: Minimum radius constraint using nearest neighbor (NN) method for KDD Cup ’99 10% Dataset when using 500 test points.

The minimum radius size constraint is calculated using the memory constraint method described in section 3.5.1. With the limitations of the computer used (AMD Athlon II X2 215 2.70 GHz Processor, Windows 7 64-Bit operating system, 4.00 Gb RAM), the maximum usable array contains 250,000,000 elements. These memory limitations result in a maximum number of detectors. Figure 4.6 shows the maximum number of detectors allowed with the given constraints as a function of the number of dimensions. Three numbers of simultaneous test points are compared. The number of allowable detectors decreases quickly as dimensionality increases. It is significant that the

number of detectors allowed decreases as dimensions are added, because we have shown previously that if a detector has a content ratio less than one, the size of a detector is already decreasing. Therefore, there is not only loss of detection capability due to number of detectors, but also due to the content ratio of those detectors.

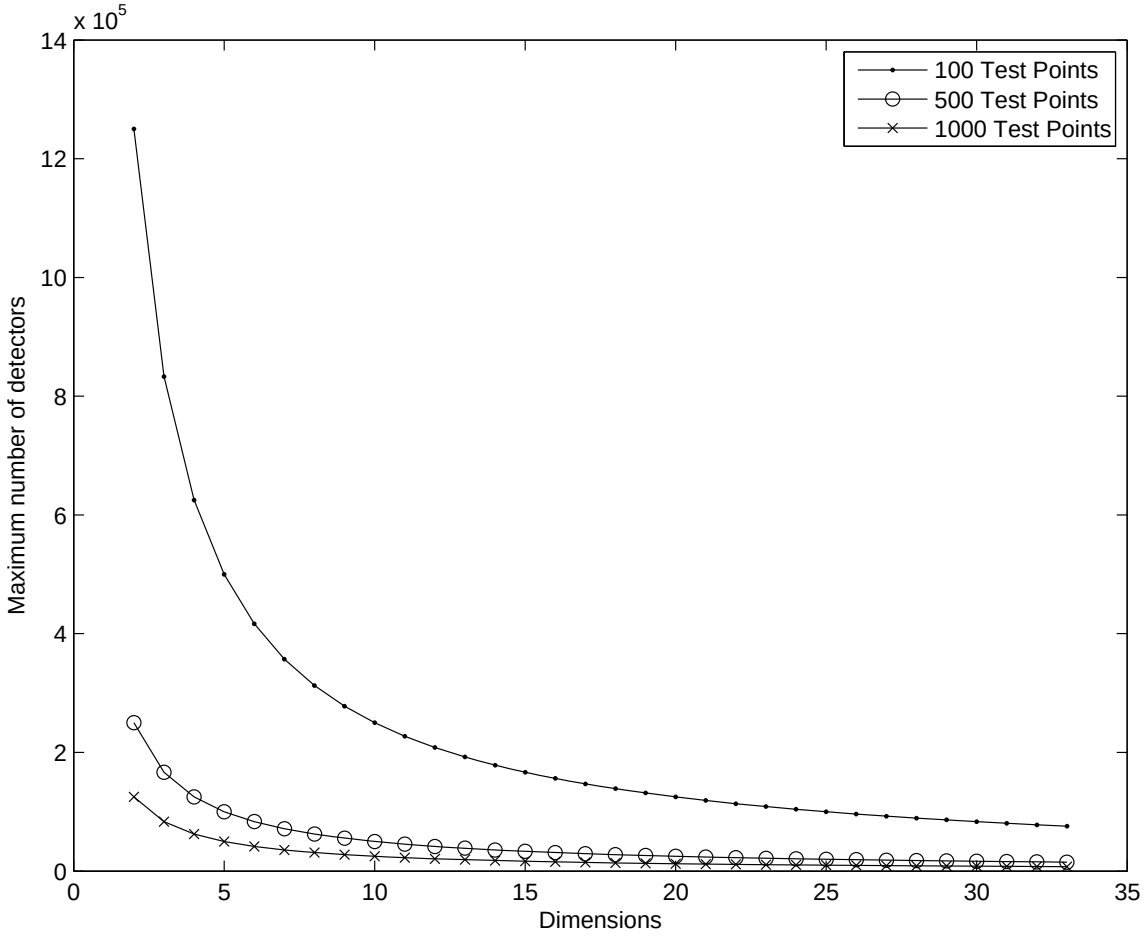


Figure 4.6: Maximum number of detectors allowed as a function of the number of dimensions, using a 250,000,000 element array limit.

Figure 4.7 compares constraints on radius size for the memory and nearest-neighbor methods. Each point in the figure represents the minimum allowable radius for the given number of dimensions when using one of the two methods, with those using the memory

constraint method divided among different coverage factors. These results use the same 250,000,000 element limit, with 500 simultaneous test points. Figure 4.7 shows that the memory constraint quickly dominates the nearest neighbor method for minimum radius size. In this specific instance, all reported memory constrained minima pass those of the nearest neigbor method after two dimensions.

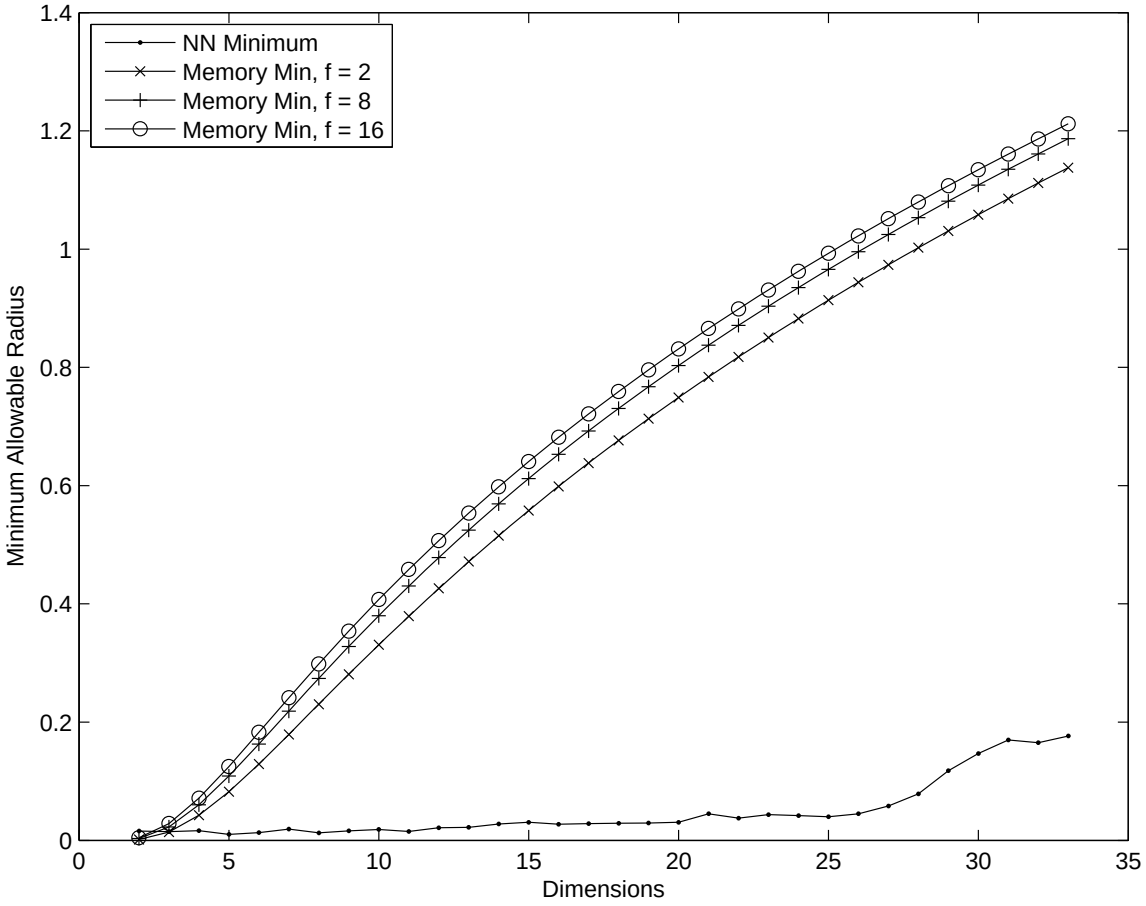


Figure 4.7: Minimum allowable radius size due to memory constraints in comparison to minimum radius size found using the nearest neighbor method.

Using the nearest neighbor method and the memory constraint method together, the dominant constraint is chosen. For the two-dimensional case, the nearest neighbor method

is the dominant constraint. For all other cases, the memory constraint method is the dominant constraint.

4.3.2 Maximum radius sizing. A similar analysis of the maximum radius sizing techniques is also undertaken to determine the dominant constraint. The first method presented in Section 3.3.2, uses the largest detector that can be placed within the dataset to determine the maximum radius size. Using this method, the maximum detector size limits shown in Figure 4.8 are obtained. By next imposing the second limit of a maximum detector radius of half the side-length of shape-space, Figure 4.8 demonstrates how the constraining factor changes as dimensions are added. Specifically, in Figure 4.8, each point represents the maximum allowable radius size as a function of the number of dimensions. The maximum allowable radius size method results are shown for all three shapes (hypersphere, hypersteinmetz, and hypercube), represented by three different trend lines. This method is then contrasted with a trend line representing the maximum radius size determined by the half side-length of shape-space.

Similar to the minimum radius, the maximum radius constraint changes after two dimensions. Therefore, the maximum radius size for two dimensions is determined by the maximum detector placement method and the maximum radius for all dimensions greater than three is set at .5, half the side-length of shape-space. An additional constraint that the half side-length of shape-space constraint places on the experiment is that no more than 11 dimensions can be used before the minimum values found via memory constraints exceed the half side-length maximum value. Therefore, the detector shape comparison experiment was constrained to use between two and eleven dimensions.

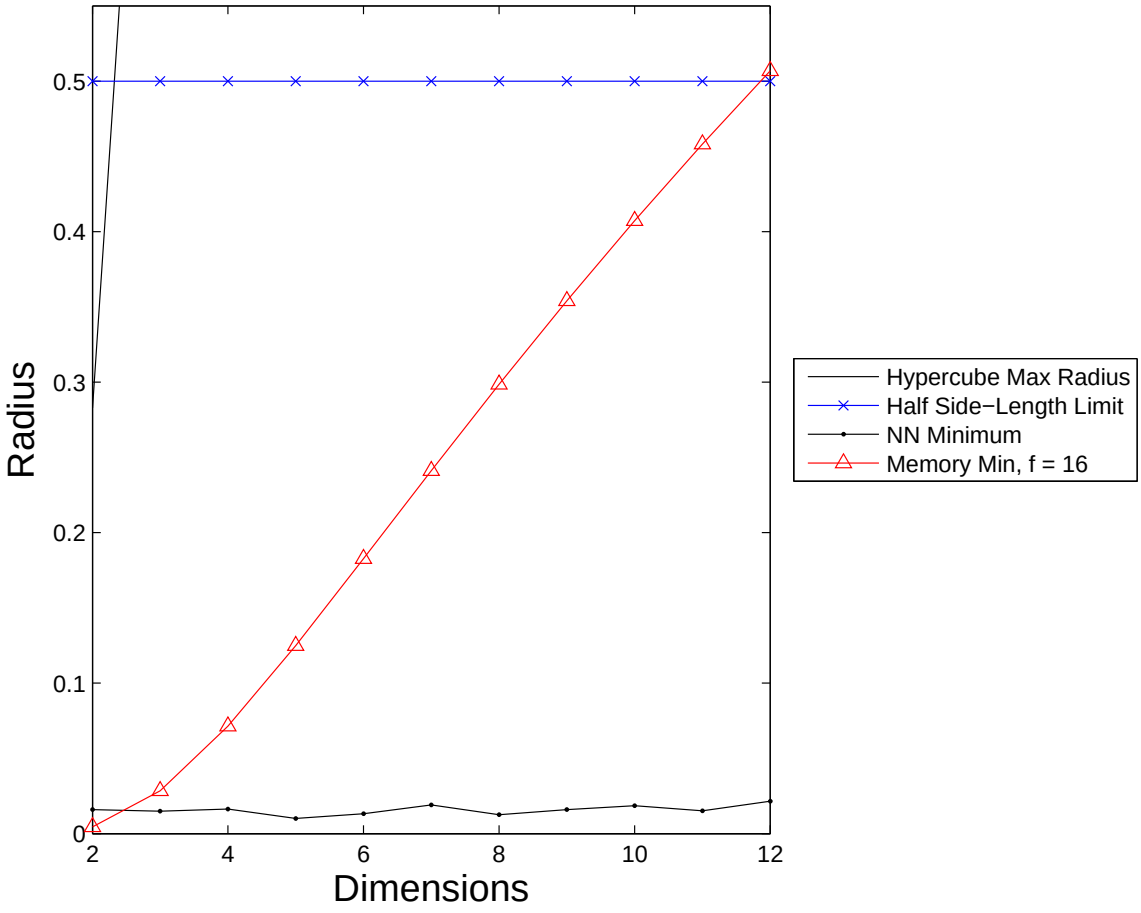
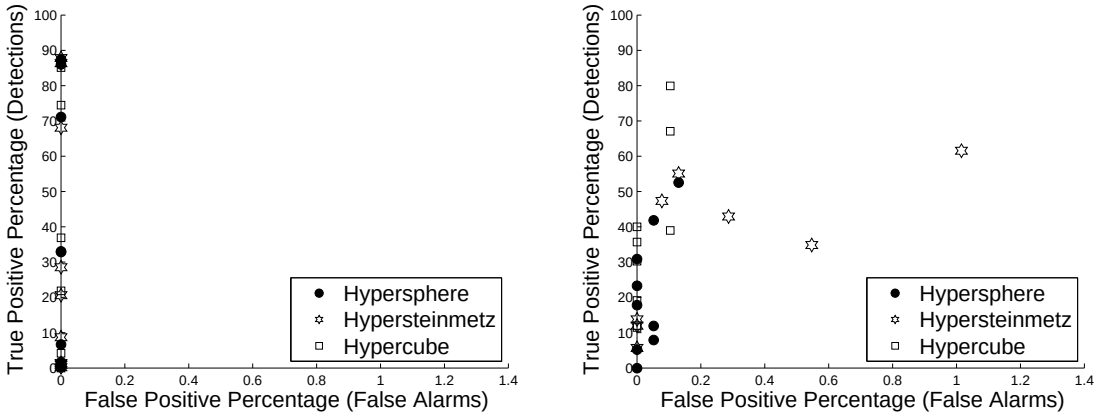


Figure 4.8: Maximum allowable radius size comparing maximum detector placement and half side-length of shape-space methods.

4.3.3 Experiment results. The detector shape comparison experiment described in Section 3.6 was run on a computer with 2 Intel Xeon processors X5472, 3.00 GHz, and 32.0 Gb RAM. The following number of runs were performed: 3 detector shapes * 4 numbers of dimensions * 3 radius sizes * 3 coverage factors * 1 dataset * 10 run/dataset = 1080 runs. An average run took approximately 40 seconds to complete, and the total runtime was approximately 12 hours.

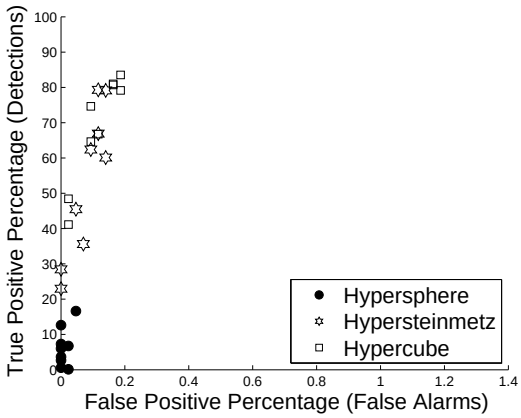
Figure 4.9 shows the receiver operating characteristic curves obtained from the runs of the detector shape comparison experiment, showing the mean percentage of false

positives versus the mean percentage of true positives over each set of 10 runs. Each sub-figure represents a set number of dimensions (2, 5, 8, and 11), and contains a comparison of the three shapes (hypersphere, hypersteinmetz, and hypercube). Complete tables of all results are given in Appendix C.

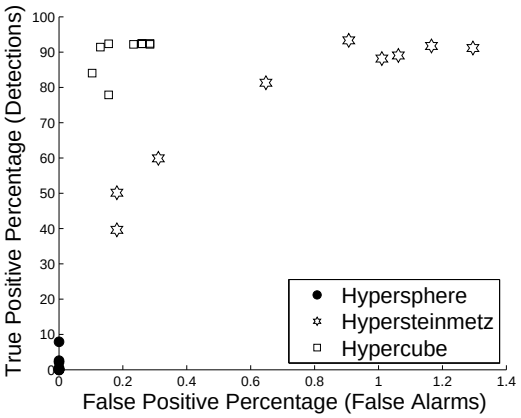


(a) 2 Dimensions

(b) 5 Dimensions



(c) 8 Dimensions



(d) 11 Dimensions

Figure 4.9: Receiver operating characteristic curves for KDD Dataset comparing different detector shapes in 2, 5, 8, and 11 dimensions. NOTE: Tthe X axis is scaled from 0% to 1.4%.

The first point of interest in Figure 4.9 is to compare the shapes as dimensionality increases. The hypersphere appears to perform progressively worse, by producing fewer true positives. However, the false positive rate does not increase. The hypersteinmetz and hypercube perform better as dimensionality increases in terms of true positives, but false positives also increase.

Tables 4.5 and 4.6 further elucidate the values shown in Figures 4.9a and 4.9d respectively. The classification accuracy results are shown for all three shapes (hypersphere, hypersteinmetz, and hypercube) for each set of coverage factors and radii in 2 and 11 dimensions respectively. The mean percentages of true positives and false positives are reported, along with the correlated standard deviations. These tables further illustrate that the hypersteinmetz and hypercube perform significantly better than the hypersphere as dimensionality is increased. The reason for this difference in performance is due to the shape of the detector, and specifically due to the content ratio of the hypersphere as compared to those of the hypercube and hypersteinmetz.

Specifically, Table 4.5 shows that the results for hyperspheres, hypersteinmetzes, and hypercubes are very similar in two dimensions, with each shape providing a highest true positive percentage around 87% and a false positive percentage of 0%. However, in 11 dimensions, shown in Table 4.6, the hypersphere does not provide a true positive percentage greater than 7.9%, while both the hypersteinmetz and hypercube each provide a true positive percentage of greater than 90%.

Table 4.5: Classification accuracy results for three different shapes in 2 dimensions

Hypersphere									
Cov Factor	2			8			16		
Radius	.015	.149	.283	.015	.149	.283	.016	.149	.283
True Pos %	71.1	6.7	0.2	86.0	32.9	1.0	87.5	33.1	1.7
TP% Std Dev	4.42	6.78	0.45	1.44	15.02	1.05	1.13	14.86	1.82
False Pos %	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
FP% Std Dev	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Hypersteinmetz									
Cov Factor	2			8			16		
Radius	.015	.149	.283	.015	.149	.283	.016	.149	.283
True Pos %	68.1	8.7	0.2	86.4	20.6	0.9	87.8	28.6	1.4
TP% Std Dev	4.18	13.14	0.26	1.58	16.92	1.03	1.24	15.29	1.21
False Pos %	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
FP% Std Dev	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Hypercube									
Cov Factor	2			8			16		
Radius	.015	.149	.283	.015	.149	.283	.016	.149	.283
True Pos %	74.5	4.2	0.0	85.1	21.9	0.6	86.7	36.9	0.6
TP% Std Dev	2.69	6.30	0.00	1.14	9.19	1.00	1.52	6.14	0.44
False Pos %	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
FP% Std Dev	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 4.6: Classification accuracy results for three different shapes in 11 dimensions

Hypersphere									
Cov Factor	2			8			16		
Radius	.379	.440	.500	.430	.465	.500	.458	.479	.500
True Pos %	0.0	0.0	0.1	0.2	0.1	2.5	7.9	2.1	0.5
TP% Std Dev	0.00	0.00	0.28	0.37	0.28	7.74	10.92	6.07	1.38
False Pos %	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
FP% Std Dev	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Hypersteinmetz									
Cov Factor	2			8			16		
Radius	.379	.440	.500	.430	.465	.500	.458	.479	.500
True Pos %	59.9	50.2	39.6	81.3	89.0	88.2	93.3	91.1	91.8
TP% Std Dev	22.06	26.23	29.22	16.20	7.76	7.13	2.31	3.95	3.79
False Pos %	0.3	0.2	0.2	0.6	1.1	1.0	0.9	1.3	1.2
FP% Std Dev	0.36	0.21	0.32	0.48	1.15	1.16	0.55	1.00	1.00
Hypercube									
Cov Factor	2			8			16		
Radius	.379	.440	.500	.430	.465	.500	.458	.479	.500
True Pos %	77.9	84.0	91.4	92.2	92.4	92.2	92.5	92.4	92.4
TP% Std Dev	22.10	16.44	1.29	2.13	2.03	2.46	2.12	2.03	2.03
False Pos %	0.2	0.1	0.1	0.3	0.2	0.2	0.3	0.3	0.3
FP% Std Dev	0.18	0.13	0.14	0.19	0.13	0.15	0.15	0.21	0.17

Table 4.7 shows the classification accuracy (true and false positive percentage means and standard deviations) for the best radius/coverage factor pairing for each number of dimensions for each shape. The best result is, as described previously, considered to be the one that is the closest in Manhattan distance to the point of 0% false positives and 100% true positives, thus equally weighting true and false positives. Each line reports the mean value and standard deviation obtained over a set of 10 runs. Table 4.7 demonstrates, again, the divergence between hyperspheres and the other two shapes. The hypersphere's best true positive percentage plummets from 87.5% in 2 dimensions down to 7.9% in 11 dimensions, while the hypersteinmetz increases from 87.8% to 93.3% and the hypercube from 86.7% up to 92.4%.

However, Table 4.7 also shows that the results for the hypersteinmetz and hypersphere perform equivalently to each other throughout all the dimensional sets. Given nine degrees of freedom, based on 10 tests per radius, the true positive percentages for all three shapes' best detector overlap 95% confidence intervals of each other in two dimensions. Additionally, the best hypercube and hypersteinmetz perform equivalently within a 95% confidence interval of each other for all true positive percentages and all but one false positive percentage (five dimensions).

Table 4.7: Classification accuracy for the best radius/coverage factor pairing for each number of dimensions for each shape

Hypersphere								
Num Dims	Coverage Factor	Radius	True Pos %	TP% Std Dev	TP% .95 CI	False Pos %	FP% Std Dev	FP% .95 CI
2	16	0.016	87.5	1.13	±0.69	0.0	0.00	±0.00
5	16	0.125	52.5	16.49	±10.08	0.1	0.14	±0.09
8	16	0.299	16.6	11.28	±6.89	0.0	0.10	±0.06
11	16	0.458	7.9	10.92	±6.67	0.0	0.00	±0.00
Hypersteinmetz								
Num Dims	Coverage Factor	Radius	True Pos %	TP% Std Dev	TP% .95 CI	False Pos %	FP% Std Dev	FP% .95 CI
2	16	0.016	87.8	1.24	±0.76	0.0	0.00	±0.00
5	16	0.125	79.1	7.29	±4.45	2.1	2.03	±1.24
8	16	0.399	79.3	10.69	±6.53	0.1	0.12	±0.07
11	16	0.458	93.3	2.31	±1.41	0.9	0.55	±0.34
Hypercube								
Num Dims	Coverage Factor	Radius	True Pos %	TP% Std Dev	TP% .95 CI	False Pos %	FP% Std Dev	FP% .95 CI
2	16	0.016	86.7	1.52	±0.93	0.0	0.00	±0.00
5	16	0.125	79.9	8.39	±5.13	0.1	0.13	±0.08
8	16	0.299	83.5	6.05	±3.70	0.2	0.18	±0.11
11	8	0.465	92.4	2.03	±1.24	0.2	0.13	±0.08

Figure 4.10 shows a three-way analysis of variance (ANOVA) of the true positive percentages of all runs, using the method described in [33]. This is used to determine which variables in the experiment contribute to the variance of the true positive classification results. The four plotted values in the figure are the most influential: radius size, coverage factor, shape, and unaccounted for factors. The actual percentages are reported in Table 4.8, along with the combined factors that are not shown in Figure 4.10. It can be seen in the figure and table that as the number of dimensions increases, the influence of the radius size, coverage factor and error each decrease; while the influence of the shape increases. The most dramatic decrease from 2 dimensions to 11 dimensions is the influence of radius size, which decreases from 91.3% in 2 dimensions down to 0.0% in

11 dimensions. While the largest increase is the influence of detector shape, increasing from 0.0% up to 85.8%. This is due to the influence of the hypersphere's content going to zero as the number of dimensions goes to infinity.

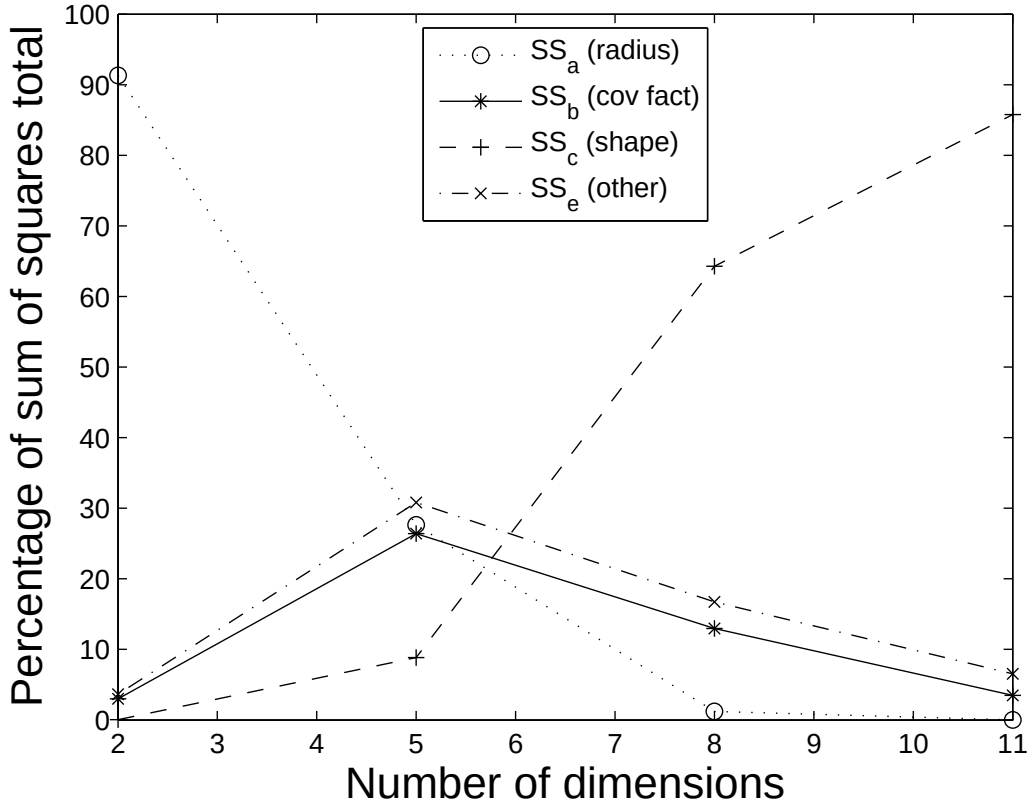


Figure 4.10: Three-way analysis of variance of true positive percentage, comparing the influence of a = radius size, b = coverage factor, c = shape, and e = unaccounted for factors as dimensionality increases

Table 4.8: Three-way analysis of variance (ANOVA) of true positive percentage, comparing the influence of a = radius size, b = coverage factor, c = shape, all combinations thereof, and e = unaccounted for factors as dimensionality increases

	Variance							
Dimensions	$SS_a\%$	$SS_b\%$	$SS_c\%$	$SS_{ab}\%$	$SS_{ac}\%$	$SS_{bc}\%$	$SS_{abc}\%$	$SS_e\%$
2	91.3	3.0	0.0	1.5	0.1	0.1	0.3	3.7
5	27.6	26.4	8.8	1.4	2.1	1.0	1.8	30.8
8	1.2	13.0	64.3	0.3	0.5	3.4	0.7	16.7
11	0.0	3.5	85.8	0.1	0.2	3.4	0.5	6.5

Two notes must be made. First, another factor in the reduction of the influence of radius size has to do with the reduction in the range of possible radii due to memory constraints. If more possible sizes were allowable, radius size would probably have a larger influence. Secondly, there are a few reasons for the variance due to unaccounted for factors SS_e. The choice of training and testing points has some influence on the results obtained, since they are randomly chosen this could have an effect. Additionally, the random placement of detectors could hold some sway into the error factor. Lastly, the ordering of dimensions could have some influence. Specifically, odd numbers of dimensions allow for biasing of features by the hypersteinmetz solid, since the hypersteinmetz must use one of the dimensions twice. As such, a future analysis could look into the best way to order dimensions and choose which dimension to double.

Lastly, one of the reasons that hypersteinmetzes were chosen was because of their feature bias in comparison to hypercubes. Classification accuracy results for hypercubes and hypersteinmetzes were very similar (within a 95% confidence interval of each other). Therefore, the current experiments did not provide enough data to compare feature bias

between hypersteinmetzes and hypercubes. As such, further experiments could be designed to specifically compare the feature bias of the different shapes.

5 Conclusions and Future Work

This research has shown that "Detector shape is an extremely important factor in the effectiveness of a real-valued negative selection system as the number of dimensions of data increases, especially in comparison to other factors such as radius size and coverage factor." Specifically, the hypersteinmetz solid, detector shape proposed herein, provides benefits of better classification accuracy in high dimensions when compared with the hypersphere.

This conclusory chapter aims to summarize the research that precedes it, in the following format. First, a set of conclusions is drawn from the results and analysis found in Chapter 4. Then, future projects that could extend the current research are proposed.

5.0.4 Detector radius sizing. The detector radius sizing experiment described in Section 3.3 and with results reported in Section 4.1 showed that the nearest neighbor method for finding a minimum radius size paired with the largest radius placement method for finding a maximum radius size provide a good boundary when searching for the optimal radius size for a real-valued negative selection algorithm. The nearest neighbor method helps to avoid overfitting by not allowing detectors that are too small, while the largest radius placement method avoids attempts to use detectors that are too large for the shape space.

5.0.5 Coverage factor. The coverage factor experiment described in Section 3.4 and with results reported in Section 4.2 showed that coverage factor is directly related to the classification accuracy results of a real-valued negative selection system. As the coverage factor is increased, there is a diminishing return on the improvement of the results. Additionally, increasing coverage factor also reduces variance between classification runs. These results, paired with the high-dimensional considerations

described in Section 3.5.2 demonstrate the vital need to understand the effective coverage provided within a high-dimensional real-valued system.

5.0.6 High-dimensional memory considerations. In addition to the considerations on coverage factor that high-dimensionality brings, the limitations of memory on the size of detector radii must be taken into context. Without taking the memory considerations of the current system setup into account, the effectiveness of the designed system is hurt significantly due to the necessary loss of coverage due to the number of usable detectors.

5.0.7 Detector shape comparison. Finally, it has been shown that detector shape not only plays a pivotal role in the coverage of high-dimensional shape-space, but also that detector shape is directly related the classification accuracy and becomes more important with increasing dimensionality. This is illustrated in two ways. First, the use of hyperspheres as detectors in high-dimensional real-valued negative selection systems has been shown problematic. Table 4.7 demonstrated that the hypersphere's best true positive percentage plummets from 87.5% in 2 dimensions down to 7.9% in 11 dimensions, while the hypersteinmetz increases from 87.8% to 93.3% and the hypercube from 86.7% up to 92.4%. Additionally, it was shown that detector shape becomes increasingly important as dimensionality increases through an analysis of variance. Table 4.8 showed that the influence of detector shape increased from 0.0% in 2 dimensions up to 85.8% in 11 dimensions. For these reasons, detector shape choice is a critical decision in the success of a real-valued negative selection based artificial immune system as dimensionality increases.

5.0.8 Effectiveness of hypersteinmetz. The hypersteinmetz solid, specifically, has proven to provide higher classification accuracies in high dimensions than the hypersphere. This is due to the fact that the content ratio of a hypersphere decreases factorially as a function of the number of dimensions, while the content ratio of a

hypersteinmetz decreases exponentially as a function of the number of dimensions. As such, the hypersteinmetz can be an effective detector shape for use in high dimensional real-valued negative selection systems.

5.1 Future Work

This thesis has covered only one small portion of the bigger vision for artificial immune systems and network anomaly detection research outlined in the introduction and motivation section. In order to work further toward the over-arching visions the following areas of research could be accomplished: feature bias comparison between hypercubes and hypersteinmetzes, creating accurate and effective data testing sets, determining the proper network implementation point of the outlined system, creating a distributed version of the system that allows for immune memory, and testing the effectiveness of the system on a scaled network.

5.1.1 Feature bias. The results obtained in this research did not conclusively show a difference in feature bias between detector shapes. It is likely that further testing of high-dimensional results could show that feature bias exists in higher quantity in hypercubes than in hypersteinmetzes. It is possible that this could be shown through analysis of the standard deviation.

5.1.2 Other network intrusion datasets. Another way that this research could be furthered is to compare results on more datasets to see if similar results are possible. One such dataset could be the University of Cambridge dataset referenced in Chapter 2. Tests on other datasets could then lead to implementation of a realtime network intrusion detection system onto a simulated network.

5.1.3 Dataset Creation. The data problems outlined previously are systemic throughout the whole of network intrusion detection research. Testing on datasets that are

not contrived simulations requires stripping important data out of actual network traffic data. Conversely, simulating data is difficult, because although it may account for the overall statistical probabilities of user network traffic, it can never completely reach real-world data. Still it is essential that new datasets be created and made available to the cyber operations research community. Whether it be created for overall intrusion detection or specific attack vector intrusion detection, a labeled dataset is of paramount importance in creating synergy within the research community.

5.1.4 Distributed Decision Making. One of the major contributions that the biological immune system can provide is that of immunological memory. In order to take full advantage of what can be done with this memory, a distributed system is extremely important. Previous work at the Air Force Institute of Technology [20, 21, 16, 17] has looked into distributed systems for network intrusion detection. By incorporating these concepts into an artificial immune system, it could be possible to allow immunities gained in one location to be conferred upon others.

5.1.5 Testing and Inoculation. Along with the distributed technology a logical follow-on is the idea of network inoculations, wherein immunity from a certain vector of attack would be conferred on a system through a benign network attack. In order to accomplish this, however, it is incumbent that the system be tested on a smaller network running real traffic patterns. This is not an easy task to accomplish and would require extensive planning of the testing and experimentation methods and goals before doing any sort of actual testing. Additionally, it would require a breadth of expertise from network engineering to computer programming that would likely necessitate an entire team rather than an individual effort.

Appendix A: Complete results of detector size range experiments

Table A.1: Results of detector size range experiments for the Comb dataset.

Dataset	Comb														
Radius	0.010	0.020	0.030	0.040	0.050	0.060	0.070	0.080	0.090	0.100	0.110	0.120	0.130	0.130	0.137
True Pos %	100.0	100.0	99.9	99.6	90.4	85.0	76.0	66.9	62.9	62.9	62.9	62.9	62.9	62.9	62.9
TP% Std Dev	0.00	0.00	0.09	0.14	0.44	0.17	0.66	0.20	0.00	0.00	0.04	0.00	0.06	0.05	0.06
False Pos %	94.0	54.9	22.4	17.2	11.2	7.0	4.6	3.3	2.3	2.2	2.2	2.2	2.1	2.1	2.1
FP% Std Dev	0.76	0.89	0.87	0.50	0.47	0.46	0.21	0.29	0.21	0.00	0.00	0.00	0.18	0.14	0.14
Radius	0.140	0.144	0.150	0.160	0.170	0.180	0.186	0.190	0.200	0.207					
True Pos %	62.8	62.9	62.8	62.9	62.8	62.8	62.8	62.8	62.9	53.1					
TP% Std Dev	0.09	0.04	0.07	0.06	0.11	0.08	0.11	0.09	0.05	0.14					
False Pos %	1.9	1.9	1.9	1.9	1.9	1.8	1.8	1.8	1.9	2.2					
FP% Std Dev	0.23	0.30	0.30	0.30	0.21	0.24	0.24	0.18	0.23	0.00					

Table A.2: Results of detector size range experiments for the Comb Negative dataset.

Dataset	Comb Negative														
Radius	0.010	0.020	0.025	0.030	0.040	0.050	0.053	0.060	0.070	0.080	0.090	0.100	0.110	0.120	0.130
True Pos %	100.0	100.0	100.0	99.9	99.4	89.8	85.3	79.3	72.7	51.1	48.8	46.8	46.2	45.4	45.0
TP% Std Dev	0.00	0.00	0.00	0.09	0.12	0.14	0.70	0.22	0.77	0.15	0.13	0.08	0.17	0.10	0.13
False Pos %	98.9	79.8	61.0	41.9	20.2	15.2	14.0	10.8	6.4	2.2	1.2	0.7	0.7	0.7	0.7
FP% Std Dev	0.39	0.48	0.73	0.73	0.70	0.25	0.60	0.32	0.58	0.19	0.12	0.00	0.00	0.00	0.00
Radius	0.140	0.150	0.160												
True Pos %	44.5	44.1	33.5												
TP% Std Dev	0.10	0.13	0.16												
False Pos %	0.7	0.7	0.7												
FP% Std Dev	0.00	0.00	0.00												

Table A.3: Results of detector size range experiments for the Intersection Thick dataset.

Dataset	Intersection Thick														
Radius	0.010	0.020	0.025	0.030	0.040	0.050	0.053	0.060	0.070	0.080	0.090	0.100	0.101	0.110	0.115
True Pos %	100.0	100.0	100.0	100.0	99.9	99.9	99.9	99.9	99.9	99.9	99.8	99.8	99.8	99.9	99.8
TP% Std Dev	0.03	0.00	0.00	0.04	0.07	0.05	0.05	0.06	0.06	0.06	0.09	0.10	0.07	0.05	0.09
False Pos %	17.5	7.5	7.0	6.5	5.5	4.5	5.0	3.5	4.0	2.5	1.5	2.0	1.0	0.5	1.0
FP% Std Dev	3.54	2.64	2.58	2.42	3.69	3.69	3.33	3.37	3.16	2.64	2.42	2.58	2.11	1.58	2.11
Radius	0.120	0.130	0.140	0.150	0.150	0.160	0.170	0.171	0.180	0.190	0.200	0.206	0.210	0.213	0.220
True Pos %	99.9	99.8	99.8	99.8	99.8	99.8	99.2	99.1	98.6	98.1	97.7	97.5	97.2	97.3	97.0
TP% Std Dev	0.10	0.08	0.08	0.10	0.11	0.09	0.19	0.20	0.24	0.21	0.19	0.29	0.23	0.24	0.30
False Pos %	0.5	0.0	1.5	1.5	0.5	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
FP% Std Dev	1.58	0.00	2.42	2.42	1.58	2.11	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Radius	0.230	0.240	0.248	0.250	0.260	0.262	0.270	0.276	0.280	0.290	0.297	0.300	0.304	0.310	0.318
True Pos %	96.6	79.4	78.9	78.6	78.5	78.6	72.0	55.3	53.7	52.4	52.1	51.9	51.6	51.2	50.9
TP% Std Dev	0.29	0.28	0.42	0.38	0.24	0.34	4.31	3.03	0.34	0.27	0.15	0.30	0.23	0.29	0.14
False Pos %	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
FP% Std Dev	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Radius	0.320	0.330	0.332	0.340	0.346	0.350	0.353								
True Pos %	50.7	50.2	50.1	49.8	49.3	49.0	48.7								
TP% Std Dev	0.15	0.10	0.13	0.25	0.18	0.15	0.18								
False Pos %	0.0	0.0	0.0	0.0	0.0	0.0	0.0								
FP% Std Dev	0.00	0.00	0.00	0.00	0.00	0.00	0.00								

Table A.4: Results of detector size range experiments for the Intersection Thick Negative dataset.

Dataset	Intersection Thick Negative							
Radius	0.010	0.011	0.018	0.020	0.030	0.038	0.040	0.050
True Pos %	100.0	100.0	100.0	100.0	100.0	98.0	96.1	88.2
TP% Std Dev	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
False Pos %	99.6	99.7	96.4	92.6	56.2	27.5	20.3	5.3
FP% Std Dev	0.18	0.10	0.30	0.25	0.49	0.45	0.22	0.05

Table A.5: Results of detector size range experiments for the Pentagram Big dataset.

Dataset	Pentagram Big														
Radius	0.010	0.020	0.030	0.040	0.050	0.060	0.070	0.080	0.090	0.100	0.110	0.116	0.120	0.130	0.140
True Pos %	100.0	100.0	100.0	100.0	99.9	99.9	99.9	99.8	99.6	99.5	99.4	99.3	99.2	99.2	98.9
TP% Std Dev	0.00	0.00	0.00	0.05	0.06	0.04	0.00	0.08	0.08	0.17	0.15	0.14	0.19	0.15	0.24
False Pos %	85.0	23.5	7.2	5.1	4.4	4.0	3.2	2.7	2.9	2.4	1.7	1.9	2.1	1.4	1.6
FP% Std Dev	1.34	1.30	0.55	0.96	0.57	0.74	0.76	0.89	0.71	0.65	0.47	0.67	0.74	0.59	0.47
Radius	0.144	0.150	0.158	0.160	0.170	0.180	0.190	0.200	0.210	0.220	0.230	0.235	0.240	0.250	0.260
True Pos %	98.8	98.7	98.4	98.3	97.8	97.2	96.8	96.6	95.7	94.9	94.1	93.7	92.7	86.2	79.5
TP% Std Dev	0.26	0.25	0.26	0.15	0.33	0.64	0.47	0.57	0.32	0.38	0.63	0.43	0.43	5.87	0.29
False Pos %	1.4	1.5	0.6	0.8	0.7	1.0	1.0	1.1	0.9	0.7	1.1	1.0	1.0	0.8	0.3
FP% Std Dev	0.38	0.28	0.59	0.28	0.50	0.47	0.47	0.55	0.35	0.50	0.32	0.57	0.35	0.42	0.35
Radius	0.270	0.280	0.290	0.300	0.310	0.320	0.326	0.330	0.340						
True Pos %	75.6	52.6	51.3	42.5	42.4	42.3	42.3	42.3	42.2						
TP% Std Dev	5.23	0.21	2.95	0.10	0.08	0.04	0.04	0.05	0.06						
False Pos %	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0						
FP% Std Dev	0.32	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00						

Table A.6: Results of detector size range experiments for the Pentagram Big Negative dataset.

Dataset	Pentagram Big Negative														
Radius	0.010	0.020	0.030	0.032	0.040	0.050	0.060	0.067	0.070	0.080	0.090	0.100	0.110	0.120	0.130
True Pos %	100.0	99.8	99.5	99.5	97.4	95.4	92.3	89.5	87.3	84.2	80.8	76.1	70.4	63.6	60.4
TP% Std Dev	0.00	0.00	0.00	0.00	0.15	0.15	0.31	0.43	0.31	0.22	0.43	0.58	0.31	0.42	0.16
False Pos %	98.6	83.7	39.9	35.7	13.9	7.3	5.6	5.3	5.1	4.6	3.8	2.9	1.8	1.4	1.1
FP% Std Dev	0.38	0.55	0.39	0.44	0.80	0.21	0.17	0.07	0.12	0.12	0.17	0.12	0.06	0.07	0.12
Radius	0.140	0.150	0.160	0.170	0.172	0.180	0.190	0.193							
True Pos %	58.4	56.7	55.2	54.3	54.2	53.2	51.5	50.3							
TP% Std Dev	0.34	0.25	0.19	0.20	0.16	0.49	0.10	106.14							
False Pos %	1.0	0.7	0.7	0.5	0.5	0.5	0.5	0.5							
FP% Std Dev	0.06	0.07	0.00	0.00	0.00	0.00	0.00	1.11							

Table A.7: Results of detector size range experiments for the Ring Thick dataset.

Dataset	Ring Thick														
Radius	0.010	0.020	0.030	0.040	0.050	0.060	0.070	0.073	0.080	0.090	0.100	0.110	0.120	0.130	0.140
True Pos %	100.0	100.0	99.8	98.5	97.1	96.2	95.1	94.9	94.2	92.8	91.7	90.8	89.7	88.3	86.7
TP% Std Dev	0.00	0.00	0.00	0.00	0.10	0.11	0.14	0.15	0.07	0.15	0.11	0.17	0.24	0.23	0.24
False Pos %	98.0	82.8	38.1	10.2	4.8	3.9	3.4	3.4	3.0	2.3	2.2	2.1	1.8	1.8	1.6
FP% Std Dev	0.33	0.44	0.73	0.99	0.08	0.12	0.18	0.12	0.12	0.13	0.14	0.13	0.12	0.16	0.12
Radius	0.150	0.150	0.157	0.160	0.164	0.170	0.180	0.185	0.190	0.192	0.200				
True Pos %	85.3	85.4	83.9	83.8	83.6	52.2	51.5	51.0	50.1	49.7	48.9				
TP% Std Dev	0.50	0.43	0.21	0.10	0.15	0.22	0.07	0.22	0.14	0.10	0.10				
False Pos %	1.2	1.2	1.0	0.9	0.8	0.7	0.6	0.6	0.6	0.6	0.6				
FP% Std Dev	0.13	0.13	0.13	0.12	0.08	0.00	0.00	0.00	0.00	0.00	0.00				

Table A.8: Results of detector size range experiments for the Ring Thick Negative dataset.

Dataset	Ring Thick Negative														
Radius	0.010	0.020	0.030	0.039	0.040	0.050	0.060	0.060	0.070	0.080	0.081	0.090	0.100	0.110	0.120
True Pos %	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
TP% Std Dev	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.05	0.00	0.04	0.05	0.04	0.05
False Pos %	90.9	40.3	17.6	13.1	12.9	10.9	9.9	9.6	9.1	7.9	7.8	7.7	6.6	7.6	6.7
FP% Std Dev	1.13	0.90	0.49	0.61	0.68	0.97	0.71	0.73	0.77	0.61	0.91	0.62	1.37	0.71	0.68
Radius	0.130	0.140	0.150	0.160	0.165	0.170	0.172								
True Pos %	100.0	100.0	100.0	100.0	100.0	99.8	95.9								
TP% Std Dev	0.08	0.06	0.00	0.04	0.00	0.10	0.18								
False Pos %	6.5	6.0	6.7	6.9	7.1	7.6	6.2								
FP% Std Dev	0.76	0.91	0.73	0.64	0.55	0.66	0.28								

Appendix B: Complete results of coverage factor experiments

Table B.1: Results of coverage factor experiments using coverage factor 2, true and false positive percentages are the mean percentage calculated over 10 runs per radius, true and false positive percentages standard deviations are calculated over 10 runs per radius.

Coverage Factor f = 2									
Dataset	Setosa								
Radius	0.080	0.172	0.265	0.357	0.449	0.541	0.633	0.726	0.818
True Pos %	80.4	66.8	67.2	62.2	57.8	48.2	40.2	57.0	81.0
TP% Std Dev	16.56	28.87	30.95	35.16	36.60	40.25	42.51	43.92	49.48
False Pos %	62.3	15.9	5.4	3.7	0.6	0.5	0.6	57.0	81.0
FP% Std Dev	17.11	19.13	10.06	6.39	4.69	1.81	2.75	35.23	49.48
Dataset	Versicolor								
Radius	0.075	0.164	0.254	0.343	0.433	0.522	0.612	0.701	0.791
True Pos %	86.2	80.2	66.6	60.0	42.6	29.6	13.0	37.0	50.0
TP% Std Dev	16.91	17.60	26.01	26.57	29.63	31.15	26.53	24.07	49.67
False Pos %	53.4	15.3	6.2	4.0	1.8	1.1	0.6	37.0	50.0
FP% Std Dev	16.12	15.42	10.19	6.91	5.64	3.92	3.37	20.73	49.67
Dataset	Virginica								
Radius	0.068	0.159	0.250	0.342	0.433	0.525	0.616	0.707	0.799
True Pos %	83.6	73.4	59.2	41.0	18.8	8.6	2.6	6.0	8.0
TP% Std Dev	15.16	22.59	21.72	28.88	27.83	21.92	16.96	19.38	18.05
False Pos %	53.5	13.0	5.0	2.8	1.4	1.4	0.8	6.0	8.0
FP% Std Dev	15.47	16.57	8.45	5.94	4.47	4.56	3.62	18.22	18.05

Table B.2: Results of coverage factor experiments using coverage factor 4, true and false positive percentages are the mean percentage calculated over 10 runs per radius, true and false positive percentages standard deviations are calculated over 10 runs per radius.

Coverage Factor f = 4									
Dataset	Setosa								
Radius	0.080	0.172	0.265	0.357	0.449	0.541	0.633	0.726	0.818
True Pos %	94.2	88.0	86.0	87.6	75.2	74.8	73.4	69.2	85.0
TP% Std Dev	11.93	19.21	20.24	23.08	28.27	33.86	30.32	36.61	37.18
False Pos %	78.9	26.4	10.7	4.0	2.5	1.3	1.0	0.0	85.0
FP% Std Dev	12.25	22.45	12.62	7.94	6.46	4.47	3.02	2.86	30.17
Dataset	Versicolor								
Radius	0.075	0.164	0.254	0.343	0.433	0.522	0.612	0.701	0.791
True Pos %	98.6	92.4	84.2	74.4	64.0	49.8	33.2	27.0	26.4
TP% Std Dev	5.39	10.58	20.07	20.66	24.73	28.18	29.75	27.02	24.93
False Pos %	65.8	22.7	11.0	4.6	2.9	2.4	1.6	1.3	1.5
FP% Std Dev	15.05	20.20	12.45	8.22	7.04	4.69	5.95	3.92	4.31
Dataset	Virginica								
Radius	0.068	0.159	0.250	0.342	0.433	0.525	0.616	0.707	0.799
True Pos %	97.0	90.6	79.4	62.8	36.6	17.6	6.4	2.6	7.0
TP% Std Dev	7.29	14.08	16.84	25.55	28.10	25.97	21.08	14.45	14.96
False Pos %	74.1	24.7	10.9	5.8	3.8	1.9	1.3	1.2	7.0
FP% Std Dev	12.60	20.15	11.37	7.64	6.21	5.77	3.02	3.75	10.93

Table B.3: Results of coverage factor experiments using coverage factor 6, true and false positive percentages are the mean percentage calculated over 10 runs per radius, true and false positive percentages standard deviations are calculated over 10 runs per radius.

Coverage Factor f = 6									
Dataset	Setosa								
Radius	0.080	0.172	0.265	0.357	0.449	0.541	0.633	0.726	0.818
True Pos %	97.6	96.6	95.4	92.8	87.8	83.4	84.0	89.8	80.4
TP% Std Dev	7.24	8.61	8.95	14.59	21.24	23.99	18.70	28.32	20.82
False Pos %	87.1	30.8	13.2	5.7	3.6	2.1	1.8	1.4	0.4
FP% Std Dev	11.96	23.68	14.02	10.33	6.38	4.98	4.75	3.64	3.29
Dataset	Versicolor								
Radius	0.075	0.164	0.254	0.343	0.433	0.522	0.612	0.701	0.791
True Pos %	99.4	98.0	91.4	85.2	76.8	63.0	42.0	40.2	32.0
TP% Std Dev	2.96	6.03	11.22	17.94	18.38	26.00	30.44	26.76	27.86
False Pos %	74.0	28.8	12.9	6.5	5.6	2.8	3.3	2.4	1.9
FP% Std Dev	13.00	18.92	12.45	9.89	7.07	6.01	5.77	6.70	5.42
Dataset	Virginica								
Radius	0.068	0.159	0.250	0.342	0.433	0.525	0.616	0.707	0.799
True Pos %	99.2	95.6	85.2	73.6	47.0	19.8	12.6	4.2	0.8
TP% Std Dev	3.61	9.50	15.04	20.88	28.16	29.23	20.89	20.60	15.30
False Pos %	79.5	27.9	13.7	7.3	4.1	3.3	2.6	1.7	1.3
FP% Std Dev	12.27	19.30	11.61	9.06	6.74	6.08	5.54	5.36	4.12

Table B.4: Results of coverage factor experiments using coverage factor 8, true and false positive percentages are the mean percentage calculated over 10 runs per radius, true and false positive percentages standard deviations are calculated over 10 runs per radius.

Coverage Factor f = 8									
Dataset	Setosa								
Radius	0.080	0.172	0.265	0.357	0.449	0.541	0.633	0.726	0.818
True Pos %	99.4	98.0	97.6	99.2	94.2	89.0	91.4	88.6	87.4
TP% Std Dev	4.69	6.46	7.61	6.16	9.09	17.63	22.03	19.17	22.53
False Pos %	91.4	38.8	14.4	8.9	3.2	2.5	1.9	1.0	1.3
FP% Std Dev	8.62	23.13	17.39	10.22	7.47	6.26	5.04	3.84	2.69
Dataset	Versicolor								
Radius	0.075	0.164	0.254	0.343	0.433	0.522	0.612	0.701	0.791
True Pos %	99.8	99.4	95.0	91.2	85.0	69.0	55.6	48.4	45.6
TP% Std Dev	2.11	3.61	8.42	12.42	15.63	23.69	27.79	26.72	28.95
False Pos %	78.6	34.7	15.5	8.4	5.6	4.2	3.9	2.4	2.7
FP% Std Dev	11.52	19.89	14.32	10.58	7.85	7.65	6.96	6.21	5.47
Dataset	Virginica								
Radius	0.068	0.159	0.250	0.342	0.433	0.525	0.616	0.707	0.799
True Pos %	99.8	98.2	91.0	84.8	59.2	29.6	13.6	5.8	1.4
TP% Std Dev	0.00	5.39	11.05	17.90	27.41	29.54	25.48	22.96	14.93
False Pos %	83.4	33.4	17.8	9.5	6.5	4.1	3.7	2.3	2.3
FP% Std Dev	10.49	19.52	12.65	10.15	7.61	7.47	6.32	5.74	4.13

Table B.5: Results of coverage factor experiments using coverage factor 10, true and false positive percentages are the mean percentage calculated over 10 runs per radius, true and false positive percentages standard deviations are calculated over 10 runs per radius.

Coverage Factor f = 10									
Dataset	Setosa								
Radius	0.080	0.172	0.265	0.357	0.449	0.541	0.633	0.726	0.818
True Pos %	99.2	99.6	98.6	98.8	97.0	91.6	94.6	94.8	94.6
TP% Std Dev	3.61	3.61	5.50	5.02	9.40	14.63	14.21	11.24	13.00
False Pos %	93.5	44.2	16.2	8.9	4.2	4.1	2.0	1.7	1.4
FP% Std Dev	7.10	22.40	17.45	11.02	7.30	5.34	5.19	4.62	3.94
Dataset	Versicolor								
Radius	0.075	0.164	0.254	0.343	0.433	0.522	0.612	0.701	0.791
True Pos %	99.8	99.6	96.6	92.6	89.0	79.0	65.6	52.2	50.2
TP% Std Dev	2.11	2.96	7.47	9.87	14.11	16.97	23.65	21.94	27.98
False Pos %	81.1	36.0	19.5	10.2	7.4	5.0	5.6	2.3	2.9
FP% Std Dev	11.04	20.69	16.24	11.57	8.89	7.79	7.37	7.07	4.69
Dataset	Virginica								
Radius	0.068	0.159	0.250	0.342	0.433	0.525	0.616	0.707	0.799
True Pos %	99.8	98.0	92.6	87.6	65.2	38.4	21.6	3.4	1.0
TP% Std Dev	0.00	6.32	10.30	14.45	23.90	25.31	29.83	23.61	8.61
False Pos %	87.2	34.9	18.4	10.5	8.9	4.4	3.6	2.5	3.1
FP% Std Dev	8.62	21.35	12.83	8.65	8.21	6.78	5.99	5.87	5.36

Table B.6: Results of coverage factor experiments using coverage factor 12, true and false positive percentages are the mean percentage calculated over 10 runs per radius, true and false positive percentages standard deviations are calculated over 10 runs per radius.

Coverage Factor f = 12									
Dataset	Setosa								
Radius	0.080	0.172	0.265	0.357	0.449	0.541	0.633	0.726	0.818
True Pos %	100.0	99.6	99.2	99.2	96.4	93.4	96.2	97.0	96.4
TP% Std Dev	0.00	2.11	3.61	5.84	10.47	11.80	12.45	9.43	8.43
False Pos %	93.7	46.2	20.9	11.4	5.8	3.8	2.5	2.0	1.6
FP% Std Dev	7.24	21.46	18.69	11.80	8.51	6.04	4.78	4.51	4.10
Dataset	Versicolor								
Radius	0.075	0.164	0.254	0.343	0.433	0.522	0.612	0.701	0.791
True Pos %	100.0	100.0	95.6	94.0	93.8	79.0	68.4	60.0	50.6
TP% Std Dev	0.00	0.00	8.78	10.23	9.49	17.37	19.71	25.95	28.16
False Pos %	82.6	39.4	20.7	10.8	8.3	5.2	3.6	4.1	3.1
FP% Std Dev	11.40	20.24	15.88	11.88	9.52	9.48	7.07	6.10	6.44
Dataset	Virginica								
Radius	0.068	0.159	0.250	0.342	0.433	0.525	0.616	0.707	0.799
True Pos %	99.6	98.4	93.8	86.8	70.4	40.4	20.6	5.0	1.0
TP% Std Dev	2.96	5.02	10.40	14.76	21.94	26.72	25.81	23.27	13.42
False Pos %	87.7	38.2	20.9	12.8	8.3	4.7	4.0	4.3	3.1
FP% Std Dev	9.92	22.34	13.14	11.86	8.80	7.58	6.19	6.54	5.98

Table B.7: Results of coverage factor experiments using coverage factor 14, true and false positive percentages are the mean percentage calculated over 10 runs per radius, true and false positive percentages standard deviations are calculated over 10 runs per radius.

Coverage Factor f = 14									
Dataset	Setosa								
Radius	0.080	0.172	0.265	0.357	0.449	0.541	0.633	0.726	0.818
True Pos %	99.8	100.0	99.2	100.0	97.4	97.2	96.6	97.6	97.2
TP% Std Dev	2.11	0.00	2.96	2.96	6.16	7.24	9.43	6.59	8.91
False Pos %	94.9	48.0	23.9	11.3	7.6	4.4	3.3	2.8	1.6
FP% Std Dev	6.74	21.23	15.74	12.73	8.64	7.03	5.70	5.23	4.75
Dataset	Versicolor								
Radius	0.075	0.164	0.254	0.343	0.433	0.522	0.612	0.701	0.791
True Pos %	100.0	100.0	97.4	95.0	92.0	82.6	75.4	63.0	57.0
TP% Std Dev	0.00	0.00	6.32	8.74	10.58	19.04	20.07	26.52	26.39
False Pos %	83.8	40.7	20.6	12.2	9.0	6.6	4.7	3.6	2.6
FP% Std Dev	9.93	19.39	15.65	13.48	9.36	8.24	9.35	7.07	6.00
Dataset	Virginica								
Radius	0.068	0.159	0.250	0.342	0.433	0.525	0.616	0.707	0.799
True Pos %	100.0	99.4	96.8	90.4	76.2	44.8	26.0	7.2	1.0
TP% Std Dev	0.00	3.61	7.24	12.16	20.33	30.36	29.52	24.80	16.12
False Pos %	88.5	41.5	23.2	15.2	9.6	7.3	4.8	4.2	3.9
FP% Std Dev	9.37	18.33	14.56	11.10	9.35	7.57	7.69	6.91	6.22

Table B.8: Results of coverage factor experiments using coverage factor 15, true and false positive percentages are the mean percentage calculated over 10 runs per radius, true and false positive percentages standard deviations are calculated over 10 runs per radius.

Coverage Factor f = 15									
Dataset	Setosa								
Radius	0.080	0.172	0.265	0.357	0.449	0.541	0.633	0.726	0.818
True Pos %	100.0	100.0	100.0	99.6	98.4	97.6	97.4	97.8	98.2
TP% Std Dev	0.00	0.00	0.00	2.11	4.14	7.48	8.43	6.59	7.61
False Pos %	95.1	50.0	24.1	12.6	7.9	4.2	2.8	2.2	1.3
FP% Std Dev	6.23	20.89	17.93	13.29	9.99	6.76	6.16	4.57	4.62
Dataset	Versicolor								
Radius	0.075	0.164	0.254	0.343	0.433	0.522	0.612	0.701	0.791
True Pos %	100.0	99.8	99.0	95.6	92.8	86.0	76.2	65.0	55.8
TP% Std Dev	0.00	2.11	4.14	7.00	11.95	12.35	19.53	21.11	25.93
False Pos %	82.9	43.1	23.2	12.5	8.9	5.8	6.3	4.8	3.3
FP% Std Dev	9.86	20.25	18.13	12.66	10.22	8.14	7.85	7.00	6.73
Dataset	Virginica								
Radius	0.068	0.159	0.250	0.342	0.433	0.525	0.616	0.707	0.799
True Pos %	100.0	99.6	96.8	89.8	76.2	48.2	27.4	10.0	1.8
TP% Std Dev	0.00	2.11	7.29	14.66	17.03	24.99	29.22	26.63	18.29
False Pos %	89.1	40.4	23.8	16.4	9.9	7.8	4.8	3.8	3.9
FP% Std Dev	8.82	20.99	12.85	10.52	8.45	7.58	8.01	5.42	5.67

Table B.9: Results of coverage factor experiments using coverage factor 20, true and false positive percentages are the mean percentage calculated over 10 runs per radius, true and false positive percentages standard deviations are calculated over 10 runs per radius.

Coverage Factor f = 20									
Dataset	Setosa								
Radius	0.080	0.172	0.265	0.357	0.449	0.541	0.633	0.726	0.818
True Pos %	100.0	100.0	100.0	100.0	99.2	97.6	98.0	98.4	98.6
TP% Std Dev	0.00	0.00	0.00	0.00	2.96	5.02	7.24	6.03	5.02
False Pos %	97.1	56.3	28.6	16.3	7.4	5.6	3.8	3.2	2.0
FP% Std Dev	5.50	19.97	17.93	12.45	10.65	6.80	6.04	5.75	4.85
Dataset	Versicolor								
Radius	0.075	0.164	0.254	0.343	0.433	0.522	0.612	0.701	0.791
True Pos %	100.0	100.0	99.2	97.4	94.8	91.4	80.8	72.4	65.4
TP% Std Dev	0.00	0.00	3.61	6.74	8.42	12.11	16.44	20.65	22.50
False Pos %	85.8	47.6	25.8	14.8	9.6	7.8	7.8	6.3	4.8
FP% Std Dev	9.14	20.36	17.90	13.18	10.83	8.49	7.97	9.21	7.93
Dataset	Virginica								
Radius	0.068	0.159	0.250	0.342	0.433	0.525	0.616	0.707	0.799
True Pos %	100.0	99.8	97.8	94.4	84.4	56.0	30.6	13.0	2.4
TP% Std Dev	0.00	2.11	6.32	8.78	16.85	25.82	28.75	27.68	19.02
False Pos %	91.6	46.1	27.1	16.5	12.3	8.0	7.0	5.0	4.3
FP% Std Dev	8.33	19.91	16.91	11.52	9.00	7.18	8.37	7.04	7.23

Table B.10: Results of coverage factor experiments using coverage factor 25, true and false positive percentages are the mean percentage calculated over 10 runs per radius, true and false positive percentages standard deviations are calculated over 10 runs per radius.

Coverage Factor f = 25									
Dataset	Setosa								
Radius	0.080	0.172	0.265	0.357	0.449	0.541	0.633	0.726	0.818
True Pos %	100.0	100.0	100.0	100.0	99.2	98.6	98.8	98.2	99.0
TP% Std Dev	0.00	0.00	0.00	0.00	3.61	4.14	4.61	5.39	5.39
False Pos %	97.1	58.6	30.3	17.0	9.0	6.4	5.0	3.5	2.6
FP% Std Dev	4.61	19.43	18.24	13.15	10.24	6.64	6.72	6.39	5.26
Dataset	Versicolor								
Radius	0.075	0.164	0.254	0.343	0.433	0.522	0.612	0.701	0.791
True Pos %	100.0	100.0	99.0	98.0	97.4	93.8	89.4	79.6	68.2
TP% Std Dev	0.00	0.00	3.61	6.03	7.00	8.21	12.32	16.12	21.04
False Pos %	87.2	50.0	27.4	18.8	13.2	9.7	7.9	7.2	5.9
FP% Std Dev	8.30	20.18	17.45	12.74	12.20	10.11	9.23	9.46	8.18
Dataset	Virginica								
Radius	0.068	0.159	0.250	0.342	0.433	0.525	0.616	0.707	0.799
True Pos %	100.0	100.0	98.4	95.6	86.4	62.8	33.6	17.0	3.8
TP% Std Dev	0.00	0.00	4.61	8.07	13.49	21.53	26.29	23.65	21.95
False Pos %	92.3	48.9	30.7	19.7	13.8	9.6	7.8	5.9	5.1
FP% Std Dev	7.65	20.85	16.37	12.18	10.69	8.68	8.24	7.73	7.05

Table B.11: Results of coverage factor experiments using coverage factor 30, true and false positive percentages are the mean percentage calculated over 10 runs per radius, true and false positive percentages standard deviations are calculated over 10 runs per radius.

Coverage Factor f = 30									
Dataset	Setosa								
Radius	0.080	0.172	0.265	0.357	0.449	0.541	0.633	0.726	0.818
True Pos %	100.0	100.0	100.0	100.0	99.8	99.0	98.8	98.6	99.2
TP% Std Dev	0.00	0.00	0.00	0.00	2.11	4.14	2.96	5.39	4.61
False Pos %	97.6	60.7	33.8	19.5	11.2	6.5	5.3	4.0	3.1
FP% Std Dev	4.18	18.85	18.17	14.53	11.04	8.10	7.81	6.74	5.32
Dataset	Versicolor								
Radius	0.075	0.164	0.254	0.343	0.433	0.522	0.612	0.701	0.791
True Pos %	100.0	100.0	99.4	99.0	98.4	95.0	88.0	81.2	75.2
TP% Std Dev	0.00	0.00	2.11	5.02	5.39	7.47	12.65	17.74	20.07
False Pos %	88.3	50.3	30.2	17.6	12.3	11.6	9.6	8.7	7.8
FP% Std Dev	8.10	18.97	18.29	16.11	11.54	10.83	10.44	9.35	9.14
Dataset	Virginica								
Radius	0.068	0.159	0.250	0.342	0.433	0.525	0.616	0.707	0.799
True Pos %	100.0	100.0	98.2	95.0	85.8	66.0	40.4	15.8	3.6
TP% Std Dev	0.00	0.00	5.39	7.29	14.76	21.18	27.22	27.58	22.40
False Pos %	93.1	50.9	32.3	21.5	16.0	11.9	8.6	6.7	5.6
FP% Std Dev	7.23	21.10	16.06	12.10	9.39	9.37	8.77	7.83	7.71

Appendix C: Complete results of detector shape comparison experiments

Table C.1: Results of detector shape comparison experiments, true and false positive percentages are the mean percentage calculated over 10 runs per radius size, true and false positive percentages standard deviations are calculated over 10 runs per radius size.

Number of dimensions $n = 2$

Shape	Hypersphere								Hypersteinmetz								Hypercube							
Coverage Factor	2		8		16		2		8		16		2		8		16							
Radius	0.015	0.149	0.283	0.014	0.149	0.283	0.016	0.149	0.283	0.015	0.149	0.283	0.014	0.149	0.283	0.016	0.149	0.283	0.015	0.149	0.283	0.014	0.149	0.283	0.016	0.149	0.283
True Pos %	71.1	6.7	0.2	86.0	32.9	1.0	87.5	33.1	1.7	68.1	8.7	0.2	86.4	20.6	0.9	87.8	28.6	1.4	74.5	4.2	0.0	85.1	21.9	0.6	86.7	36.9	0.6
TP% Std Dev	4.42	6.78	0.45	1.44	15.02	1.05	1.13	14.86	1.82	4.18	13.14	0.26	1.58	16.92	1.03	1.24	15.29	1.21	2.69	6.30	0.00	1.14	9.19	1.00	1.52	6.14	0.44
False Pos %	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
FP% Std Dev	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Number of dimensions $n = 5$

Shape	Hypersphere								Hypersteinmetz								Hypercube							
Coverage Factor	2		8		16		2		8		16		2		8		16							
Radius	0.082	0.291	0.500	0.109	0.304	0.500	0.125	0.312	0.500	0.082	0.291	0.500	0.109	0.304	0.500	0.125	0.312	0.500	0.082	0.291	0.500	0.109	0.304	0.500	0.125	0.312	0.500
True Pos %	11.9	5.2	0.0	41.8	7.9	17.8	52.5	30.9	23.3	34.8	13.8	5.7	61.6	42.9	12.0	79.1	55.1	47.3	39.0	11.5	12.0	67.1	35.7	19.1	79.9	40.0	30.3
TP% Std Dev	9.26	10.09	0.00	15.60	10.45	19.49	16.49	20.35	24.22	15.66	17.01	12.82	8.15	23.24	19.36	7.29	17.32	17.71	15.33	13.47	16.68	12.64	18.03	19.20	8.39	17.08	16.34
False Pos %	0.1	0.0	0.0	0.1	0.1	0.0	0.1	0.0	0.0	0.5	0.0	0.0	1.0	0.3	0.0	2.1	0.1	0.1	0.1	0.0	0.0	0.1	0.0	0.0	0.1	0.0	0.0
FP% Std Dev	0.11	0.00	0.00	0.11	0.11	0.00	0.14	0.00	0.00	0.95	0.00	0.00	1.98	0.43	0.00	2.03	0.18	0.18	0.18	0.00	0.00	0.13	0.00	0.00	0.13	0.00	0.00

Number of dimensions $n = 8$

Shape	Hypersphere								Hypersteinmetz								Hypercube							
Coverage Factor	2		8		16		2		8		16		2		8		16							
Radius	0.230	0.365	0.500	0.274	0.387	0.500	0.299	0.399	0.500	0.230	0.365	0.500	0.274	0.387	0.500	0.299	0.399	0.500	0.230	0.365	0.500	0.274	0.387	0.500	0.299	0.399	0.500
True Pos %	0.1	0.6	2.7	6.8	6.1	3.7	16.6	7.3	12.7	35.6	23.0	28.5	60.1	62.4	45.5	79.2	79.3	66.9	64.6	48.5	41.1	80.7	74.6	66.8	83.5	81.0	79.2
TP% Std Dev	0.45	1.78	8.46	6.83	8.29	6.09	11.28	6.67	14.38	18.64	19.22	24.05	14.99	14.91	20.93	10.56	10.69	20.14	12.71	25.35	21.23	9.10	13.51	20.70	6.05	9.35	6.70
False Pos %	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.1	0.1	0.0	0.1	0.1	0.1	0.1	0.0	0.0	0.2	0.1	0.1	0.2	0.2	0.2
FP% Std Dev	0.07	0.00	0.00	0.07	0.00	0.00	0.10	0.00	0.00	0.11	0.00	0.00	0.12	0.12	0.10	0.12	0.12	0.12	0.12	0.07	0.07	0.11	0.12	0.16	0.18	0.16	0.18

Number of dimensions $n = 11$

Shape	Hypersphere								Hypersteinmetz								Hypercube							
Coverage Factor	2		8		16		2		8		16		2		8		16							
Radius	0.379	0.440	0.500	0.430	0.465	0.500	0.458	0.479	0.500	0.379	0.440	0.500	0.430	0.465	0.500	0.458	0.479	0.500	0.379	0.440	0.500	0.430	0.465	0.500	0.458	0.479	0.500
True Pos %	0.0	0.0	0.1	0.2	0.1	2.5	7.9	2.1	0.5	59.9	50.2	39.6	81.3	89.0	88.2	93.3	91.1	91.8	77.9	84.0	91.4	92.2	92.4	92.2	92.5	92.4	92.4
TP% Std Dev	0.00	0.00	0.28	0.37	0.28	7.74	10.92	6.07	1.38	22.06	26.23	29.22	16.20	7.76	7.13	2.31	3.95	3.79	22.10	16.44	1.29	2.13	2.03	2.46	2.12	2.03	2.03
False Pos %	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.2	0.2	0.6	1.1	1.0	0.9	1.3	1.2	0.2	0.1	0.1	0.3	0.2	0.2	0.3	0.3	0.3
FP% Std Dev	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.36	0.21	0.32	0.48	1.15	1.16	0.55	1.00	1.00	0.18	0.13	0.14	0.19	0.13	0.15	0.15	0.21	0.17

Bibliography

[1] Aggarwal, C., A. Hinneburg, and D. Keim. "On the surprising behavior of distance metrics in high dimensional space". *Database TheoryICDT 2001*, 420–434, 2001.

[2] Araujo, Nelcilena, Ruy de Oliveira, Ed'Wilson Ferreira, Ailton Akira Shinoda, and Bharat Bhargava. "Identifying important characteristics in the KDD99 intrusion detection dataset by feature selection using a hybrid approach". *Telecommunications (ICT), 2010 IEEE 17th International Conference on*, 552–558. IEEE, 2010.

[3] Athanasiades, Nicholas, Randal Abler, John Levine, Henry Owen, and George Riley. "Intrusion detection testing and benchmarking methodologies". *First IEEE International Information Assurance Workshop*, 63–72. IEEE, 2003.

[4] Bellman, R.E., R.E. Bellman, R.E. Bellman, and R.E. Bellman. *Adaptive control processes: a guided tour*. Princeton University Press, 1961.

[5] Boln-Canedo, V., N. Snchez-Maroo, and A. Alonso-Betanzos. "Feature selection and classification in multiple class datasets: An application to KDD Cup 99 dataset". *Expert Systems with Applications*, 38(5):5947 – 5957, 2011. ISSN 0957-4174. URL http://www.sciencedirect.com/science/article/pii/S0957417410012650.

[6] Chmielewski, A. and S.T. Wierzchon. "On the distance norms for detecting anomalies in multidimensional datasets". *Zeszyty Naukowe Politechniki Białostockiej*, 2:39–49, 2007.

[7] Clark, Philip J. and Francis C. Evans. "Distance to nearest neighbor as a measure of spatial relationships in populations". *Ecology*, 35(4):445–453, 1954.

[8] Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, third edition, 2009.

[9] Dasgupta, Dipankar. *An Overview of Artificial Immune Systems and Their Applications*, 3–21. Artificial Immune Systems and Their Applications. Springer-Verlag, Berlin, 1999. ISBN 3-540-64390-7.

[10] Dasgupta, Dipankar. "Artificial Immune Systems: Datasets", 2012. URL http://ais.cs.memphis.edu/index.php?page=datasets.

[11] Davis, Jonathan J. and Andrew J. Clark. "Data preprocessing for anomaly based network intrusion detection: A review". *Computers and Security*, 1–23, 2011.

[12] Fisher, R.A. "The use of multiple measurements in taxonomic problems". *Ann. Eug. Part II*, 7:179188, 1936.

[13] Forrest, S., A. S. Perelson, L. Allen, and R. Cherukuri. "Self-nonself discrimination in a computer". *Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy*, 202–212. IEEE, 1994.

[14] Gonzalez, F., D. Dasgupta, and R. Kozma. "Combining negative selection and classification techniques for anomaly detection". *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, volume 1, 705–710. IEEE, 2002.

[15] Gonzalez, Fabio, Dipankar Dasgupta, and Luis Fernando Nino. "A randomized real-valued negative selection algorithm". *Proceedings of the 2nd International Conference on Artificial Immune Systems*, 261–272. ICARIS, 2003.

[16] Hancock, D. *A Multi Agent System for Flow-Based Intrusion Detection Using Reputation and Evolutionary Computation*. Master's thesis, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, March 2011.

[17] Hancock, David and Gary P. Lamont. "Multi agent systems on Military networks". *Symposium on Computational Intelligence in Cyber Security (CICS)*, 100–107. IEEE, April 2011.

[18] Harris, J. W. and H. Stocker. *Segment of a Circle 3.8.6*, 92–93. Springer-Verlag, 1998.

[19] Heidemann, John and Cristos Papadopoulos. "Uses and challenges for network datasets". *Cybersecurity Applications and Technology Conference for Homeland Security*, 73–82. IEEE, 2009.

[20] Holloway, E. *Self Organized Multi Agent Swarms (SOMAS) for Accomplishing Network Goals*. Master's thesis, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, March 2009.

[21] Holloway, E.M., G.B. Lamont, and G.L. Peterson. "Uses and challenges for network datasets". *Symposium on Computational Intelligence in Cyber Security, 2009. CICS '09*, 144–151. April.

[22] Ji, Zhou and Dipankar Dasgupta. "Real-valued negative selection algorithm with variable-sized detectors". *Proceedings of the 2004 Conference on Genetic and Evolutionary Computation*, 287–298. Springer, 2004.

[23] Ji, Zhou and Dipankar Dasgupta. "Estimating the detector coverage in a negative selection algorithm". *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, 281–288. ACM, 2005.

[24] Ji, Zhou and Dipankar Dasgupta. "Applicability issues of the real-valued negative selection algorithms". *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, 111–118. ACM, 2006.

[25] Ji, Zhou and Dipankar Dasgupta. "V-detector: An efficient negative selection algorithm with "probably adequate" detector coverage". *Information Sciences*, 179(10):1390–1406, 2009.

[26] Kayacik, H. Gunes, A Nur Zincir-Heywood, and Malcolm I. Heywood. "Selecting features for intrusion detection: a feature relevance analysis on KDD 99 intrusion detection datasets". *Proceedings of the Third Annual Conference on Privacy, Security and Trust (PST-2005)*. Citeseer, 2005.

[27] Kendall, Kristopher. *A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems*. Ph.D. thesis, Massachusetts Institute of Technology, MA, USA, 1999.

[28] 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-lineintrusion detection evaluation". *Proceedings of the 2000 DARPA Information Survivability Conference and Exposition*, volume 2, 12–26. 2000.

[29] Lippmann, Richard P., Joshua W. Haines, David J. Fried, Jonathan Korba, and Kumar Das. "The 1999 DARPA off-line intrusion detection evaluation". *Computer Networks*, 34(4):579–595, 2000.

[30] Ma, Wanli, Dat Tran, and Dharmendra Sharma. "A practical study on shape space and its occupancy in negative selection". *Proceedings of the 2010 IEEE Congress on Evolutionary Computation*, 1–7. IEEE, 2010.

[31] Marhusin, Mohd F., David Cornforth, and Henry Larkin. "An overview of recent advances in intrusion detection". *Proceedings of the 2008 IEEE International Conference on Computer and Information Technology*, 432–437. IEEE, 2008.

[32] McHugh, John. "Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory". *ACM Transactions on Information and System Security*, 3(4):262–294, 2000.

[33] Milton, J. Susan and Jesse C. Arnold. *Introduction to probability and statistics: principles and applications for engineering and the computing sciences*. McGraw-Hill, Inc., 2002.

[34] Moore, Andrew W., Denis Zuev, and Michael L. Crogan. "Discriminators for use in flow-based classification". *Technical report (RR-05-13), University of Cambridge, Computer Laboratory*, 2005.

[35] Nguyen, Hai, K. Franke, and S. Petrovic. "Improving Effectiveness of Intrusion Detection by Correlation Feature Selection". *Availability, Reliability, and Security, 2010. ARES '10 International Conference on*, 17–24. feb. 2010.

[36] Ohm, Paul, Douglas C. Sicker, and Dirk Grunwald. "Legal issues surrounding monitoring during network research". *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, 141–148. ACM, 2007.

[37] Patcha, Animesh and Jung-Min Park. "An overview of anomaly detection techniques: Existing solutions and latest technological trends". *Computer Networks*, 51(12):3448–3470, 2007.

[38] Perelson, Alan S and George F Oster. "Theoretical studies of clonal selection: Minimal antibody repertoire size and reliability of self-non-self discrimination". *Journal of Theoretical Biology*, 81(4):645–670, 1979.

[39] Radack, Shirley. "Intrusion detection and prevention systems". *Information Tehcnology Laboratory Bulletin*, February 2007.

[40] Rexworthy, Ben. "Intrusion detection systems–an outmoded network protection model". *Network Security*, 2009(6):17–19, June 2009.

[41] Scarfone, Karen and Peter Mell. *Guide to intrusion detection and prevention systems (IDPS)*. National Institute of Standards and Technology, Washington, D.C., 2007.

[42] Shapiro, Joseph M. *An Evolutionary Algorithm to Generate Ellipsoid Detectors for Negative Selection*. Master's thesis, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, March 2005.

[43] Shapiro, Joseph M., Gary B. Lamont, and Gilbert L. Peterson. "An evolutionary algorithm to generate hyper-ellipsoid detectors for negative selection". *2005 conference on Genetic and evolutionary computation (GECCO)*, 337–344. ACM, July 2005.

[44] Stibor, Thomas, Jonathan Timmis, and Claudia Eckert. "A comparative study of real-valued negative selection to statistical anomaly detection techniques". *ICARIS 2005*, 262–275. Springer, 2005.

[45] Stibor, Thomas, Jonathan Timmis, and Claudia Eckert. "On the Use of Hypershperes in Artificial Immune Systems as Antibody Recognition Regions". *ICARIS 2006*, 215–228. Springer, 2006.

[46] Tavallaee, Mahbod, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. "A detailed analysis of the kdd cup 99 data set". *Second IEEE Symposium on Computational Intelligence for Security and Defence Applications 2009*, 53–58. 2009.

[47] of the President of the United States, Office. "The Comprehensive National Cybersecurity Initiative", 2010.

[48] United States Air Force. "Cyberspace Operations: Air Force Doctrine Document 3-12", 2010.

[49] United States Air Force Space Command. "The United States Air Force Blueprint for Cyberspace", November 2009.

[50] United States Department of Defense. “The National Military Strategy for Cyberspace operations”, December 2006.

[51] United States Department of Defense. “Department of Defense Strategy for Operating in Cyberspace”, July 2011.

[52] Weisstein, Eric W. “Hypersphere”. http://mathworld.wolfram.com/Hypersphere.html. From MathWorld–A Wolfram Web Resource.

[53] Wu, Shelly X. and Wolfgang Banzhaf. “The use of computational intelligence in intrusion detection systems: A review”. *Applied Soft Computing*, 10(1):1–35, 2010.

[54] Xi, L., F. Zhang, and D. Wang. “Optimization of Real-Valued Self Set for Anomaly Detection Using Gaussian Distribution”. *Artificial Intelligence and Computational Intelligence*, 112–120, 2009.

[55] Yue, Xin, Fengbin Zhang, Liang Xi, and Dawei Wang. “Optimization of self set and detector generation base on real-value negative selection algorithm”. *Proceedings of the 2010 International Conference on Computerand Communication Technologies in Agriculture Engineering*, 12–15. IEEE, 2010.

[56] Zhou, Lin and Feng Jiang. “A rough set based decision tree algorithm and its application in intrusion detection”. *Pattern Recognition and Machine Intelligence*, 6744:333–338, 2011.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From – To)*
22-03-2012	Master's Thesis	08-2010 – 22-03-2012

4. TITLE AND SUBTITLE
Detector Design Considerations in High-Dimensional Artificial Immune Systems

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Bindewald, Jason M., Capt

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
Wright-Patterson AFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GCO/ENG/12-02

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

10. SPONSOR/MONITOR'S ACRONYM(S)

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

12. DISTRIBUTION/AVAILABILITY STATEMENT
Distribution A. Approved for Public Release; Distribution Unlimited

13. SUPPLEMENTARY NOTES
This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT
This research lays the groundwork for a network intrusion detection system that can operate with only knowledge of normal network traffic, using a process known as anomaly detection. Real-valued negative selection (RNS) is a specific anomaly detection algorithm that can be used to perform two-class classification when only one class is available for training. Researchers have shown fundamental problems with the most common detector shape, hyperspheres, in high-dimensional space. The research contained herein shows that the second most common detector type, hypercubes, can also cause problems due to biasing certain features in high dimensions. To address these problems, a new detector shape, the hypersteinmetz solid, is proposed, the goal of which is to provide a tradeoff between the problems plaguing hyperspheres and hypercubes. In order to investigate the potential benefits of the hypersteinmetz solid, an effective RNS detector size range is determined. Then, the relationship between content coverage of a dataset and classification accuracy is investigated. Subsequently, this research shows the tradeoffs that take place in high-dimensional data when hypersteinmetzes are chosen over hyperspheres or hypercubes. The experimental results show that detector shape is the dominant factor toward classification accuracy in high-dimensional RNS.

15. SUBJECT TERMS
Hypersteinmetz, Real-valued negative selection, Artificial Immune Systems, High-dimensions, Anomaly Detection

16. SECURITY CLASSIFICATION OF: UU			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
REPORT	**ABSTRACT**	**c. THIS PAGE**			Dr. Angela Sodemann, Adjunct Professor
U	U	U	UU	131	**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255-3636 x4236 angela.sodemann@afit.edu