On the Appropriateness of Negative Selection for Anomaly Detection and Network Intrusion Detection

Dissertationsschrift

in englischer Sprache
vorgelegt
am Fachbereich Informatik
der Technischen Universität Darmstadt von

Dipl.-Inf. Thomas Stibor

geboren am 7.4.1976 in Schlackenwerth

zur Erlangung des Grades eines
Doktor der Naturwissenschaften (Dr. rer. nat.)

Darmstadt 2006
Hochschulkennziffer D17

Erstreferent:	Prof. Dr. Claudia Eckert
Korreferent:	Prof. Dr. Werner Dilger
Korreferent:	Dr. Jonathan Timmis

Tag der Einreichung:	16.01.2006
Tag der Disputation:	07.03.2006

"*Of course it is very interesting to know how humans can learn. However, this is not necessarily the best way for creating an artificial learning machine. It has been noted that the study of birds flying was not very useful for constructing the airplane*"

Vladimir N. Vapnik [The Nature of Statistical Learning Theory]

Kurzfassung (Deutsch)

Das Immunsystem ist ein komplexes System welches Menschen und Tiere gegen Krankheiten schützt, die durch fremde Eindringlinge wie z.B. Viren, Bakterien und Pilze hervorgerufen werden. Aus Sicht der Informatik scheinen die Erkennungs- und Abwehrmechanismen des Immunsystems neuartige Konzepte und Techniken zur Eindringlingserkennung in Computer-Netzwerke (engl. network intrusion detection) und im Bereich Anomalieerkennung zu bieten. In dieser Arbeit wird das Prinzip der “negativen Selektion” als Paradigma für Eindringlingserkennung in Computer Netzwerke und Anomalieerkennung untersucht. Unter negativer Selektion versteht man in der Immunologie die Zerstörung von unreifen Antikörpern, die körpereigene Antigene erkennen. Antikörper, die die negative Selektion überstehen, sind selbsttolerant und besitzen die Fähigkeit nahezu jegliche fremde Körpersubstanz zu erkennen. Das Immunsystem is also in der Lage mittels der negativen Selektion, “selbst” und “fremd” zu unterscheiden. Abstrahiert man dieses Prinzip und kodiert Antigene als binäre Netzwerk-Pakete oder als reelle n-dimensionale Datenpunkte und Antikörper als binäre Detektoren oder hochdimensionale Kugeln, so erhält man eine immun-inspirierte Technik für o.g. Anwendungsbereiche. Man spricht von künstlichen Immunsystemen, wenn Prinzipien und Abläufe im Immunsystem abstrahiert und zum Problemlösen angewandt werden.

In dieser Arbeit wird untersucht, ob sich die negative Selektion des künstlichen Immunsystems zur Eindringlingserkennung und für Anomalieerkennungs Probleme eignet. Hierfür wird erst die immunlogische negative Selektion beschrieben und anschließend die künstliche Immunsystem negative Selektion dargestellt. Weiterhin wird beschrieben, welche Netzwerk-Informationen notwendig sind, um ein Eindringen zu erkennen. Es zeigt sich, dass die bisherigen Arbeiten auf dem Gebiet die mittels negativer Selektion eine Eindringlingserkennung versuchen, *nicht* für reale Eindringlingserkennung geeignet sind. Ebenfalls wird untersucht, ob eine andere Antikörper-Antigen Kodierungsform, d.h. reelle n-dimensionale Datenpunkte und hochdimensionale Kugeln als Erkennungseinheiten zur Anomalieerkennung geeignet sind.

Auch hier zeigen die erzielten Resultate, insbesondere beim Vergleich zu statistischen Anomalieerkennungs-Methoden, dass diese Kodierungsform in Anwendung mit der negativen Selektion unbefriedigende Ergebnisse liefert.

Als zusammenfassendes Resultat bleibt leider ein negatives Ergebnis zurück, welches zeigt, dass die negative Selektion *nicht* zur Eindringlingserkennung in Computer Netzwerke und Anomalieerkennung geeignet ist.

Abstract (English)

The immune system is a complex system which protects humans and animals against diseases caused by foreign intruders such as viruses, bacteria and fungi. It appears as if the recognition and protection mechanism of the immune system can lead to the development of novel concepts and techniques for detecting intrusions in computer networks, particularly in the area of anomaly detection. In this thesis, the principle of "negative selection" as a paradigm for detecting intrusions in computer networks and anomaly detection is explored. Negative selection is a process of the immune system, which destroys immature antibodies which are capable of recognizing self-antigens. Antibodies which survive the negative selection process are self-tolerant and are capable of recognizing almost any foreign body substance. Roughly speaking one can say that the negative selection endows the immune system with an ability to distinguish between self and non-self. Abstracting the principle of negative selection, the coding antigens as bit-strings which represent network packets or as real-valued n-dimensional points and antibodies as binary detectors or as hyperspheres, one obtains an immune-inspired technique for use in the above mentioned areas of application. We are talking about artificial immune systems, when principles and processes of the immune system are abstracted and applied for solving problems.

In this thesis, we explore the appropriateness of the artificial immune system negative selection for intrusion detection and anomaly detection problems. In the first instance, we describe the immune system negative selection principle, and the subsequent the artificial immune system negative selection principe. We then describe which network information are required to detect an intrusion. Results reveal that previous works that apply the negative selection for this application area, are not appropriate for real-world intrusion detection problems. Moreover we explore if a different antibody-antigen representations, i.e. real-valued n-dimensional points and high-dimensional hyperspheres are appropriate for anomaly detection problems. The results obtained, reveal that negative selection is not appropriate for anomaly detection problems, especially when compared to statistical anomaly detection methods.

In summary, we can unfortunately state that negative selection, is *not* appropriate for network intrusion detection and anomaly detection problems.

Preface

At the beginning of the first PhD year, my research attitude began very optimistically and euphoric, trying to develop an intrusion detection system by means of negative selection for real-world security problems. However time progressed, and the optimism and euphoria changed to disappointment. Of course, it seems to be obvious and intuitive to apply the negative selection for anomaly and intrusion detection problems. However, a closer and deeper view on the negative selection, reveals a lot of *fundamental* problems. To summarize, the negative selection is a very intuitive and attractive immune inspired approach, but it is not appropriate and not applicable for real-world anomaly detection and (network) intrusion detection problems.

Acknowledgments

A lot of friends and researches from all over the world have helped me for achieving the results present in this thesis. First of all, I would like to greatly thank Prof. Dr. Claudia Eckert for giving me the opportunity of doing a PhD and for her continuous support and listening to my problems “the negative selection can never work”. I would also like to thank Prof. Dr. Werner Dilger for being the co-referent. During my PhD years, I had the opportunity to attend quite a few conferences all over world. The places include countries like the England, Canada, Scotland and the United States. At the first attended conference I met Dr. Jon Timmis from whom I learned so much during our fruitful cooperation. Jon invited me for a research visit of 6 months at the University of Kent at Canterbury in England. During those 6 months, most of the research work was done. I will never forget the wonderful time in Canterbury. Finally, my greatest thanks goes to my parents Georg & Tamara Stibor and the rest of our family, for supporting me during my whole academic and non-academic life.

Contents

Chapter 1

Introduction

1.1 Motivation

The task of the immune system is to protect the body against invaders (e.g. viruses, bacteria) which cause diseases, and in the worst case, can lead to death. To recognize the invaders efficiently the immune system utilize many detection techniques which in total makes the immune system to an effective pattern classification system. One outstanding recognition property is the detection of previously unseen invaders, or in other words, the immune system can detect and react to invaders that the body has never encountered before. This outstanding recognition property is achieved by means of the negative selection. Negative selection is one of the earliest proposed immune inspired algorithms and also one of the most frequently used technique in the field of artificial immune systems (AIS). The negative selection algorithm is applied to anomaly detection problems and also to (network) intrusion detection problems. However, negative selection has been criticized in recent times in the field of AIS [41, 18, 28]. It is interesting to note that in published papers dealing with negative selection, the classification performance is never compared to well established and understood techniques from the field of statistics or machine learning. Many comparative studies have compared the performance of negative selection algorithm version 1 to negative selection algorithm version n. It is not the objective of this thesis to find a reason for these studies. However, the author is convinced and hopefully the reader of this thesis will be, that negative selection is a technique that is not well suited and not applicable to real-world classification problems. This hard statement is theoretically and empirically explored in this thesis. It is also interesting to note, that problems with negative selection were (more or less) mentioned [41, 18, 28] previously, but it was never investigated and discussed further in terms of a pure pattern classification problem. In this thesis we explore the negative selection as a (pure) pattern classification technique for anomaly detection and network intrusion detection problems.

1.2 State Of The Art

Historically seen, Forrest et al. [27] were the first who proposed a negative selection algorithm for detecting data manipulations caused by computer viruses. Antibodies and antigens were abstracted as bit-strings and the affinity between both bit-strings were calculated as the number of consecutive matching bits. In Forrest's et al. approach, unmodified files, i.e. files with correct integrity were represented as bit-strings and antibodies were generated by means of negative selection against these bit-strings. Files which were modified by viruses, should be recognized by the generated antibodies. Forrest's et al. proposed negative selection algorithm has several drawbacks which where informally discussed and criticized in [28]. In succeeding works, D'haeseleer et. al [15, 14] proposed several improvements[1] of the negative selection algorithm, however most of the problems still remained. In the year 1996, Hofmeyr started to investigate the negative selection algorithm as a technique for network intrusion detection. Hofmeyr et al. [36, 35] considered self-antigens as normal network traffic and attempted to apply the negative selection algorithm to detect unnormal i.e. malicious network traffic. In Hofmeyr's et al. approach only IP addresses and TCP ports were considered to build a meaningful "self-profile" of the normal network traffic. However this approach was also criticized [41], as the primitive fields[2] were not enough to build a meaningful profile. In the years 2003 and 2004, Esponda et al. [22, 23, 21] published several outstanding papers in the field of negative representations of information and lies the cornerstone for better understanding the representation of data in the complementary space.

Another different negative selection approach were proposed by González et al. [31, 30]. Antibodies were not represented as bit-strings, instead they were represented as hyperspheres. González called this approach, real-valued negative selection. González focused not on intrusion detection problems, but instead on real-valued anomaly detection problems. More concrete, he proposed a real-valued negative selection algorithm, which generates hyperspheres with equal radius lengths. A new unseen point which falls inside a hypersphere is considered as an anomalous point, otherwise as a normal point. In subsequent works Ji et al. [39, 40] proposed a real-valued negative

[1] time and space complexity

[2] IP addresses and TCP ports

selection algorithm with variable-sized radii. They argued that hyperspheres with variable-sized radii possess several advantages, as the complementary space can be covered with fever hyperspheres with variable-sized radii then with fixed-sized radii.

1.3 Outline Of This Thesis

The second chapter of this thesis briefly describes the immune system and the necessary immunological principles, which are abstracted and applied in this thesis. To aid understanding, the core immunological principles and the immunological coherences are simplified. The third chapter, presents the step from an immune system to an artificial immune system, and the artificial immune system framework is illustrated. Furthermore, the generic negative selection is presented and the probabilistic approximation on the number of randomly generated detectors is discussed. In chapter 4, the anomaly detection problem is motivated and two known statistical techniques for comparative studies are described. Furthermore, the concept of an intrusion detection system is motivated and two different network intrusion detection models are presented. In chapter 5, the Hamming negative selection with the r-chunk affinity rule is exhaustively explored and the appropriateness as an network intrusion detection system is discussed. In chapter 6, a real-valued negative selection algorithm for anomaly detection problems is described and studied. In chapter 7, a technique called ROC analysis is motivated and applied for exploring the classification performance of real-valued negative selection when compared to the statistical anomaly detection approaches. The classification performance is explored on a low and high dimensional problem set. In chapter 8, general limitations of real-valued negative selection on high-dimensional problem sets are shown. In chapter 9, the obtained results are summarized and final conclusions are drawn.

Chapter 2

Immune System

The immune system is a fascinating complex system which protects the body against diseases and infections caused by pathogens [38, 54]. Pathogens are foreign substances like viruses, fungi, parasites and bacteria which attack the body continuously, and in the worst case, can lead to death. To detect and eliminate pathogens efficiently, the immune system possesses a multi-layered protection, detection and elimination architecture (see Fig. 2.1). The skin is the first barrier and prevents the physical intrusion of pathogens in the bloodstream. A further barrier is physiological, where conditions such as pH and temperature provide inappropriate living conditions for intruded pathogens. Once pathogens overcome these barriers, the innate or adaptive immune response is triggered. The innate immune system can detect only a limited number of pathogens and is not able to detect unknown or unseen pathogens. However, the reaction and elimination process is very fast compared with adaptive immune response. This is accomplished by phagocytes, which are able to ingest and to destroy pathogens via a process known as

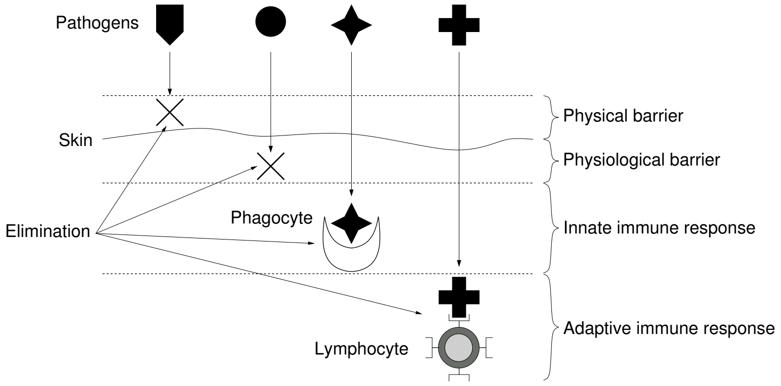


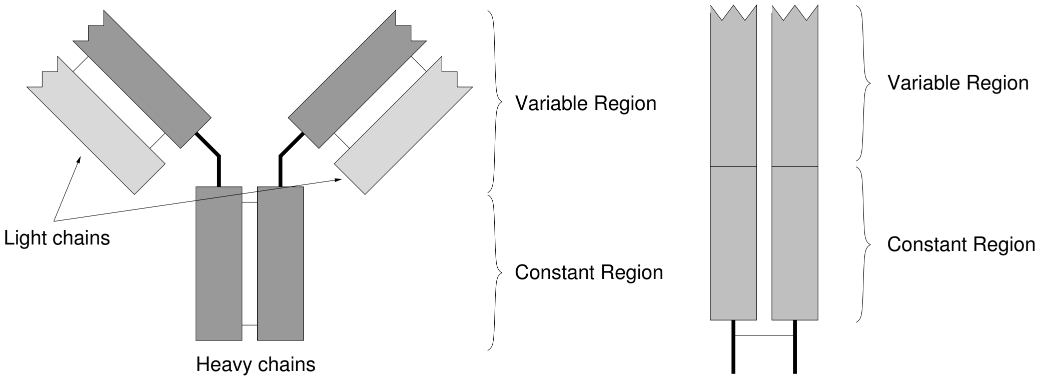
Figure 2.1: Multi-layered protection and elimination architecture

phagocytosis. The adaptive immune system is not able to react in such a fast manner. Instead, it is able to detect and eliminate pathogens, never encountered before. This capability also encompasses pathogens which are synthetically produced by chemical laboratories and would never appear in the nature. To detect and eliminate pathogens of such diversity, the adaptive immune system contains certain types of white blood cells called lymphocytes. Lymphocytes do not recognize whole complete pathogens, instead they recognize pathogenic patterns, called *antigens*. Antigens are any substances built out of amino acids which induce an innate or adaptive immune responds. To recognize antigens, lymphocytes carry recognition units (called *antibodies*) on their surface which have the capability to recognize and classify proteins[1], which belong to the body (called self) and foreign substances (called non-self). Lymphocytes are subdivided in two different classes: B and T-Lymphocytes. B-Lymphocytes mature in the bone marrow and manufacture soluble antibodies. T-Lymphocytes mature in the thymus and do not produce soluble antibodies, instead this class support B-Lymphocytes in their detection and elimination process and recognize antigens which are associated with a host cell.

2.1 B- and T-Lymphocytes

B-Lymphocytes search in the extracellular spaces, such as the bloodstream, for pathogens and produce antibodies against antigens unless they are stimulated by (helper) T-Lymphocytes. T-Lymphocytes are divided in two subclasses. The first subclass called helper T-Lymphocytes support B-Lymphocytes in their detection and activation process with proteins called lymphokines. B-Lymphocytes which recognize an antigen are not able to cause an immune response. In addition, they require an activation signal (the lymphokines) from the helper T-lymphocytes which also recognizes these antigen. The second subclass, called killer T-Lymphocytes, recognize infected cells (intracellular infections) and produce cytotoxic substances which kill directly the infected cells. T-Lymphocytes are able to recognize such intracellular infections by means of MHC molecules. A major histocompatibility complex (MHC) is a cellular molecule which is found in every host cell. More precisely, there are two classes of MHC molecules, called MHC-I and MHC-II. MHC-I molecules are found in all host cells, where MHC-II molecules are found in antigen presenting cells called APC.

[1] organic compound that consists of amino acids



(a) BCR antibody consists of two heavy and two light chains. Each chain consists of a variable and a constant region

(b) TCR antibody consists of a variable and a constant region

Figure 2.2: B-Lymphocyte antibody (BCR) and T-Lymphocyte antibody (TCR)

2.2 Antibody Diversity

An antibody is, from the chemical point of view, a protein and recognizes a specific antigen unique to its target. A B-Lymphocyte antibody (termed BCR) has a Y-shaped form consisting of two heavy and two light chains (see Fig. 2.2(a)). The heavy chains are so called, because they weigh more than the light chains. The heavy chain consists of approximately 450-500 amino acids, the light chain of approximately 220 amino acids. Each chain consists of a constant and a variable region. The constant region consists of an amino acid sequence which is essentially the same for all antibodies in the body and encodes information, *how* the antigen should be eliminated, once it is bound. The variable region determines the specificity[2] of the antibody and consists of an amino acid sequence which is the same for all antibodies of a B-Lymphocyte, but will vary from one B-Lymphocyte to another. The heavy chain of the BCR is built out of three gene segments : V (variable), D (diversity) and J (joining) and the light chain only of the V and J segment. Each gene segment is composed again of certain gene combinations. The total number of possible BCR protein encodings is calculated as the product of the number of possible gene combinations for each segment. In human body this results in approximately $3,4 \cdot 10^6$ different BCR protein encodings [38].

A T-Lymphocyte antibody (termed TCR) is likewise built out of three gene segments (V-D-J), but has a non Y-shaped form (see Fig. 2.2(b)). The

[2] binding affinity to the antigen

specificity of the variable region is generated similar to the specificity of the BCR, but with deviations in the number of possible gene combinations [38]. The total number of TCR protein encodings in human body results in $5,8 \cdot 10^6$ possibilities [38].

The actual number of antigens that the immune system can recognize with antibodies is far greater than 10^6. It has been estimated to be greater than 10^{16} [38]. An interesting question is : how can the immune system recognize such a high number of antigens with a far lower number of BCR and TCR encodings ? The answer lies in the junctional diversity and somatic hypermutation. The junctional diversity causes additional diversity by courtesy of adding and removing N-Nucleotides[3] during the assembly of the gene segments. The somatic hypermutation causes a high level of mutations in the variable regions of the gene segments.

2.3 Negative Selection

Lymphocytes are able to recognize (foreign) antigens but also cells and molecules that belong to the body (called self-antigens). These self-reactive lymphocytes cause autoimmune diseases and can lead to death in the worst case. Negative selection is a process that eliminates self-reactive lymphocytes. Self-reactive lymphocytes can occur, because the BCRs and TCRs are randomly composed from different gene segments and undergo a junctional diversity and somatic hypermutation process. This (random) process can therefore produce lymphocytes which are able to recognize self-antigens. The negative selection of T-Lymphocytes occurs within the thymus. The thymus forms a highly impermeable barrier to macromolecules called blood-thymic barrier. The blood-thymic barrier allows Thymocytes (immature T-Lymphocytes) to mature and undergo selection in an environment protected from contact with foreign antigens. During the selection process, APCs present self-peptide/MHC complexes to the T-Lymphocytes. T-Lymphocytes that react strongly (bind with high affinity) with the self-peptide/MHC complexes are eliminated through a controlled cell death (called apoptosis). As a result, only those T-Lymphocytes remain which can recognize foreign antigens and are not self-reactive (see Fig. 2.3).

[3] N-Nucleotide regions are encoded by a special enzym (termed TdT), instead of the V-D-J gene segments

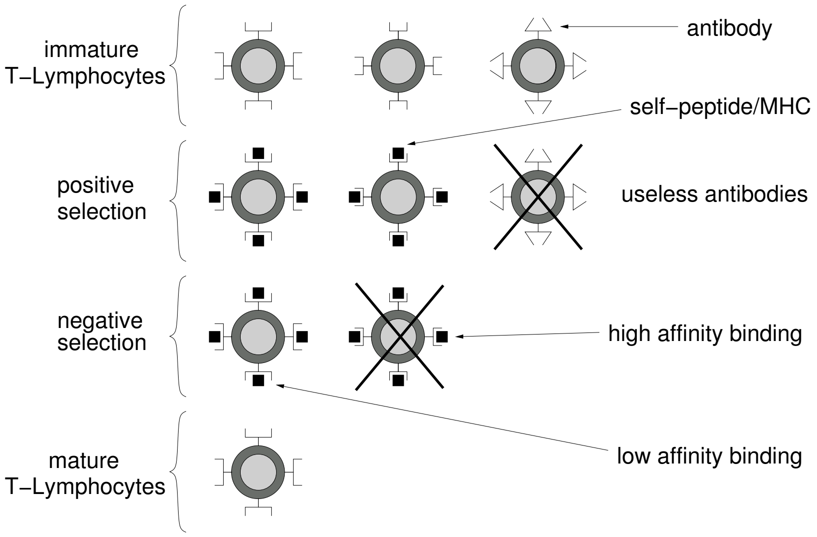


Figure 2.3: Positive and negative selection process

2.4 Positive Selection

T-Lymphocytes not only undergo a negative, but also a positive selection process. T-Lymphocytes which carry “useless” antibodies are not useful in the recognition process. The positive selection rescues those T-Lymphocytes from apoptosis that carry convenient antibodies and react *weakly* with self-peptide/MHC complexes (see Fig. 2.3). Experiments and observations reveal that most of the immature T-Lymphocytes die in the thymus and only 2 % – 5 % survive and leave the thymus as mature T-Lymphocytes.

An open question currently in Immunology is : how exactly is the weak and strong affinity expressed [59] ? If the binding strength is similar in the negative and positive selection, then *no* T-Lymphocytes would survive these two selection processes. The positive selection would keep T-Lymphocytes with a certain affinity and the negative selection would eliminate then these T-Lymphocytes.

It is out of the scope of this thesis, and also the author’s lack of knowledge to answer this immunological question. Nevertheless, the latest immunological hypothesis supports the weak-strong affinity theory [59].

2.5 Summary

The main task of the immune system is to defend the body against diseases caused by pathogens. To detect and eliminate pathogens, the immune system contains certain types of white blood cells called lymphocytes, which can recognize pathogenic patterns, called antigens. Lymphocytes can be thought of as detectors, as they carry recognition units (termed antibodies) on their surface which have the capability to recognize and classify proteins, which are produced inside the host (termed self) and outside the host (termed non-self). To avoid a misclassification of self proteins by lymphocytes, the immune system eliminates self reactive lymphocytes in a censoring process called *negative selection*. After this censoring process, the immune system contains lymphocytes which recognize non-self proteins. Since the amount of lymphocytes at any given time is limited, lymphocytes which are not involved in a recognition process (stimulated) are removed by a mechanism called apoptosis (cell death) and new lymphocytes are added by the immune system. This continual turnover of new and old lymphocytes enables the immune system to recognize all possible non-self proteins over time with a limited number of antibodies. More precisely, the immune system disposes about 10^6 different proteins which are randomly composed from different gene segments. This random composition, together with the junctional diversity and the somatic hypermutation, achieves a potential repertoire greater than 10^{16}. As explained above, the negative selection process eliminates self reactive lymphocytes. In addition, the immune system also performs *positive selection*. A cell which is infected with a virus is not directly detectable by antibodies, because the cell carries no binding information on their surface. To solve this problem, all cells contain MHC molecules which are able to present intruded viral peptides on the cell surface. The MHC presented information consists of non-self peptides, but also of self peptides. The process of positive selection ensures that those lymphocytes are selected, whose antibodies are capable of recognizing and binding weakly with self-peptide/MHC complexes associated with non-self peptides.

Roughly speaking, one can say that the negative selection allows the immune system to recognize *non-self* and the positive selection allows the immune system to recognize *self*.

Chapter 3

Artificial Immune Systems

An artificial immune system (AIS) is a paradigm inspired by the immune system and is used for solving computational and information processing problems. A widely accepted definition of an AIS is by de Castro and Timmis [13]:

Definition 3.1. *Artificial immune systems are adaptive systems, inspired by theoretical immunology and observed immune functions, principles and models, which are applied to problem solving.*

An AIS can be described and developed using a framework (see Fig. 3.1) which contains the following basic elements:

- A representation for the artificial immune elements.
- A set of functions, which quantifies the interactions of the artificial immune elements.
- A set of algorithms which is based on observed immune principles and methods.

Immune elements are e.g. antibodies, antigens or MHC molecules which represent (encode) solutions for problems. The quality[1] of the represented solutions is quantified by functions. Typical functions are metric functions (distance measurements), which quantify the interactions between antibodies and antigens. Immune algorithms are motivated by the observed immunological principles, e.g. the negative, positive or the clonal selection principle and are abstracted as algorithms for solving problems.

This 3-step formalization (representation, affinity, algorithm) for using the AIS framework is explained in the following sections.

[1] Fitness values in genetic algorithms

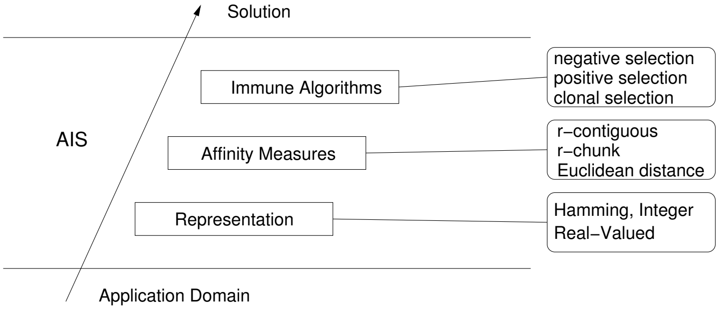


Figure 3.1: The AIS framework proposed by de Castro and Timmis

3.1 Shape-Space Formalism

The notion of *shape-space* was introduced by Perelson and Oster [50] and allows a quantitative affinity description between antibodies and antigens. More precisely, a shape-space is a metric space with an associated distance (affinity) function. All artificial immune elements, like antibodies and antigens, are represented in a shape-space. The affinity can be described with a multitude number of distance functions — a detailed overview is provided in [13]. In this work, we consider two different shape-spaces and affinity functions only, because the investigated immune algorithms operate on these.

3.1.1 Hamming Shape-Space and Matching Rules

The Hamming shape-space U_l^Σ is built out of all elements of length l over a finite alphabet Σ.

Example 3.1.

$$\begin{array}{cc}
\Sigma = \{0,1\} & \Sigma = \{A,C,G,T\} \\
 & \\
000\ldots000 & AAA\ldots AAA \\
000\ldots001 & AAA\ldots AAC \\
\ldots\ldots\ldots\ldots & \ldots\ldots\ldots\ldots \\
\ldots\ldots\ldots\ldots & \ldots\ldots\ldots\ldots \\
\underbrace{111\ldots111}_{l} & \underbrace{TTT\ldots TTT}_{l}
\end{array}$$

In example 3.1 two Hamming shape-spaces for different alphabets and alphabet sizes are presented. At the left side, a hamming shape-space defined over the binary alphabet with elements of length l is shown. At the right side, a hamming shape-space defined over the DNA bases alphabet (Adenine, Cytosine, Guanine, Thymine) is presented. A formal description of antigen-antibody interactions not only requires an encoding, but also appropriate affinity functions. Percus et. al [49] proposed the *r-contiguous* matching rule for abstracting the affinity of an antibody needed to recognize an antigen.

Definition 3.2. *An element $e \in U_l^\Sigma$ with $e = e_1e_2\dots e_l$ and detector $d \in U_l^\Sigma$ with $d = d_1d_2\dots d_l$, match with r-contiguous rule, if a position p exists where $e_i = d_i$ for $i = p,\dots,p+r-1$, $p \leq l-r+1$.*

Informally, two elements, with the *same length*, match if at least r contiguous characters are identical.

Example 3.2.

$$\Sigma = \{0,1\} \qquad \begin{matrix} \overbrace{0\;\;1\;\;\mathbf{1}\;\;\mathbf{0}\;\;\mathbf{1}\;\;0\;\;1}^{l} & \textit{element} \\ 0\;\;0\;\;\underbrace{\mathbf{1}\;\;\mathbf{0}\;\;\mathbf{1}}_{r}\;\;1\;\;0 & \textit{detector} \end{matrix}$$

Example 3.2 shows an element and a detector of length $l = 7$. The detector recognizes the element for $r \leq 3$ with the r-contiguous matching rule. This example shows one possible detector. The number of all possible detectors is 2^4 and can be characterized with the generic r-contiguous detector

$$* \quad * \quad 1 \;\; 0 \;\; 1 \quad * \quad *$$

where $*$ the asterisk represents either a 1 or 0.

An additional rule, which subsumes[2] the r-contiguous rule is the r-chunk matching rule [4].

Definition 3.3. *An element $e \in U_l^\Sigma$ with $e = e_1e_2\dots e_l$ and detector $d \in \mathbb{N} \times D_r^\Sigma$ with $d = (p \mid d_1d_2\dots d_r)$, for $r \leq l$, $p \leq l-r+1$ match with r-chunk rule, if a position p exists where $e_i = d_i$ for $i = p,\dots,p+r-1$.*

Informally, element e and detector d match if a position p exists, where all characters of e and d are identical over a sequence of length r.

[2] include within a larger entity

Example 3.3.

$$\Sigma = \{0,1\} \qquad \overbrace{0\ \ 1\ \ \underbrace{1\ \ 0\ \ 1}_{r}\ \ 0\ \ 1}^{l} \qquad \textit{element}$$

$$\downarrow$$

$$\textit{detectors} \quad \{0|011,\ 1|110,\ 2|101,\ 3|010,\ 4|101\}$$

Example 3.3 shows the same element as in example 3.2 and all possible generable r-chunk detectors. As $l = 7$ and $r = 3$, one obtains $l - r + 1 = 5$ possible r-chunk detectors. The arrow points to an r-chunk detector of position $p = 2$.

We use the term *subsume* as any r-contiguous detector can be represented as a set of r-chunk detectors. Consider element 0110101 in example 3.3 as an r-contiguous detector of length $l = 7$. This r-contiguous detector can be represented as five r-chunk detectors (see example 3.3). This implicates that any set of elements from U_l^Σ that can be recognized with a set of r-contiguous detectors can also be recognized with some set of r-chunk detectors. The converse statement is surprisingly *not* true, i.e. there exists a set of elements from U_l^Σ that can be recognized with a set of r-chunk detectors, but *not* recognized with any set of r-contiguous detectors. We demonstrate this converse statement on an example, a formal approach is provided in [23].

Example 3.4. *Given a Hamming shape-space* $U_5^{\{0,1\}}$*, a set* $S = \{01011, 01100, 01110, 10010, 10100, 11100\}$ *of self elements and a detector length* $r = 3$.

All possible generable r-contiguous detectors for the complementary space $U_5^{\{0,1\}} \setminus S$ *are* $D_{r-contiguous} = \{00000, 00001, 00111, 11000, 11001\}$.

All possible generable r-chunk detectors are
$D_{r-chunk} = \{0|000, 0|001, 0|110, 1|000, 1|011, 1|100, 2|000, 2|001, 2|101, 2|111\}$.

The set $D_{r-contiguous}$ *recognizes the elements*
$\mathcal{P}_1 = U_5^{\{0,1\}} \setminus (S \cup \{01010, 01101, 10011, 10101, 11101, 11110\})$,
whereas the set $D_{r-chunk}$ *recognizes the elements*
$\mathcal{P}_2 = U_5^{\{0,1\}} \setminus (S \cup \{10011, 01010, 11110\})$. *Hence* $|\mathcal{P}_1| \leq |\mathcal{P}_2|$.

Example 3.4 shows, that the set of r-chunk detectors $D_{r-chunk}$ recognizes more elements of $U_5^{\{0,1\}}$ than then the set of r-contiguous detectors $D_{r-contiguous}$ and therefore the r-chunk matching rule subsumes the r-contiguous rule.

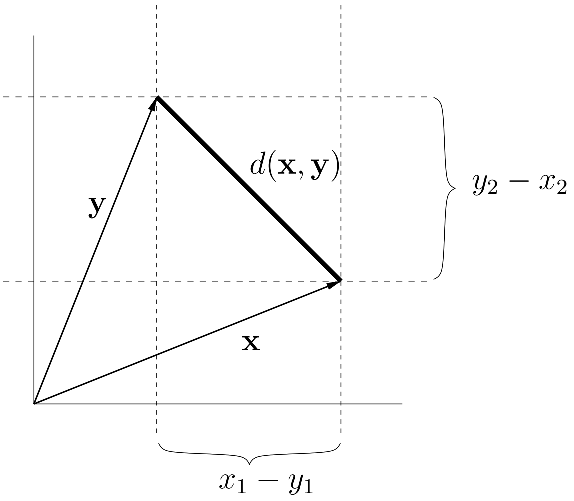


Figure 3.2: Euclidean distance between the vectors $\mathbf{x} = (x_1, x_2)$ and $\mathbf{y} = (y_1, y_2)$

Another shape-space proposed by theoretical immunologists [50, 51] is the real-valued shape-space also referred to as generalized shape. From a physical point of view, the binding between antibody and antigen involves short-range noncovalent interactions based on electrostatic charge, hydrogen binding and van der Waals interactions. This physical binding properties can be formulated as a geometric quantity[3] in real-valued shape-space.

3.1.2 Real-Valued Shape-Space and Euclidean Distance

Real-valued shape-space is a n-dimensional Euclidean space $\mathbb{R}^n$, where every element is represented as a n-dimensional point, or simply as a vector represented by a list of n real numbers. The Euclidean distance[4] d, is the (standard) distance between any two vectors $\mathbf{x}, \mathbf{y}$ in $\mathbb{R}^n$ (see Fig. 3.2) and is defined as :

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1)^2 + \ldots + (x_n - y_n)^2}$$

An antibody and an antigen is represented as a n-dimensional point. The interaction is quantified by means of Euclidean distance, i.e. the smaller the distance, the higher the affinity between antibody and antigen.

[3] n-dimensional point

[4] also termed 2-norm

3.2 Generic Negative Selection Algorithm

Negative selection — as mentioned in section 2.3 — is a mechanism of the immune system to protect the body against self reactive lymphocytes. As a result, only those lymphocytes survive this selection process, that do not bind strongly with self-antigens. This principle inspired Forrest et al. [27] to propose a generic negative selection algorithm for detecting data anomalies, and in later works, this principle was applied for detecting (network) intrusions [1, 35, 30, 5, 58, 72] and also for detection anomalies in time series [12]. The basic idea is to generate a number of detectors in the complementary space[5], and then to apply these detectors to classify new (unseen) data as self (no data manipulation) or non-self (data manipulation). For this purpose, the whole shape-space U is divided in a self set S and a non-self set N with

$$U = S \cup N \quad and \quad S \cap N = \emptyset.$$

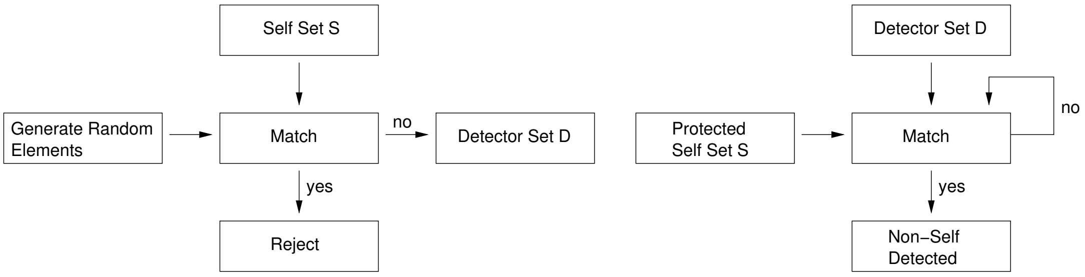
The generic negative selection algorithm proposed by Forrest et al. is illustrated in figure 3.3 and summarized in the following steps :

```
Algorithm 1: Generic Negative Selection Algorithm
  input  : $S$ = set of self elements
  output: $D$ = set of generated detectors
  begin
      1. Define self as a set $S$ of elements in shape-space $U$
      2. Generate a set $D$ of detectors, such that each fails to match any
      element in $S$
      3. Monitor $S$ for changes by continually matching the detectors in
      $D$ against $S$
  end
```

The first version of the negative selection algorithm [27] mirrored closely the generation of T-Lymphocytes in the immune system. Candidate detectors[6] were drawn at random from U_l^2 and checked against all elements in S. This process of random generation and checking was repeated until the required number of detectors was generated. Freitas and Timmis [28] noted several weaknesses of the detector generation and problems of this anomaly detection technique in general. To summarize, the algorithm is inefficient, since a vast number of randomly generated detectors need to be discarded, before the required number of suitable ones are obtained — this is a simple

[5] space which contains no seen self elements

[6] r-contiguous



(a) Generation of Detector Set

(b) Monitor Protected Strings for Manipulation

Figure 3.3: Generic Negative Selection Algorithm proposed by Forrest et al.

random search. And second, this algorithm [27] and later proposed algorithms [15, 71, 2, 60] are defined over the Hamming shape-space associated with affinity matching functions which induces additional “positional bias” problems. This second problem, is investigated and discussed in detail in chapter 5.

Subsequently, we summarize and present results of the first version of the negative selection algorithm [27] — the random search method.

Forrest’s et al. [27] theoretical analysis on the number of randomly drawn detectors based on a mathematical approximation proposed by Percus et al. [49] which is unfortunely incorrect for $r \leq l$. Percus et al. approximate the probability P_S that a random detector recognizes[7] a random antigen with

$$P_S = m^{-r}\left[(l-r)(m-1)/m+1\right] \tag{3.1}$$

where m is the alphabet size[8] and l, r the parameters from definition 3.2. Percus et al. mentioned that this approximation is valid when $m^{-r} \ll 1$. In a succeeding work, Wierzchoń [70] has shown that approximation 3.1 is only valid when $r \geq l/2$.

The correct probability approximation[9] for $r \leq l$ and alphabet size 2 was originally derived by William Feller and is presented in his textbook [25]. To approximate the probability that a random detector recognizes with r-contiguous matching rule a random antigen is formally defined by Feller as follows :

“*A sequence of n letters S and F contains as many S-runs of length r as*

[7] with r-contiguous matching rule

[8] in our notation $|\Sigma|$

[9] also presented in Ranang master thesis

there are non-overlapping uninterrupted blocks containing exactly r letters S each".

Given a Bernoulli trial with outcomes S (success) and F (failure), the probability of no success run of length r in l trials is, according to Feller

$$\frac{1-px}{(r+1-rx)q} \cdot \frac{1}{x^{l+1}} \tag{3.2}$$

where

$$p = q = \frac{1}{2} \quad \text{and} \quad x = 1 + qp^r + (r+1)(qp^r)^2 + \ldots$$

as term 3.2 gives the probability of no success run of length r in l trials, the correct approximation that a random detector recognizes with r-contiguous matching rule a random antigen results in

$$P_{WF} = 1 - \left(\frac{1-px}{(r+1-rx)q} \cdot \frac{1}{x^{l+1}} \right) \tag{3.3}$$

We like to emphasize here that the link between r-contiguous matching rule and term 3.3 was first demonstrated by Ranang [52] with a simple counter-example. Ranang shows that for $m = 2, l = 49$ and $r = 4$ the approximation 3.1 results in

$$P_S = 2^{-4} \left[\frac{(49-4)(2-1)}{2} + 1 \right] = 1.46875$$

which is greater than 1 and therefore does *not* describe a correct probability distribution.

Verifying term 3.3 for $l = 49$ and $r = 4$ results in $P_{WF} \approx 0.82$. The difference between term 3.1 and 3.3 for small values of r are extremely large (see Fig. 3.4). Reaching a certain value for r, both terms adjust and decrease asymptotically to 0 — probably this is the reason that nobody, except Ranang, noticed this incorrect approximation.

3.2.1 Probability of Detection for Random Detector Generation

In this section we summarize results on the number of required r-contiguous detectors when generated randomly shown in [27]. We replace the wrong probability approximation (P_S) — used in [27] — with the correct one (P_{WF}) and discuss the obtained results.

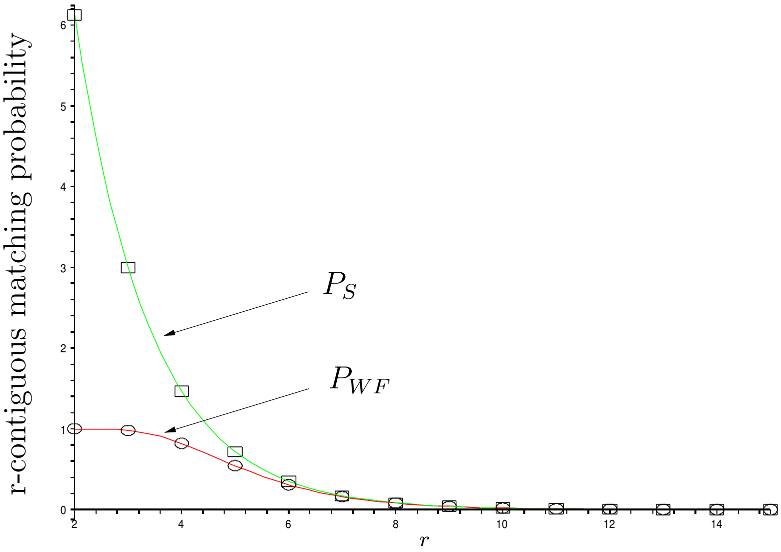


Figure 3.4: Difference between Percus et al. and Feller's (probability) approximation for $l = 49$ and $r = \{2, \ldots, 15\}$

Forrest et al. [27] proposed a straightforward calculation on the number of required r-contiguous detectors by allowing with probability P_{fail} that the randomly generated detectors fail to detect a data manipulation.

Let U_l^2 be a Hamming shape-space, $\{D_0, D, S\} \subseteq U_l^2$ and

$$\begin{aligned}
|D_0| &= \text{number of initial detector (before negative selection)}\\
|D| &= \text{number of detector (after negative selection)}\\
|S| &= \text{number of self elements in } S\\
P_{WF} &= \text{probability according to Feller's approximation (3.3)}\\
P_{\neg S} &= \text{probability of a random element from } U_l^2 \text{ not matching any}\\
&\ \text{element from } S\\
&= (1 - P_{WF})^{|S|} \approx e^{-P_{WF} \cdot |S|}\\
P_{fail} &= \text{probability that } |D| \text{ detectors fail to detect a data manipulation}\\
&= (1 - P_{WF})^{|D|} \approx e^{-P_{WF} \cdot |D|}
\end{aligned}$$

Given a pre-defined number of randomly drawn initial detectors $|D_0|$, $|S|$ and P_{WF}, one obtains the number of suitable detectors $|D|$ not matching any element in S

$$|D| \quad = \quad |D_0| \cdot P_{\neg S} \tag{3.4}$$

The number of detectors $|D|$ that fails to detect a data manipulation with probability P_{fail} is

$$|D| \quad = \quad -\frac{\ln(P_{fail})}{P_{WF}} \tag{3.5}$$

Combining (3.4) and (3.5)

$$|D_0| \quad = \quad \frac{-\ln(P_{fail})}{P_{WF} \cdot P_{\neg S}} \tag{3.6}$$

one obtains the number of initial detectors $|D_0|$. That means, for detecting a data manipulation by allowing with probability P_{fail} that $|D|$ detectors fail to detect a data manipulation, one requires a size of $|D_0|$ initial detectors.

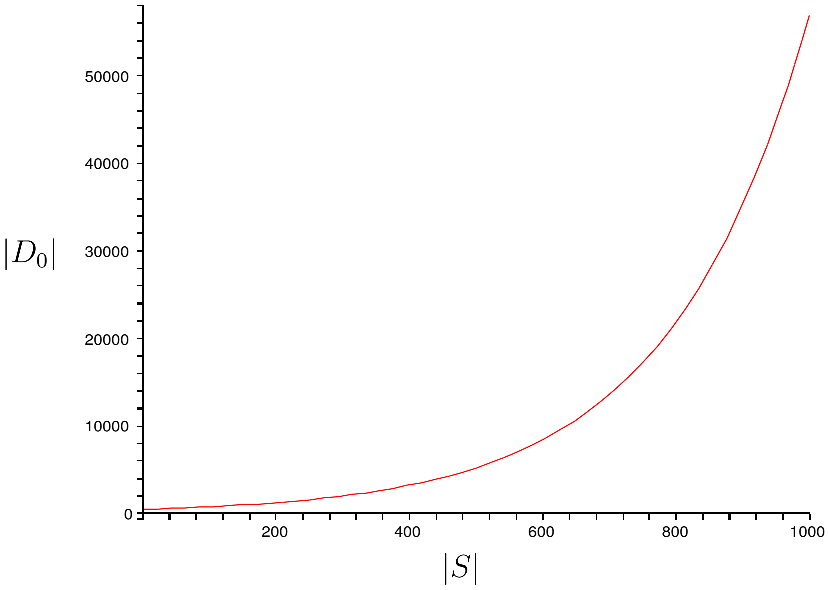


Figure 3.5: Coherence between the number of self elements $|S|$ and the number of initial detectors $|D_0|$ according to $-\ln(P_{fail})/(P_{WF} \cdot P_{\neg S})$ for $l = 49, r = 12$ and $P_{fail} = 0.1$

In figure 3.5 one can see the exponential grow of term 3.6 with respect to the number of self elements. The value $P_{fail} = 0.1$ is chosen, as in [27], the values $l = 49$ and $r = 12$ are chosen as proposed in [35, 5] — the chosen parameters are explained in section 5.9 and become clear in the context of network intrusion detection.

By transforming term 3.6 in $(e^{P_{WF} \cdot |S|} \cdot (-\ln(P_{fail})/P_{WF}))$ it appears that term 3.6 grows exponentially in $|S|$, i.e. this random search approach becomes infeasible for large $|S|$.

3.3 Summary

An artificial immune system is a paradigm inspired by the immune system and is used for solving computational and information processing problems. Mistakenly an artificial immune system is often only considered — outside the artificial immune system community — as an approach for solving security problems, like viruses or worms attacks, as it seems to be perfectly related to such problems. In this chapter we have shown the 3-step formalization (representation, affinity, algorithm) approach proposed by de Castro and Timmis. The approach requires a proper immune element representation, suitable affinity functions and immune algorithms. In artificial immune systems, the Hamming shape-space and the real-valued shape-space is used. For the Hamming shape-space the binding strength (affinity) is abstracted by means of the r-contiguous and r-chunk matching rules. For the real-valued shape-space, the Euclidean distance is applied. Furthermore the generic negative selection proposed by Forrest et al. is presented and also the probabilistic results on the number of finding suitable detectors when generated randomly. The detector generation process in the generic negative selection was strongly criticized by Freitas and Timmis as it is a simple random search method. Moreover, Ranang has shown with a simple counter-example that the approximation proposed by Percus et al. is incorrect. We have revised the probabilistic results with the correct approximation and have shown the infeasible time complexity of the random detector generation process.

Chapter 4

Anomaly Detection and Network Intrusion Detection

In this chapter, we provide an overview of anomaly detection and present established statistical anomaly detection techniques. This techniques will be later used for a comparative study to the real-valued negative selection (chapter 7). Additionally, we describe two network intrusion detection models here, for later discussing the appropriateness and applicability of Hamming negative selection for these models (chapter 5).

Anomaly detection, also referred to as novelty detection [45], outlier detection [45] or one-class learning is a pattern classification problem. The goal of (supervised) pattern classification, also referred to as pattern recognition, is to find a functional mapping between input data X to a class label Y so that $Y = f(X)$. The mapping function is the pattern classification algorithm which is trained (or learned) with a given number of labeled data called *training data*. The aim is to find the mapping function, which gives the smallest possible error in the mapping, i.e. the minimum number of samples where Y is the wrong label, especially for *test data* not seen by the algorithm during the learning phase. In the simplest case there are only two different classes $\mathcal{C}_0, \mathcal{C}_1$ and the task is to estimate a function $f : \mathbb{R}^N \rightarrow \{\mathcal{C}_0, \mathcal{C}_1\}$, using training data pairs generated i.i.d.[1] according to an unknown probability distribution

$$P(\mathbf{x}, y) := (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\mathbf{n}, y_n) \in \mathbb{R}^N \times Y, \quad Y \in \{\mathcal{C}_0, \mathcal{C}_1\}$$

such that f will correctly classify unseen samples $\mathbf{x}$. When the training data consists *only* of samples from a single class $(\mathbf{x}, y \in \mathcal{C}_0)$ or a single class and a strongly under-represented second class $(\mathbf{x}, y \in \{\mathcal{C}_0, \mathcal{C}_1\})$ where $|\mathcal{C}_0| \gg |\mathcal{C}_1|$

[1] independently drawn and identically distributed

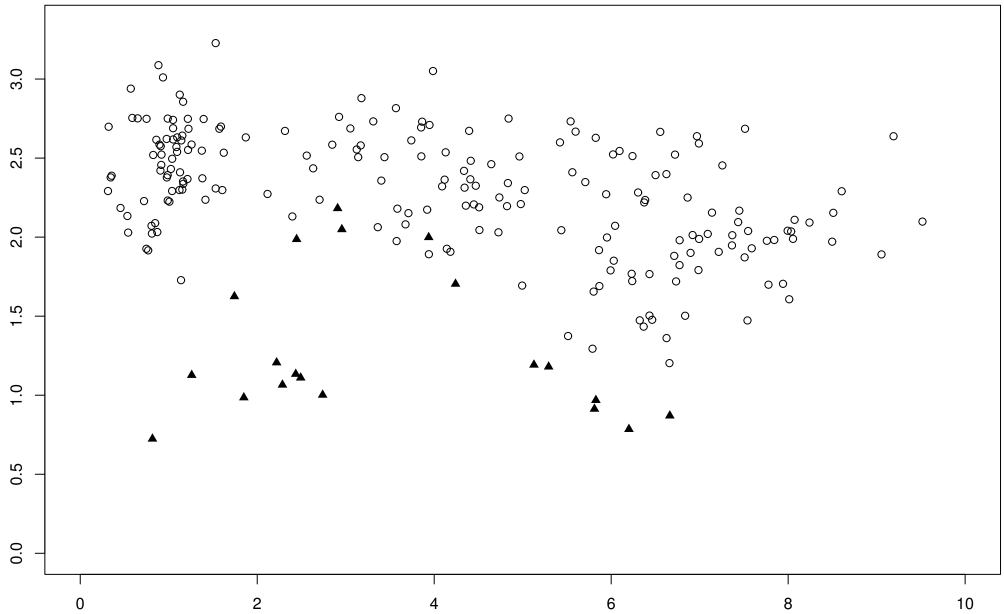
and the test data contains samples from two or more classes, the classification task is called *anomaly detection.* Examples of anomaly detection are machine fault recognition or medical diagnosis, where only training data containing normal behavior is available, as it is difficult or impossible to obtain abnormal behavior. In a probabilistic sense, anomaly detection is equivalent to deciding whether an unknown test sample is produced by the underlying probability distribution that corresponds to the training set of normal examples. Such approaches are based on the assumption that *anomalous* data are not generated by the source of *normal* data (see Fig. 4.1). Before we start to introduce the statistical novelty detection techniques, we show a dilemma which arises in any pattern classification techniques. The dilemma is termed *overfitting/underfitting* and can occur as a result of wrong function flexibility and a limited number of seen training samples. A sufficiently flexible mapping function can always perfectly fit the training data, i.e. the function completely adapts to all available training samples (see Fig. 4.2(a)). However, this results in a poor generalization as unseen samples are not sufficiently integrated in the mapping function. On the other hand, the opposite effect can occur, i.e. the mapping function has limited flexibility to capture all characterists in the data (see Fig. 4.2(b)). The best way to avoid overfitting/underfitting effects is to use a sufficiently large number of training data, as only a large number of samples reveal the true distribution more closely (see Fig. 4.2(d)) and to control the flexibility of the mapping function.

We present this dilemma as we attempt to explain the overfitting/underfitting effects in Hamming negative selection (see chapter 5).

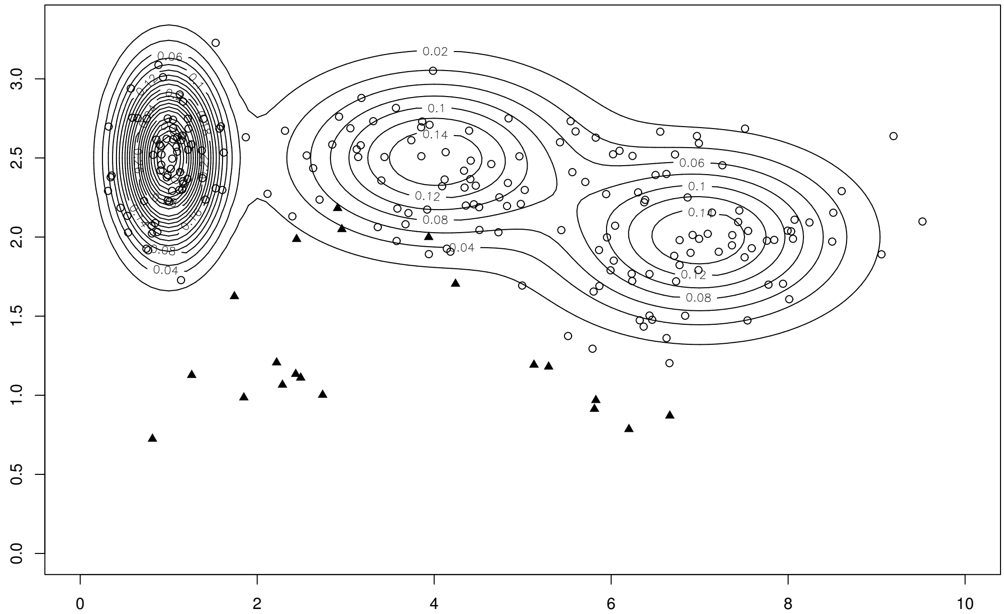
4.1 Statistical Novelty Detection Techniques

Through the application of statistical methods, novelty can be quantified as a deviation from a probability distribution $p(\mathbf{x})$ which is generated from normal data. The quantity can be expressed by a threshold, where (unseen) data samples for which $p(\mathbf{x})$ falls below this threshold, are considered as abnormal samples. By applying such a threshold, all new data samples can be classified into two classes $\mathcal{C}_0$ or $\mathcal{C}_1$, where the training data are assumed to be drawn entirely from $\mathcal{C}_0$. To minimize the probability of misclassification, a new data sample $\mathbf{x}$ is assigned to the class with the larger posterior probability [17]. This classification decision is based on the Bayes theorem and can be written as :

$$\text{Decide} \quad \mathcal{C}_0 \quad if \quad p(\mathbf{x}|\mathcal{C}_0) > \frac{p(\mathbf{x}|\mathcal{C}_1)P(\mathcal{C}_1)}{P(\mathcal{C}_0)}; \quad \text{otherwise} \quad decide \quad \mathcal{C}_1$$

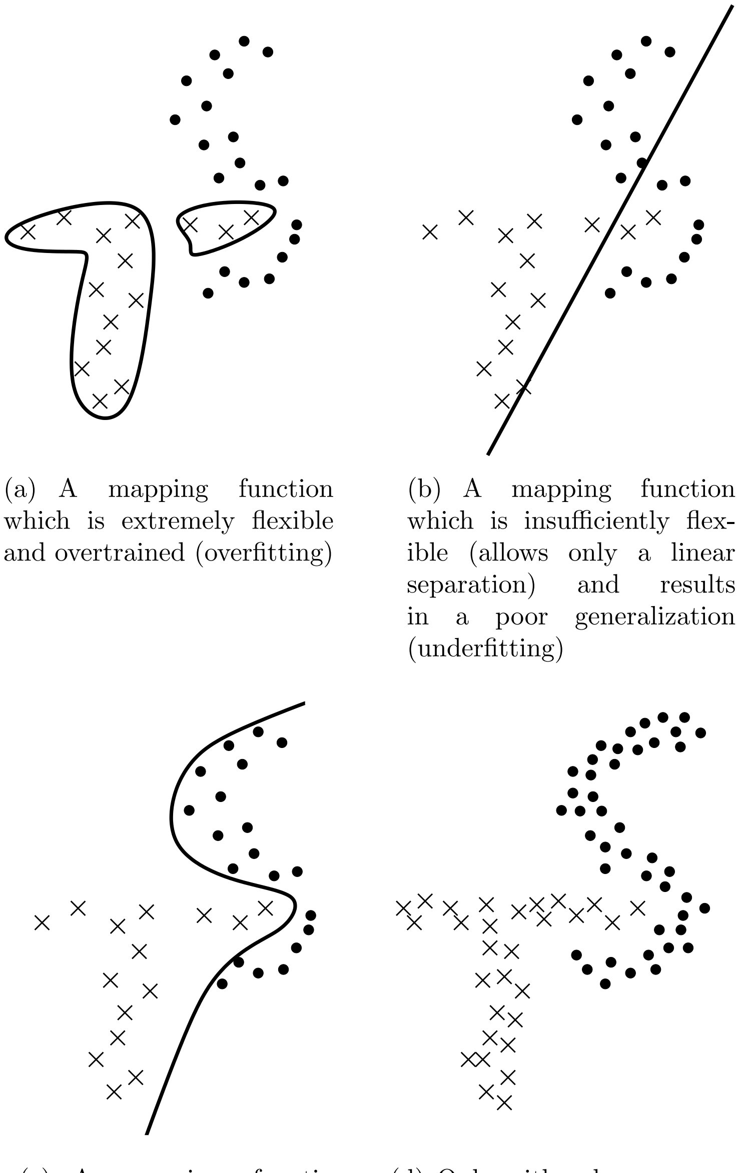


(a) A "typical" anomaly detection problem with two given classes (C_0 = circles and C_1 = triangles), where the anomalous class C_1 is strongly under-presented ($|C_0| = 200, |C_1| = 20$)



(b) The underlying probability distribution of class C_0 is depicted as a density plot. One can see that the anomalous data is not generated by the probability distribution of class C_0

Figure 4.1: Anomaly detection



(a) A mapping function which is extremely flexible and overtrained (overfitting)

(b) A mapping function which is insufficiently flexible (allows only a linear separation) and results in a poor generalization (underfitting)

(c) A mapping function which generalizes well, especially on unseen data samples

(d) Only with a large number of samples we are able to see the true distribution more closely

Figure 4.2: Overfitting-Underfitting dilemma

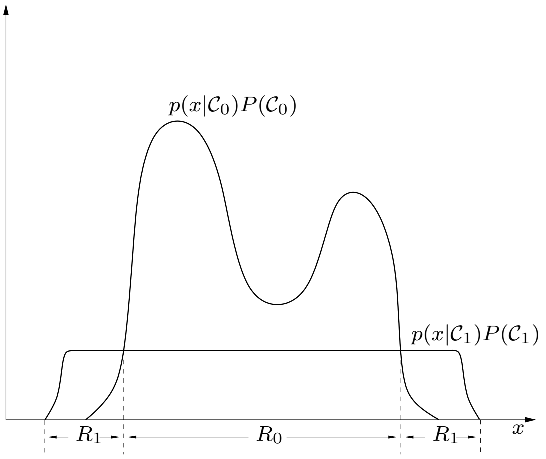


Figure 4.3: Bayesian decision for determining whether an input sample belongs to class $\mathcal{C}_0$ (falling in region $\mathcal{R}_0$) or $\mathcal{C}_1$ (falling in region $\mathcal{R}_1$) modeled with class-conditional density functions (taken from [7]).

where $P(C_k)$ is the *prior* probability of a sample belonging to class $\mathcal{C}_k$ with $k \in \{0, 1\}$ and $p(\mathbf{x}|C_k)$ is the class-conditional density. The class-conditional density $p(\mathbf{x}|\mathcal{C}_1)$ of the novel data represents the threshold and is unknown *a-priori*. Therefore, it can be modeled as a uniformly distributed density (see Fig. 4.3), which is constant over some large region of the input space [7]. The point of intersections divide the input space into two *decision regions* $\mathcal{R}_0$ and $\mathcal{R}_1$. An input sample falling in region $\mathcal{R}_0$ is assigned to class $\mathcal{C}_0$, otherwise it falls in region $\mathcal{R}_1$ and is assigned to class $\mathcal{C}_1$.

4.1.1 Parzen-Window Estimators

Parzen-Window is a nonparametric method for estimating density functions [57]. Given a set $\mathcal{A} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ of n i.i.d. samples drawn according to an unknown density function $p(\mathbf{x})$. The Parzen-Window method estimates $p(\mathbf{x})$ based on the n samples in $\mathcal{A}$ by

$$\hat{p}(\mathbf{x}) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right)$$

where K is a kernel function which must satisfy the condition

$$\int_{-\infty}^{+\infty} K(x)dx = 1$$

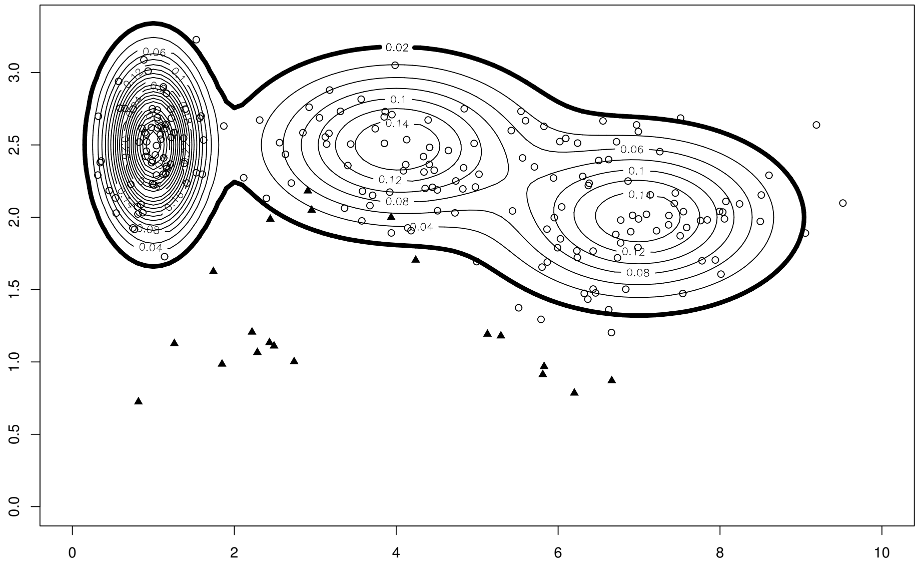


Figure 4.4: Density threshold anomaly detection

and h is the window width (also called smoothing parameter). A commonly used kernel function is the multivariate Gaussian kernel function

$$\hat{p}(\mathbf{x}) = \frac{1}{n(2\pi)^{d/2}\sigma^d} \sum_{i=1}^{n} \exp\left\{-\frac{||\mathbf{x}-\mathbf{x}_i||^2}{2\sigma^2}\right\}$$

where $\mathbf{x}_i$ are training samples which characterize the normal behavior and d is the dimensionality of the data space. The Gaussian kernel function is completely specified by the variance parameter σ which controls the degree of smoothness of the estimated density function.

By combining the Parzen-Window method and the Bayes classification method, one can obtain a statistical classification technique. First, a density function $\hat{p}(\mathbf{x})$ is estimated based on the "normal" training samples and second, a uniformly distributed density function[2] $p_u(\mathbf{x})$ is modeled *a-priori.* An unseen sample which falls in region $\mathcal{R}_0$ is classified as normal, otherwise it falls in region $\mathcal{R}_1$ and is classified as an anomalous sample. Figure 4.4 shows 200 data samples (circles) generated by a (known) probability distribution which is composed of three Gaussian distributions with different means and variances. The threshold of the uniformly distributed density function is de-

[2] the threshold

fined as $p_u = 0.02$ (bold curve). Three anomalous samples (triangles) are misclassified, all other are detected as anomalous.

The Parzen-Window method estimates the unknown probability distribution with a certain accuracy, when a certain amount of data samples is available and the variance parameter σ is properly chosen — this is illustrated in figure 4.5. Figure 4.5 shows a Gaussian ($\mu = 0, \sigma = 1$) probability distribution and Parzen-Window estimated probability distributions with different parameter settings. For $\sigma = 0.5$ and 50 given data samples (see Fig. 4.5(b)), the Gaussian distribution is properly estimated. In contrast, for 10 given data samples, the Gaussian probability distribution cannot be properly estimated (see Fig. 4.5(c)). Similar skewed results are obtained, when the variance parameter is incorrectly chosen (see Fig. 4.5(d) and Fig. 4.5(e)).

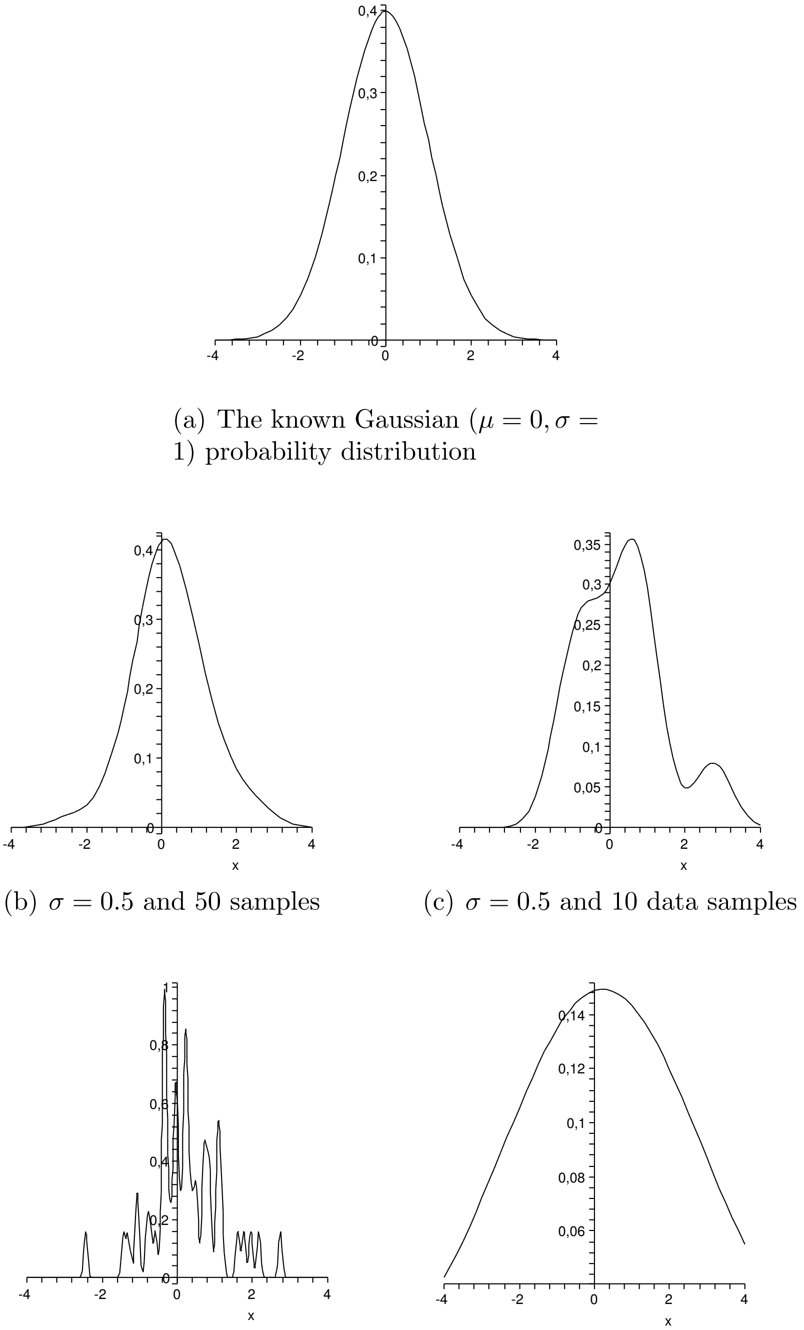
However, a large amount of data samples strongly bias the computational efficiency — each training sample is required to estimate the density for each single test sample — and makes this method computational infeasible for very large data sets.

4.1.2 One-Class Support Vector Machine

In many applications for instance machine fault detection or medical diagnosis, it is sufficient to estimate the support of the probability distribution, as opposed to the full density. A one-class Support Vector Machine (termed one-class SVM) [55] avoids estimating the full density. Instead, it estimates quantiles of the probability distribution, i.e. its support. The one-class SVM maps the input data into a higher-dimensional feature space $\mathcal{F}$ via a non-linear mapping Φ and treats the origin as the only member of the second class. In addition, a fraction ν of "outliers" are allowed, which lie between the origin and the hyperplane, where the hyperplane has maximum distance to the origin (see Fig. 4.6). In other words, the one-class SVM algorithm returns a function f that takes the value $+1$ in a region where the density "lives" and -1 elsewhere. Therefore, for a new point $\mathbf{x}$, the value $f(\mathbf{x})$ is determined by evaluating which side of the hyperplane it falls on in feature space.

More precisely, the optimal hyperplane is constructed, by solving the optimization problem

$$\begin{array}{ll} \min\limits_{\alpha} & \frac{1}{2}\sum\limits_{i,j=1}^{l} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \\ \text{subject to} & 0 \leq \alpha_i \leq 1/(\nu l),\ i = 1, \ldots, l \end{array}$$



(a) The known Gaussian ($\mu = 0, \sigma = 1$) probability distribution

(b) $\sigma = 0.5$ and 50 samples

(c) $\sigma = 0.5$ and 10 data samples

(d) $\sigma = 0.05$ and 50 data samples

(e) $\sigma = 2.5$ and 50 data samples

Figure 4.5: Parzen-Window (Gaussian Kernel) estimated probability distribution with different variance parameter σ and number of data samples

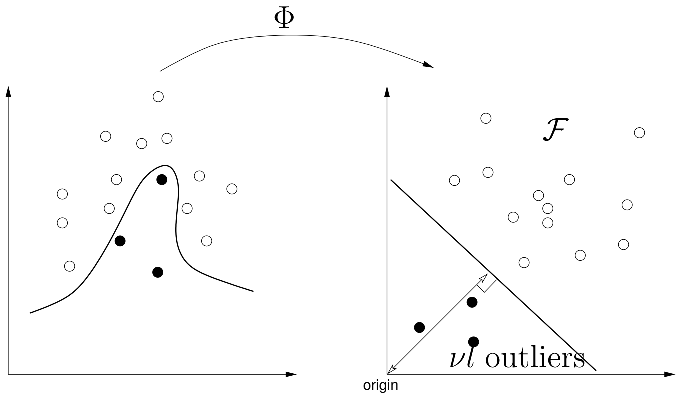


Figure 4.6: Map the training data into a high-dimensional feature space $\mathcal{F}$ via Φ. Construct a separating hyperplane with maximum distance to the origin, with the constrains that $\nu \cdot l$ outliers, lie between the origin and the hyperplane.

$$\sum_{i=1}^{l} \alpha_i = 1$$

where $\alpha_{1...l}$ are Lagrange multipliers, k the kernel function and $\mathbf{x}_{1...l}$ the training samples.

By solving this optimization problem, one obtains the decision function for testing sample $\mathbf{x}$

$$f(\mathbf{x}) \quad = \quad \operatorname{sgn}\left(\sum_{i=1}^{l} \alpha_i k(\mathbf{x}, \mathbf{x}_i) - \rho\right)$$

which will be positive for most examples $\mathbf{x}_i$ in the training set. The value of ρ can be recovered by exploiting the fact, that for any Lagrange multipliers α_i, the corresponding pattern $\mathbf{x}_i$ satisfies

$$\rho = (\mathbf{w} \cdot \Phi(\mathbf{x}_i)) = \sum_j \alpha_j k(\mathbf{x}_j, \mathbf{x}_i)$$

where $\mathbf{w}$ is the normal vector of the hyperplane.

An equivalent[3] one-class SVM approach is proposed by Tax and Duin [66]. Their approach is based on the idea of constructing a minium enclosing hypersphere in the feature space $\mathcal{F}$ and allowing a number of outliers, i.e. elements not supporting the data distribution, which lie outside the hypersphere. It is interesting to note, that both one-class SVM [55, 66] approaches can be transformed in a threshold Parzen-Window Estimator.

[3] By using the radial basis function kernel

4.2 Network Intrusion Detection Systems

Intrusion Detection Systems (IDSs) [3, 48] are software and hardware systems that automate the process of monitoring the events occurring in a computer system or network and analyze them for signs of *intrusions*. Heady et al. [33] defined an *intrusion* as “any set of actions that attempt to compromise the integrity, confidentially and availability of information resources”. Intrusions are caused by attackers accessing the system, authorized users of the systems who attempt to gain additional privileges for which they are not authorized, and computer worms and viruses which carry malicious code. IDSs are based on the belief that an intruder’s behavior will be noticeably different from that of a legitimate user and that many unauthorized actions are detectable. Typically, IDSs employ anomaly and ruled based misuse models in order to detect intrusions and are different in host-based and network-based systems. Host-based systems employ the host operating system’s audit trails as the main source of input to detect intrusive activity, while network-based IDSs build their detection mechanism on monitored network traffic. One of the most popular network-based IDS is the open source program *Snort* [42]. Snort is a rule-based IDS which contains a database, where known malicious patterns (termed signatures) are stored. Each network packet monitored by Snort is disassembled in several distinct packet components and compared to the signatures in the database. When a signature matches with a packet component, an event is triggered and appropriate actions can be executed. The following examples show different Snort rules, to detect a DDoS[4] communication, a worm and a buffer overflow attack.

Example 4.1.

```
alert icmp
 $EXTERNAL_NET any -> $HOME_NET any
(msg:"DDOS Stacheldraht client spoofworks";
icmp_id:1000; itype:0; content:"spoofworks";
classtype:attempted-dos; sid:227; rev:6;)
```

Example 4.2.

```
alert udp $EXTERNAL_NET any ->
$HOME_NET 1434 (msg:"MS-SQL Worm propagation
attempt"; content:
"|81 F1 03 01 04 9B 81 F1 01|";
classtype:misc-attack; sid:2003; rev:8;)
```

[4] Distributed Denial of Service

Example 4.3.

```
alert tcp $EXTERNAL_NET any ->
$HOME_NET 143
(msg:"IMAP partial body buffer overflow
attempt";content:"PARTIAL"; nocase;
content:"BODY["; distance:0; nocase;
pcre:"/\sPARTIAL.*BODY\[[^\]]{1024}/smi";
classtype:misc-attack; sid:1755; rev:14;)
```

In example 4.1, a snort signature is shown, which indicates the presence of a variant of the Stacheldraht DDoS tool [16]. Stacheldraht is a distributed denial of service tool, uses a tiered structure of compromised hosts to coordinate and participate in a denial of service attack. There are “handler” hosts that are used to coordinate the attacks and “agent” hosts that launch the attack. Communication between the handler and the agent is conducted using *icmp_echoreply*. The communication information is located inside an ICMP packet and consists of the ASCII string “spoofworks”. Example 4.2 shows the snort signature in hexadecimal representation of the “well known” SQL worm, which infected millions of computers, where an un-patched Microsoft SQL database was running. Example 4.3 shows the snort signature of a buffer overflow exploit to an IMAP Server. This event is generated when a remote authenticated user sends a malformed request for partial mailbox attributes to an IMAP server. The examples(4.1,4.2,4.3) emphasize the fact that it is necessary to inspect the network packet payload, to recognize and determine the *type* of the intrusion.

In contrast, Intrusion Detection Systems which employ anomaly models, establish profiles of normal activities of the operating system or the network traffic and detect intrusions by identifying significant deviations from the observed profiles. Network-based IDSs establish profiles based on connection vectors. A connection vector consists of different fields which characterize a network packet and the established connection such as source, destination, length of the message, time it was sent, the frequency of the communication, etc.

Table 4.1 shows a connection vector, which encompasses 15 fields with characteristics about the network packet and the connection, but without payload information. The fields described in table 4.1 are required to characterize a network session.

Ideally, a combination of anomaly and ruled based misuse model is applied, due to the drawbacks of both models. A ruled based misuse model cannot detect attacks for which it has no signatures — they do not react

Table 4.1: Connection Vector for Anomaly based Network Intrusion Detection (taken from [48])

Component	Description
Connection_ID	Unique integer used to reference the particular connection.
Initiator_address	The internet address of the host which initiated the connection.
Receiver_address	The internet address of the host to which the connection was made.
Service	An integer used to identify the particular service used for this connection.
Start_time	The time stamp on the first packet received for this connection.
Delta_time	The difference between the time stamp of the most recent packet of this connection and the Start_time.
Connection_state	The state of the connection. States for a connection include information such as: NEW-CONNECTION, CONNECTION-IN-PROGRESS, and CONNECTION-CLOSED.
Security_state	The current evaluation of the security state of this connection.
Initiator_pkts	The number of packets the host which initiated the connection has placed on the network.
Initiator_bytes	The number of bytes, excluding protocol headers, contained in the packets.
Receiver_pkts	The number of packets the host which received the connection has placed on the network.
Receiver_bytes	The number of bytes, excluding protocol headers, contained in the packets.
Dimension	The dimension of the Initiator_X and the Receiver_X vectors. This value is the number of strings patterns being looked for in the data.
Initiator_X	A vector representing the number of strings matched in Initiator_bytes.
Receiver_X	A vector representing the number of strings matched in Receiver_bytes.

well to the unknown. Anomaly based models have the weakness of high false alarm rates, i.e. “normal” is recognized as an intrusion.

4.3 Summary

The first part of this chapter presents the anomaly detection problem by means of a statistical approach. Anomaly detection can be considered as a pattern classification problem, where typically only a single class of data is available, or a second class of data is under-represented e.g. machine fault detection or medical diagnosis. In a probabilistic sense, novelty detection is equivalent to deciding whether an unknown test sample is produced by the underlying probability distribution that corresponds to the training set of normal examples. Such approaches are based on the assumption that anomalous data are not generated by the source of normal data.

The second part of this chapter, motivates and presents two different network intrusion detection models. Intrusion can be defined as any set of actions that attempt to compromise the integrity, confidentiality and availability of information resources. Network intrusion detection models are divided in ruled based misuse model and anomaly model. A ruled based misuse model contains a database, where known malicious patterns (termed signatures) are stored. In contrast, an anomaly model establishes profiles of normal activities of the operating system or the network traffic, and detects intrusions by identifying significant deviations from the observed profiles which are based on connection vectors. A connection vector must contain a required amount of network information for characterizing profiles of normal network activities — the required fields of a connection vector is shown in table 4.1.

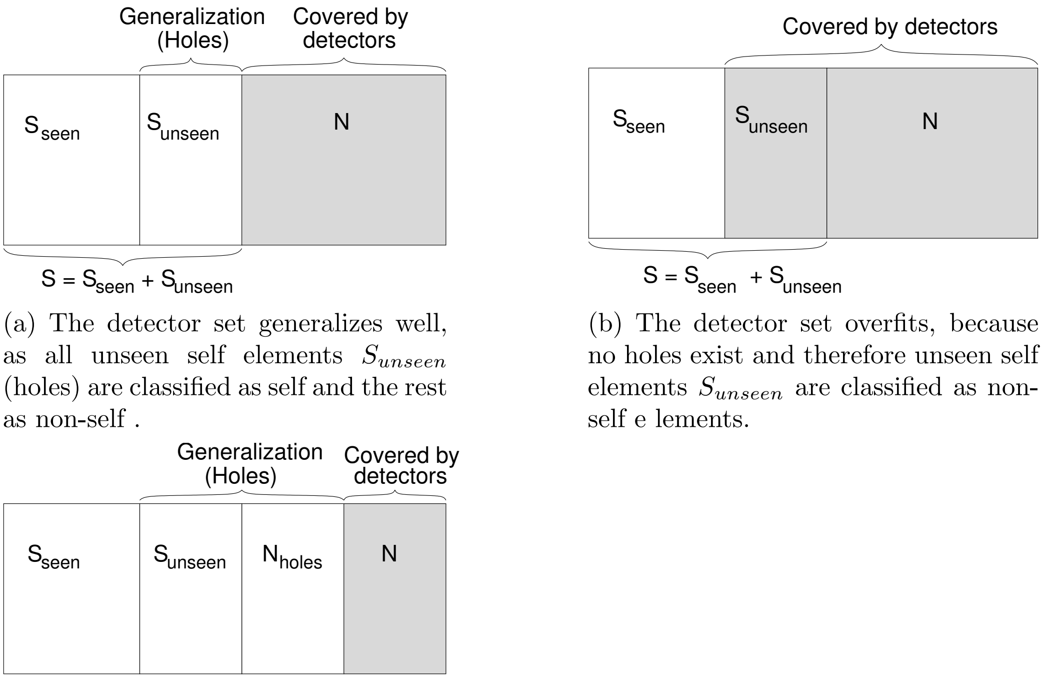
Chapter 5

Hamming Negative Selection

The generic negative selection algorithm can be used with arbitrary shape-spaces and affinity functions. In this chapter, we investigate the negative selection algorithm defined over the Hamming shape-space and the r-chunk matching rule. More specifically, the learning capability of the Hamming negative selection is explored. Furthermore, an r-chunk detector algorithm is proposed and analyzed in regard to the number of generable detectors and the space and runtime complexity. The results obtained are empirically verified, and discussed in the context of the applicability for a network intrusion detection system.

5.1 Generalization by Undetectable Elements

All matching rules (e.g. r-contiguous and r-chunk) which compare character sequences defined over a Hamming shape-space cause undetectable elements — termed holes [29]. Holes are elements of N, or self elements not seen during the training phase. For these elements, it is not possible to generate detectors and therefore, they cannot be recognized and classified as non-self elements. The term holes is an improper expression, because holes are *necessary* to generalize beyond the training set. A detector set which generalizes well, ensures that seen and unseen self elements are *not* recognized by any detector, whereas all other elements are recognized by detectors and classified as non-self (see Fig. 5.1(a)). A detector set which covers all non-self elements and all unseen self elements *overfits*, because no holes (no generalization) exists (see Fig. 5.1(b)). In contrast, a large number of holes implies that a large number of unseen self elements and non-self elements as well, are not covered by the detector set and therefore the detector set *underfits* (see Fig. 5.1(c)). Balthrop et al. [4] proposed a method (termed crossover-closure)



(a) The detector set generalizes well, as all unseen self elements S_{unseen} (holes) are classified as self and the rest as non-self .

(b) The detector set overfits, because no holes exist and therefore unseen self elements S_{unseen} are classified as non-self e lements.

(c) The detector set underfits, because a larger number of non-self elements N_{holes} are not recognized by the detectors and therefore are classified as self.

Figure 5.1: Holes are necessary to generalize beyond the training set. Too many holes results in an underfitting, whereas, no holes results in an overfitting.

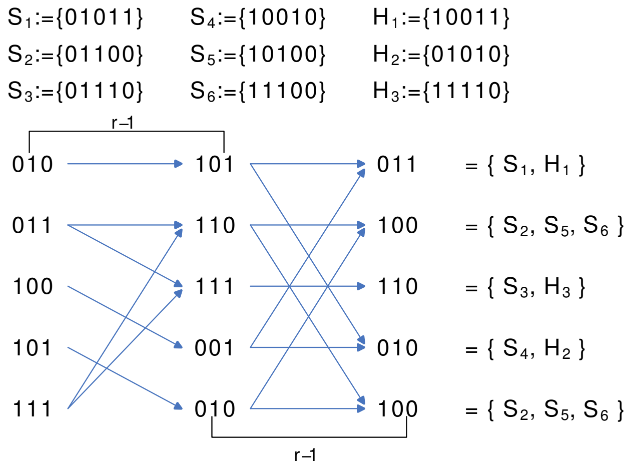


Figure 5.2: Construction to find holes for $r = 3$ and $l = 5$ for the r-chunk matching rule for $S = \{01011, 01100, 01110, 10010, 10100, 11100\}$. For elements $H_1 = \{10011\}, H_2 = \{01010\}, H_3 = \{11110\}$ it is not possible to generate a detector which recognizes H_1, H_2, H_3 as non-self. All possible generable detectors are $D = \{0|000, 0|001, 0|110, 1|000, 1|011, 1|100, 2|000, 2|001, 2|101, 2|111\}$

to find holes for the r-chunk matching rule by given parameters l, r and S. We have summarized Balthrop's method algorithmically (see algorithm 2) and show (see Fig. 5.2) an illustrative example of the construction for a set $S = \{01011, 01100, 01110, 10010, 10100, 11100\}$ and $l = 5, r = 3$.

From algorithm 2, it can be seen that holes arise in commonly occurring distinct self strings. The number of holes depends on the number of self elements, the element length l and the r-chunk length r. The runtime complexity of this algorithm is dominated by the instructions in line 4 and results in a total runtime complexity of $O(|S|^{l-r})$. The instructions in line 4 are equivalent to $l - r + 1$ nested for-loops, where each for-loop iterating from 1 to $n = |S|$. Therefore, in total $|S|^{l-r+1}$ substring comparisons are required. In the next section, we show formulas to calculate the number of generable detectors and the number of holes, under the assumption that S is randomly drawn from U_l^{Σ}.

```
Algorithm 2: Construct-Holes
input : Set $S = \{S_1, S_2, \ldots, S_n\}$ of self strings, element length $l$ and
        r-chunk length $r$
output: Set $H$ of undetectable elements (holes)
begin
    Cut $S_i$ in $l - r + 1$ substrings $S_{i,j} := S_i[j, \ldots, r - 1 + j]$
    for $j = 1, \ldots, l - r + 1$ , $i = 1, \ldots, n$

    Connect substring $S_{i,j}$ to $S_{k,j+1}$ with a direct edge, if the last $r - 1$
    characters of $S_{i,j}$ and the first $r - 1$ characters of $S_{k,j+1}$ are
    identical, for $i = 1, \ldots, n$, $k = 1, \ldots, n$ and $j = 1, \ldots, l - r + 1$

    Traverse direct edges and shuffle all coincident substrings $S_{i,j}$ for
    $i = 1, \ldots, n$ and $j = 1, \ldots, l - r + 1$ to obtain the set
    $H = (S \cup H) \setminus S$ of undetectable elements
end
```

5.2 Number of Holes

Esponda et al. [23] discussed and described formulas for estimating the number of holes for the binary alphabet $\Sigma = \{0, 1\}$. Summarizing Esponda et al. derivation, the number of holes $|H|$ can be calculated as follows :

$$|H| = CC(l, r, |S|) - |S| \tag{5.1}$$

where

$$CC(l, r, |S|) = \left(2^r - 2^r \left(1 - \frac{1}{2^r}\right)^{|S|}\right) \cdot (1 + P(r, |S|))^{(l-r)} \tag{5.2}$$

The subterm (5.2) yields the number of all strings, which can be constructed by algorithm 2. As stated above, this algorithm will also construct the input self set S (see Fig. 5.2) and therefore, this proportion must be subtracted (see Eq. (5.1)). The subterm $P(r, |S|)$ results from the likelihood of a substring s_i having outgoing edges.

$$P(r, |S|) = 1 - \frac{1}{2}(P_0 + P_1)$$

where

$$P_0(r, |S|) \quad = \quad \left(1 - \frac{1}{2^{r+1}}\right)^{4|S|}$$

$$P_1(r, |S|) = 4\left(1 - \frac{1}{2^{r+1}}\right)^{2|S|} - 4\left(1 - \frac{1}{2^{r+1}}\right)^{3|S|}$$

P_0 is the probability of a substring to have no outgoing edges and P_1 to have one outgoing edge. Using the subterm $P(r, |S|)$ and simplifying term (5.2) one obtains

$$H(|S|, l, r) = T_1 \cdot T_2 - |S| \tag{5.3}$$

where

$$\begin{aligned} T_1 &= 2^r - 2^r\left(1 - \frac{1}{2^r}\right)^{|S|} \\ T_2 &= \left[2 - \frac{1}{2}\left(1 - \frac{1}{2^{r+1}}\right)^{4|S|} - 2\left(\left(1 - \frac{1}{2^{r+1}}\right)^{2|S|} - \left(1 - \frac{1}{2^{r+1}}\right)^{3|S|}\right)\right]^{(l-r)} \end{aligned}$$

To summarize, given an alphabet of cardinality 2, l, r and $|S|$ randomly drawn self elements from U_l^2, the number of holes (undetectable elements) is described by formula 5.3.

5.3 Detector Generation Algorithm

We propose an algorithm called *Build-Rchunk-Detectors* which generates all possible r-chunk detectors, which will not cover any element in S. This algorithm uses a hashtable $\mathcal{H}$ data structure to insert, delete, and search efficiently for boolean values which are indexed with a key composed of an r-chunk string concatenated with a detector position p. Since the algorithm requires all keys from $\{0, \ldots, |\Sigma|^r\}$ concatenated with p, $\mathcal{H}$ contains $p|\Sigma|^r$ elements, where r is the $r - chunk$ length. In addition, the hashtable is an appropriate data structure to analyse randomized operations. Figure 5.3 shows the hashtable with generated detectors for alphabet $\Sigma = \{0, 1\}$. The symbol | means concatenation of two elements.

The *Build-Rchunk-Detectors* algorithm takes four input parameters, r-chunk length r, self set S, element length l, alphabet Σ and outputs an array of all possible detectors, which do not match self elements. The algorithm is divided into three phases. The initial phase (line 2 to 4) initializes all keys with the boolean value *true*. The label phase (line 5 to 10) iterates with a

Algorithm 3: Build-Rchunk-Detectors

```
input  : $S$ = Set of self elements, $r$ = r-chunk length, $l$ = element
         length, $\Sigma$ = alphabet
output: $D$ = Set of generated r-chunk detectors
begin
    for $i \leftarrow 0$ to $|\Sigma|^r - 1$ do
        for $p \leftarrow 0$ to $l - r$ do
            $\mathcal{H}.put(i|p, true)$
    foreach $s \in S$ do
        $c \leftarrow 0$
        while $r + c < length[s]$ do
            $rchunk \leftarrow substring[s, r + c]$
            $\mathcal{H}.put(rchunk|c, false)$
            $c \leftarrow c + 1$
    for $i \leftarrow 0$ to $|\Sigma|^r - 1$ do
        for $p \leftarrow 0$ to $l - r$ do
            $C \leftarrow \mathcal{H}.get(i|p)$
            if $C = true$ then
                $D[k] \leftarrow \mathcal{H}.get(i|p)$
                $k \leftarrow k + 1$
    return $D$
end
```

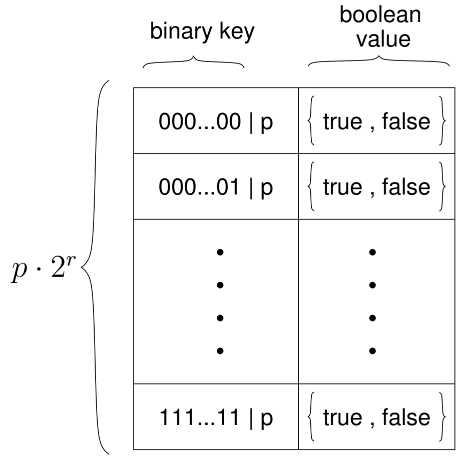


Figure 5.3: Hashtable $\mathcal{H}$ configuration with $\Sigma = \{0, 1\}$

length r sliding window[1] over all self elements s and replaces the hashtable boolean value, whose key matched the r-chunk with *false*. The last phase named *find* (line 11 to 16), iterates over all hashtable elements and extracts those which have a boolean value of *true*. The returned array D contains all possible detectors which do not cover any self element. The space complexity is determinated by r and position p, where p is negligible. The hashtable uses keys of length r and, therefore the total space size results in $O(|\Sigma|^r)$. The runtime complexity is determinated by r, S and the self elements length l. The three phases need $O((l - r) \cdot |\Sigma|^r) + O(|S| \cdot (l - r)) + O(|\Sigma|^r)$ time to generate all possible detectors. The total runtime complexity results in $O(|\Sigma|^r)$ and therefore is exponential in the length r.

5.4 Number of Detectors

In this section, we estimate the average number of generable r-chunk detectors. This number is determinated by the cardinality $|S|$, the element length l of S and the r-chunk length r. We use the hashtable $\mathcal{H}$ in algorithm 3 to estimate the average number of generable detectors.

Proposition 5.1. *Given a universe U_l^Σ which contains all elements of length l over the alphabet Σ, r-chunks length r and a self set S randomly drawn from U_l^Σ, the average number of detectors which do not cover any element in S is*

$$\left(1 - \frac{1}{(l - r + 1) \cdot |\Sigma|^r}\right)^{|S| \cdot (l - r + 1)} \cdot (l - r + 1) \cdot |\Sigma|^r \tag{5.4}$$

[1] substring operation

Proof 5.1. *The hashtable $\mathcal{H}$ contains $p|\Sigma|^r$ elements. We draw $n = |S| \cdot (l - r + 1)$ elements and want to find zero labeled* false *elements. The probability distribution therefore is $P(k) = \binom{n}{k} q^k \cdot (1-q)^{n-k}$. For $k = 0$ and $q = ((l-r+1) \cdot |\Sigma|^r)^{-1}$ this results in*

$$\left(1 - \frac{1}{(l-r+1) \cdot |\Sigma|^r}\right)^{|S| \cdot (l-r+1)}$$

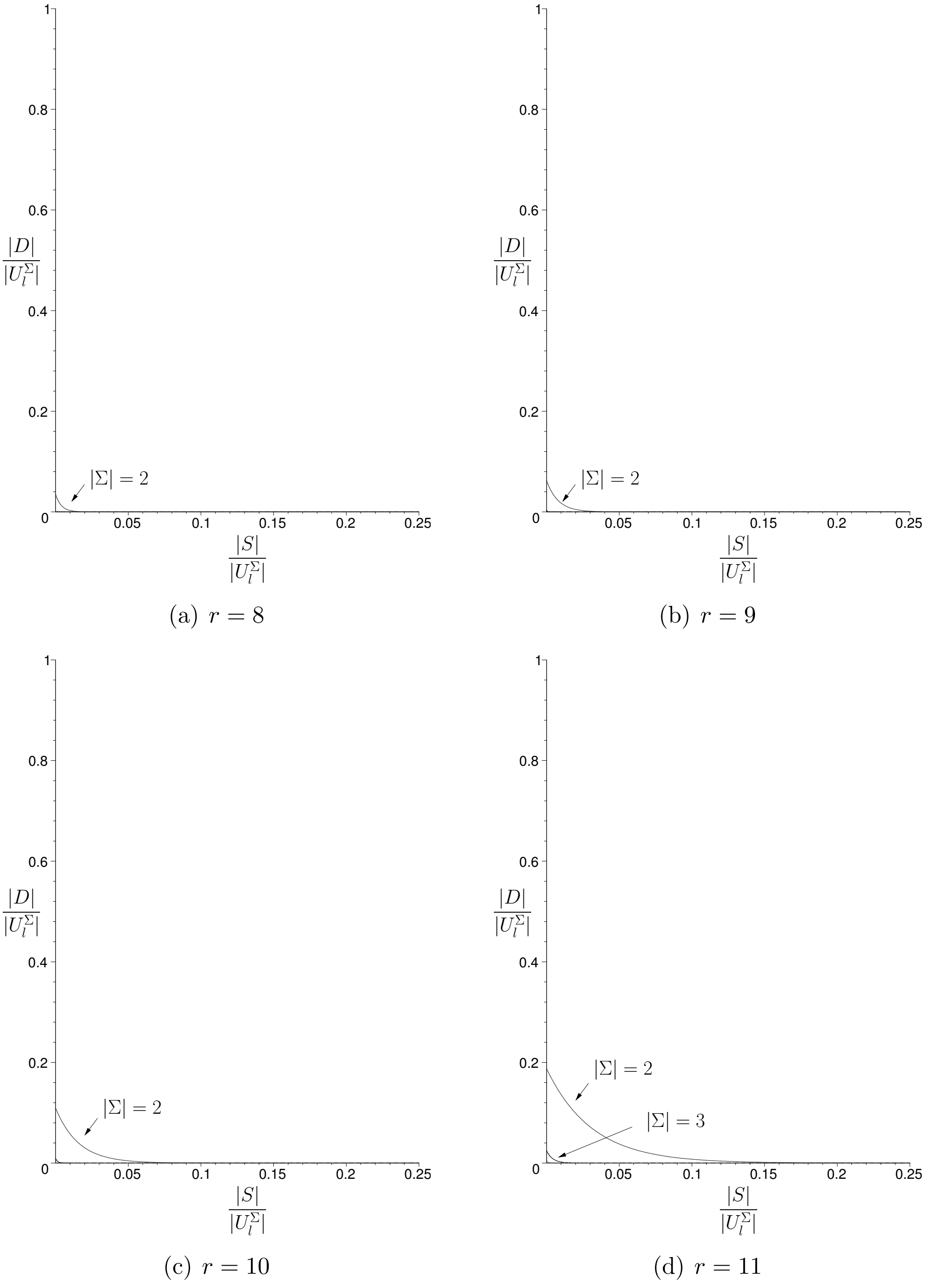
and the total average number results in

$$|D| = \left(1 - \frac{1}{(l-r+1) \cdot |\Sigma|^r}\right)^{|S| \cdot (l-r+1)} \cdot (l-r+1) \cdot |\Sigma|^r \qquad (5.5)$$

□

5.4.1 Higher Alphabets

In this section, we investigate the parameter dependencies of $|\Sigma|, |S|, r$ and their effects on the number of generable detectors. Therefore, we plot term (5.4) with small computable parameters, since term (5.4) increases exponentially. We choose $l = 16$, $r = 8, \ldots, l-1$, $|\Sigma| = 2, 3, 4, 5$ and select S randomly from U_l^Σ with a percentage proportion of $|S|/|U_l^\Sigma| = 0\%, \ldots, 25\%$ of the total universe U_l^Σ. In the plots the ordinate depicts the amount of generable detectors in proportion to the total universe. Since the universe and the amount of detectors increased with higher alphabets, we represent the relative number between $|D|$ and $|U_l^\Sigma|$. As it can be seen in figure 5.4(a) to 5.4(c), detectors can be only generated for $|\Sigma| = 2$ and $|S|/|U_l^\Sigma| < 5\%$. For higher alphabets it is not possible to generate detectors for $r \leq 10$. This phenomenon results from the r-chunk matching rule, which is not suitable for higher alphabets, since increased alphabet sizes influence the amount of generable detectors. As it can be seen in figures 5.4(a) to 5.5(d), most detectors are generable for an alphabet of size two. It is clear that these detectors recognize less elements from U_l^Σ than higher alphabets detectors for same value of l and r. But, as we see in figures 5.4(a) to 5.5(d), increasing the alphabet size implies less generable detectors for a fixed r. To generate a destined amount of detectors for arbitrary alphabet sizes, the r-chunk length must lie near l, which results in larger space complexity. If $|\Sigma| > 5$ and $r > 16$ it is not feasible to generate all possible detectors, due to the large space complexity.



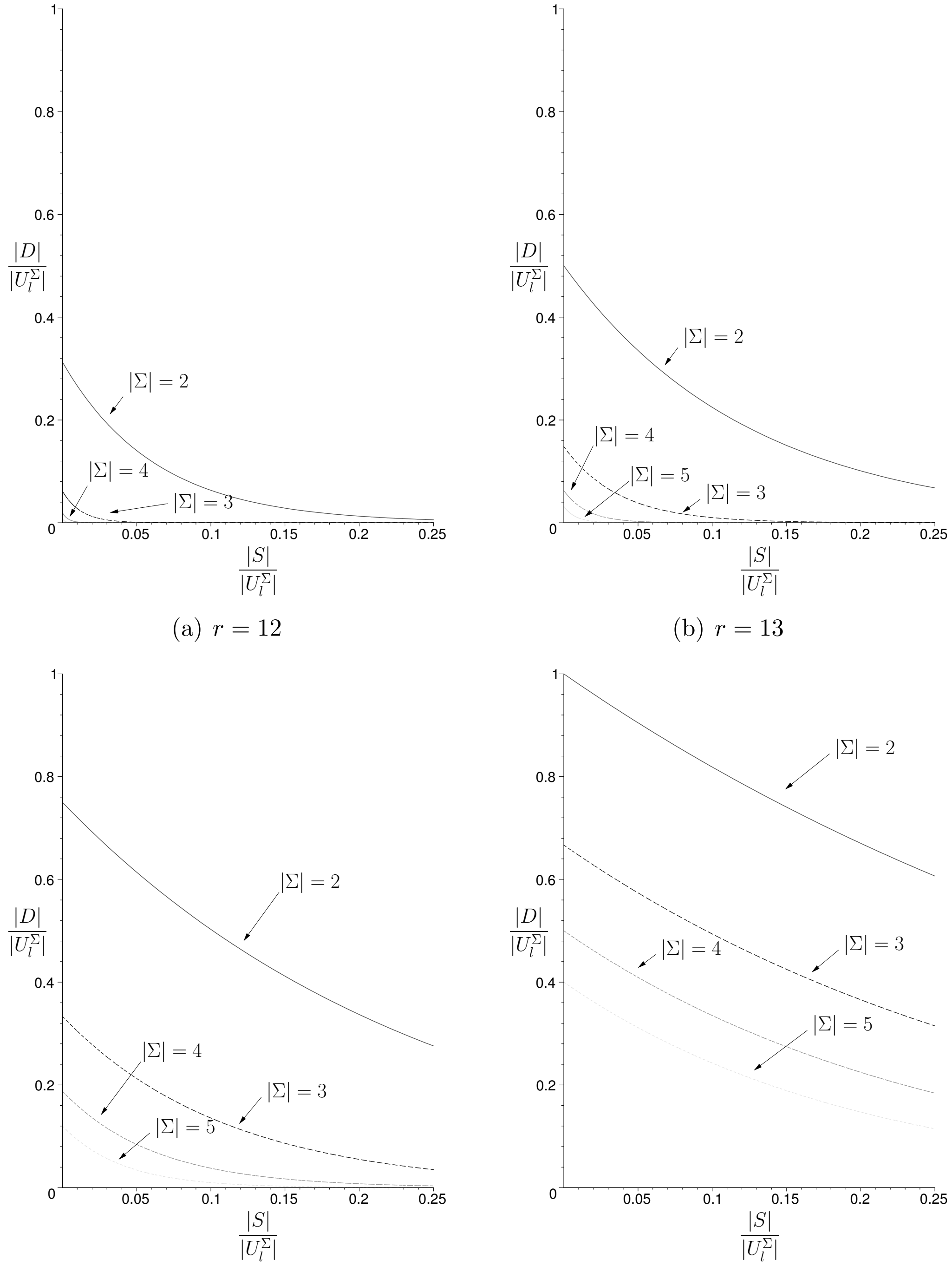
(a) $r = 8$

(b) $r = 9$

(c) $r = 10$

(d) $r = 11$

Figure 5.4: Plots of term (5.4), with $l = 16$, $r = \{8, 9, 10, 11\}$ and $|\Sigma| = \{2, 3, 4, 5\}$



(a) $r = 12$

(b) $r = 13$

(c) $r = 14$

(d) $r = 15$

Figure 5.5: Plots of term (5.4), with $l = 16$, $r = \{12, 13, 14, 15\}$ and $|\Sigma| = \{2, 3, 4, 5\}$

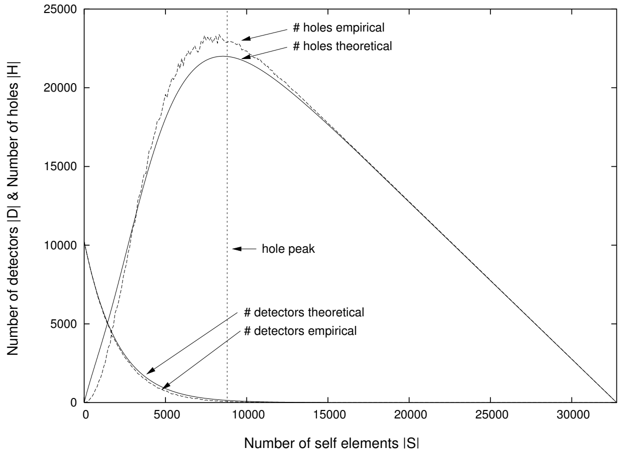


Figure 5.6: Numbers of detectors and numbers of holes, for $l = 15$, $r = 11$, $|\Sigma| = 2$ and $|S| = \{0, \ldots, 2^{15}\}$, calculated with Eqs. (5.4) and (5.3) and empirically by the algorithm output

5.5 Empirical Formula Verifications

In this section, the formulas to estimate the number of generable detectors and the number of holes are empirically verified and investigated. Therefore the proposed algorithm[2] (3) which generates all possible r-chunks detectors given the alphabet size Σ, string length l and r-chunk length r is used. With simple modifications — we count the number of non-self strings not detected by any detector, the algorithm also outputs the number of holes. Since the algorithm first initializes a hashtable of size $|\Sigma|^r \cdot (l - r + 1)$, we perform our simulations on a small value of l. The values $l = 15$, $r = 11$ and $|\Sigma| = 2$ were used. For each $|S| = \{0, \ldots, 2^{15}\}$ (randomly determined) a detector generation (algorithm) run was performed and the resulting number of detectors and holes were depicted in a graph (see Fig. 5.6). To verify the accuracy of Eq. (5.4) which calculates the number of detectors and Eq. (5.3) which calculates the number of holes, these results were also depicted in the graph (see Fig. 5.6). One can see, that the theoretical derived formulas, approximate well the empirical results. More interesting is the fact that for $l = 15, r = 11$, the number of generable detectors exponentially decrease to the number of self elements, whereas the number of holes exponentially increase. Reaching a certain proportion of self elements, no detectors can be generated and *all* non-self elements and unseen self elements are holes. In figure (5.6) this is termed *hole peak*. After the hole peak is passed, the holes decrease nearly

[2] This is also verified with algorithm (2)

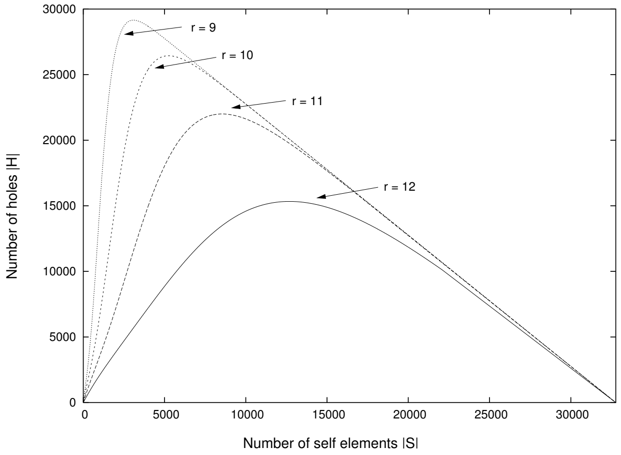


Figure 5.7: Number of holes for r-chunks length $r = \{9, 10, 11, 12\}$, $l = 15$, $|\Sigma| = 2$ and $|S| = \{0, \ldots, 2^{15}\}$

linear to 0, because the number of self elements increase and the relation $U = S \cup N$ must hold.

5.6 Controling Number of Detectors and Holes with r-chunk length r

Investigating equations (5.4) and (5.3), one can see, that the exponential curves behavior can be controlled by the parameters $|S|, l, r$. Commonly, the length l is fixed *a-priori*. The number of self elements $|S|$ depends on the element length l and is not adaptable, if the length l is defined *a-priori*. Therefore, the parameter r can be used to control the number of generable detectors and the number of holes. In figures (5.7) and (5.8) the effects for different r-chunks lengths are depicted. One can see, that increasing r closer to l, the number of holes decrease and the number of generable detectors increase. Furthermore, the hole peak moves toward a larger amount of self elements. This is an important property, since not all self elements are seen during the training phase. When r is *not* close to l, the number of holes (the generalization) increases exponentially with the number of self elements until the hole peak is reached. This means that the detector set exponentially underfits.

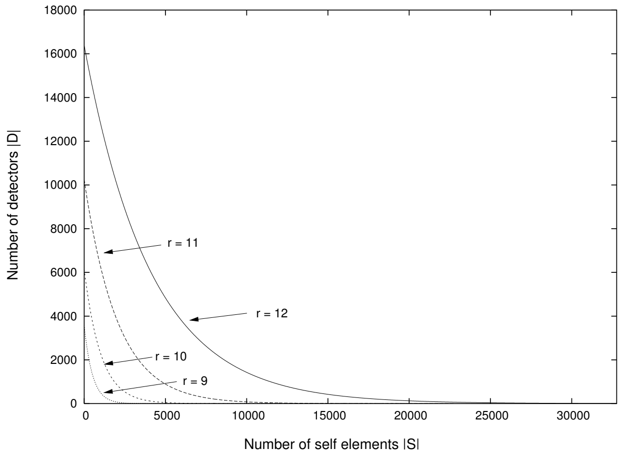


Figure 5.8: Number of generable detectors for r-chunks length $r = \{9, 10, 11, 12\}$, $l = 15$, $|\Sigma| = 2$ and $|S| = \{0, \ldots, 2^{15}\}$

5.7 Generalization Regions Experiments

Proposition 5.1 and formula 5.3 provide results under the assumption that the set S is randomly drawn from U. However these results provide no information *where* holes occur. Holes must occur in regions where most self elements are concentrated. As holes are not detectable by any detector, holes must represent unseen self elements (generalization). For studying the number, and the occurrence of holes which are dependent on the r-chunk length, we have created a number of artificial data sets (illustrated in figures A.1,A.2,A.3,A.4 and can be found in the Appendix). The first data set contained 1000 random points $p \in [0,1]^2$ which lie within a single circular cluster with centre $(0.5, 0.5)$ and radius $r = 0.1$. Each point $p = (x, y)$ is mapped to a binary element $e_0, e_1, \ldots, e_{15}$, where the first 8 bits encode the integer x-value $\lceil 255 \cdot x + 0.5 \rceil$ and the last 8 bits the integer y-value $\lceil 255 \cdot y + 0.5 \rceil$, i.e. $[0,1]^2 \rightarrow (i_x, i_y) \in [1, \ldots, 256 \times 1, \ldots, 256] \rightarrow (b_x, b_y) \in U_8^{\{0,1\}} \times U_8^{\{0,1\}}$. This mapping was proposed in [29]. The second and third data set contains 1000 randomly generated self elements which are lying within an ellipse and a rectangle. The fourth data set contains 1000 Gaussian ($\mu = 0.5, \sigma = 0.1$) generated points.

In figures A.1,A.2,A.3,A.4, one can see that for $r < 8$, holes occur in regions which lie outside of the self region — or put another way, only a limited number of holes exists at all (see e.g. Fig. A.2). Furthermore, it was observed that for $8 \leq r \leq 11$, holes occur in the generated self region (as they should) and a detector length of $r = 10$ provides the best generalization results. However, for $r > 11$ the detector length is too large and as a result, the self region is covered by the detectors rather than by the holes. It is worth noting, that a certain detector length — also called detector specificity — *must* be reached to obtain holes within the generated self region.

By calculating the *entropy* [44] of the binary representation of S for different r-chunk length r, it is possible to obtain an explanation for *why* a detector specificity $r \geq 8$ is required to obtain holes close or within the self region.

Entropy is defined as

$$H(X) \quad = \quad \sum_{x \in \mathcal{A}_X} P(x) \, \log_2 \left(\frac{1}{P(x)} \right) \quad \text{[bits]} \tag{5.6}$$

where the outcome x is the value of a random variable which takes one of the possible values $\mathcal{A}_X = \{a_1, a_2, \ldots, a_n\}$, having probabilities $\{p_1, p_2, \ldots, p_n\}$ with $P(x = a_i) = p_i$. Roughly speaking, the entropy is a measurement of randomness (uncertainty) in a sequence of outcomes. The entropy is maximal[3], when all outcomes have an equal probability. This can be simply verified by finding the extremum of $H(X)$ by means of a Lagrange multiplier[4] α.

Let

$$H(p_1, p_2, \ldots, p_n) = p_1 \cdot \log_2 \left(\frac{1}{p_1} \right) + p_2 \cdot \log_2 \left(\frac{1}{p_2} \right) + \ldots + p_n \cdot \log_2 \left(\frac{1}{p_n} \right)$$

be a multivariate function subject to the constraint

$$g(p_1, p_2, \ldots, p_n) = p_1 + p_2 + \ldots + p_n - 1 = 0$$

and α a Lagrange multiplier with $\nabla f = \alpha \nabla g$. Taking the partial derivates $\frac{\partial f}{\partial p_i}$ gives

$$\frac{\partial f}{\partial p_i} = \frac{\ln(1/p_i)}{\ln(2)} - \frac{1}{\ln(2)} \qquad \text{for} \quad i = 1, \ldots, n$$

and $\nabla f = \alpha \nabla g$ gives

$$\frac{\partial f}{\partial p_i} = \alpha \frac{\partial g}{\partial p_i} \Longleftrightarrow p_i = \frac{1}{e\,2^{\alpha}} \qquad \text{for} \quad i = 1, \ldots, n$$

putting p_i in the constraint g gives

$$\underbrace{\frac{1}{e\,2^{\alpha}} + \frac{1}{e\,2^{\alpha}} + \ldots + \frac{1}{e\,2^{\alpha}}}_{n \text{ terms}} = 1 \Longleftrightarrow \log_2 \left(\frac{n}{e} \right) = \alpha$$

[3] largest uncertainty

[4] also used in the one-class SVM for solving the optimization problem

hence

$$p_1 = p_2 = \ldots = p_n = \frac{1}{e\,2^{\alpha}} = \frac{1}{e\,2^{\log_2\left(\frac{n}{e}\right)}} = \frac{1}{n}$$

This shows that uniform distribution[5] has the maximum entropy.

In our entropy experiment, all 1000 generated self points are concatenated to one large bit string L_S of length $16 \cdot 10^3$. The bit string L_S is divided into $\lfloor 16 \cdot 10^3/r \rfloor$ substrings (the outcomes $\mathcal{A}_X$). The entropy for $r = \{2, 3, \ldots, 15\}$ for each data set is calculated and the ratio $H(X)/r$ to the maximum possible entropy is calculated, and depicted in a graph (see Fig. A.5). The maximum possible entropy for r-chunk length r is r bits (each r bit sequence is equally likely). In figure A.5, the coherence between $H(X)/r$ and r for each data set is presented. One can see that when the r-chunk length r is increased close to l, the entropy decreases as the bit strings of length r become more specific, rather than random. Of most interest is the value at $r = 8$. For this value, the entropy ratio $H(X)/r$ results in a spiky jump, when compared to the neighbor values $r = 7$ and $r = 9$. Through exploring the mapping function $[0,1]^2 \rightarrow (i_x, i_y) \in [1, \ldots, 256 \times 1, \ldots, 256] \rightarrow (b_x, b_y) \in U_8^{\{0,1\}} \times U_8^{\{0,1\}}$, one can see that the bit string of length 16 is semantically composed of two bit strings of length 8 which represents the (x, y) coordinates. An r-chunk length $r < 8$ destroys the mapping information — the semantic representation of the (x, y) coordinates — and therefore the bit strings of length r have a random character rather than a semantic representation of the (x, y) coordinates. As a consequence, holes occur in regions, where actually *no* self regions should be (see Fig. A.1(a)-A.1(f), A.2(a)-A.2(f), A.3(a)-A.3(f), A.4(a)-A.4(f)).

It has been noted, that a similar statement — without empirical results — was mentioned by Freitas and Timmis [28] with regard to the r-contiguous matching rule :

"It is important to understand that r-contiguous bits rule have a positional bias".

Our entropy experiments support and empirically confirm this statement.

Furthermore, the observations implicate an additional "positional bias" problem. When elements of different lengths are concatenated to a data chunk, and the r-chunk length is too large (too specific) for some small length elements and also too small (too generic) for some large length elements, then holes occur in the wrong regions (see Fig. 5.9). Figure 5.9 shows elements e_1, e_2 — which represent coordinates (x_1, x_2) and (y_1, y_2) — of different lengths and an r-chunk detector of length $r = 12$. This r-chunk

[5] all outcomes have an equal probability

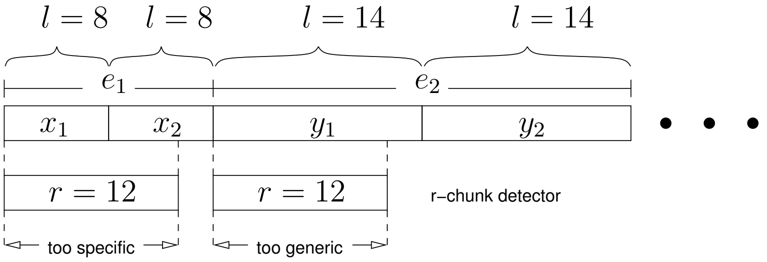


Figure 5.9: Concatenating elements e_1, e_2 of different length, can result in wrong generalization, as no suitable r-chunk detector length exists which capture the representations of e_1 and e_2.

length is too specific for length $l = 16$ of e_1, but likewise too generic for length $l = 28$ of e_2. As a consequence, no suitable r-chunk detector length for this example in figure 5.9 exists.

We emphasize this "positional bias" problem here as in Hamming negative selection approaches, when applied as a network intrusion detection technique, elements[6] of different lengths are concatenated: the implications are clear and discussed in section 5.9.

5.8 Detector Generating Algorithms with Exponential Complexity

The first proposed negative selection algorithm (see algorithm 1) was inspired heavily by the generation of T-Cells in the immune system. Candidate detectors were drawn at random from U and checked against all elements in S. This process of random generation and checking was repeated until the required number of detectors was generated. This random search for detectors has a constant space complexity in $|S|$, but an exponential runtime complexity in $|S|$ and is therefore not applicable (see term 3.6). In a succeeding work, D'haeseleer et. al [15] proposed two detector generating algorithms for the r-contiguous matching rule and the binary alphabet with an improved runtime complexity. The *linear time detector generating algorithm* has a linear runtime complexity in $|S|$ and $|D|$, but still requires time and space exponential in r :

$$\begin{aligned} \text{Time :} \quad & O\left((l-r)\cdot|S|\right) + O\left((l-r)\cdot 2^r\right) + O\left(l\cdot|D|\right) \\ \text{Space :} \quad & O\left((l-r)^2\cdot 2^r\right) \end{aligned}$$

[6] IP-Addresses, Ports, etc.

The second algorithm termed *greedy detector generating algorithm* has a similar complexity :

$$\begin{aligned} \text{Time :} \quad & O\left((l-r)\cdot|D|\cdot 2^r\right) \\ \text{Space :} \quad & O\left((l-r)^2\cdot 2^r\right) \end{aligned}$$

Another algorithm termed *binary template* for r-contiguous matching rule was proposed by Wierzchoń [71]. It has also an exponential complexity in r :

$$\begin{aligned} \text{Time :} \quad & O\left((l-r)\cdot 2^r\cdot|D|\right)+O\left(2^r\cdot|S|\right) \\ \text{Space :} \quad & O\left((l-r)\cdot 2^r\right)+O\left(|D|\right) \end{aligned}$$

Ayara et. al [2] proposed an r-contiguous detector generating algorithm termed *NSMutation* which based on an evolutionary (mutation operator) search method. The NSMutation has the following complexity :

$$\begin{aligned} \text{Time :} \quad & O\left(|\Sigma|^l\cdot|S|\right)+O\left(|D|\cdot|\Sigma|^r\right)+O\left(|D|\right) \\ \text{Space :} \quad & O\left(l\cdot(|S|+|D|)\right)+O(|D|) \end{aligned}$$

For the r-chunk matching rule and arbitrary alphabet sizes Σ, Stibor et. al [60] proposed an algorithm (see algorithm 3) with complexity :

$$\begin{aligned} \text{Time :} \quad & O\left((l-r)\cdot|\Sigma|^r\right)+O\left(|S|\cdot(l-r)\right)+O\left(|\Sigma|^r\right) \\ \text{Space :} \quad & O\left((l-r)\cdot|\Sigma|^r\right) \end{aligned}$$

All five proposed algorithms have a runtime or a space complexity, which is exponential in r and are *only* applicable for small values of r. Using for example a value $r = 64$ and an alphabet size $|\Sigma| = 2$, the space and time complexity are infeasibly high.

5.9 Hamming Negative Selection as a Network Intrusion Detection Technique

In many works, the appealing connections between the immune system and the IDSs are stressed. Intuitively, it seems obvious to abstract immune system principles and conceptualize algorithms for intrusion detection problems. This is especially so, as the capability of the immune system for dynamically adapting to previously unseen disease, is an attractive property for developing intrusion detection systems.

In this section, previously proposed approaches with Hamming negative selection as a network intrusion detection technique are presented and exhaustively discussed by means of our presented results.

Hofmeyr et al. [35] and later Balthrop et al. [5] developed a network intrusion detection system by applying the Hamming negative selection and the r-contiguous matching rule. The r-contiguous detectors were randomly generated (see algorithm 1 and figure 3.3). Moreover, in Hofmeyr's and Balthrop's approach, each network packet was encoded as a bit string (termed datapath) of length 49, consists of a source and destination IP address and a TCP port. More precisely, as an IP address has a total length of 32 bit, the total length must results in $32 * 2 + 8 = 76$ bits. To reduce the length from 76 to 49 Hofmeyr [36] argued :

"*Because at least one computer in the datapath must be on the LAN, the first 8 bits represent an internal computer, and are taken from the least significant byte of its IP address [footnote : It is assumed here that all computers on the LAN have the same class C network address. This representation is thus limited in that there can be at most 256 computers on the LAN, which is true for the network used to collect data for all experiments in this dissertation]*".

As a result, Hofmeyr has considered only class C network addresses, i.e. maximum 254 allowed hosts[7] which are represented by 8 bits instead of 32 bits. The further network information were represented as follows :

"*The following 32 bits represent the other computer involved in the communication, which will require 32 bits for the IP address if that computer is external. If the other computer is internal, then only 8 bits are needed, but 32 bits (the full IP address) are still used to maintain a fixed length representation. If an external computer is involved, it is always represented in the second 32 bits, so an extra bit is used to indicate whether or not the first computer is the server; if the first is the server, the bit is set to one, otherwise it is set to zero. The final 8 bits represent the type of service*".

To illustrate that more clearly see figure 5.10.

[7] 256 minus network and broadcast address yields 254

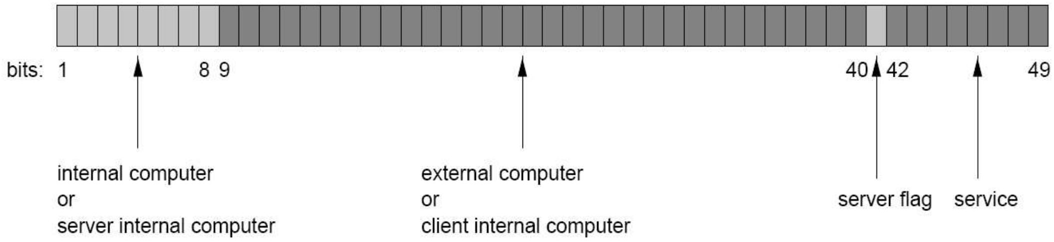


Figure 5.10: Hofmeyr's proposed network connection vector, which should characterize network traffic (taken from [36])

For (real-world) network intrusion detection problems like those described in section 4.2 this network encoding is *insufficient*. For detecting this kind of network misuse, a simple packet based firewall with logging functionality is an appropriate technique [19]. It is interesting to note, that a similar network packet encoding is also reported in the latest work in that field [72] :

"*The source IP addresses and ports together with destination IP addresses and ports are parsed by the data pre-processing module*".

Historically seen, Kim and Bentley were the first who mentioned that this encoding is insufficient [41]. They mentioned that the primitive fields[8] were not enough to build a meaningful profile. As a result, they proposed to represent network traffic as strings of length 33 defined over an alphabet of cardinality 10. They empirically conclude, that a random detector generation with parameter $|\Sigma| = 10$, $l = 33$ and $r = \{3, 4\}$ is completely impractical. Their observations can be explained with results shown in section 5.4.1. For such a large alphabet cardinality and small values of r it is not possible to generate any detectors.

Moreover, in the Hofmeyr's et al. [35] approach the cardinality of self set S (normal network behavior) was dramatically reduced to a very small cardinality :

"*The self trace S_t was collected over 50 days, during which a total of* 2.3 *million TCP connections were logged, each of which is a datapath triple. These 2.3 million datapaths were filtered down to* 1.5 *million datapaths. ..... After this, a self set S, consisting of* 3900 *unique strings, was extracted from S_t*".

[8] IP address, port

A similar statement is found in Balthrop's et al. [5] paper :

"*Mapping the packets into* 49-*bit representation yields a total of* 131 *unique strings in the training set, and* 551 *strings in the test set.*"

A simple real-world example shows that this data reduction is unrealistic. Given for instance a class C network with 254 hosts and *only* 254 servers worldwide distributed in the internet which offers services[9] on 500 available ports, then one obtains $254^2 * 500 = 32258000$ datapath triples.

By applying the correct estimation of the number of initial detector strings (before negatively selected), one obtains an infeasible complexity for generating suitable detectors, as term 3.6 is exponential in $|S|$.

Example 5.1. *Generating* only one *suitable detector for* $|S| = 32258000, l = 49, r = 12$, *i.e.*

$$|D| = 1 = |D_0| \cdot P_{\neg S}$$

results in

$$|D_0| \approx 1.148 \cdot 10^{66781}$$

this means that $1.148 \cdot 10^{66781}$ *initial detectors must be checked, before one suitable detector is found. Applying term 3.6, i.e. calculating the number of initial detectors* $|D_0|$ *by allowing with probability* P_{fail} *that* $|D|$ *detectors fail to detect an intrusion, one obtains for* $|S| = 32258000, l = 49, r = 12, P_{fail} = 0.1$

$$|D_0| \approx 5.558 \cdot 10^{66783}$$

Example 5.1 makes it clear that a random detector generation is not applicable for real-world network intrusion detection problems.

Furthermore, the chosen detector length ($r = 12$) is from our point of view, far too short. Following our analysis the detector length should induce a very large numbers of holes. Additionally, it is not clear, if for such a small value of r and different concatenated bit string lengths, the induced holes are lying in appropriate generalization regions.

Moreover, we showed in section 4.2, how IDS Snort recognizes attackers and worms by means of pattern matching. Usually, the IDS signatures have a length of 10 bytes or more. The signature which matches for example the IMAP buffer overflow, has a string length of 39 bytes. Intrusion detection systems which employ anomaly models, require connection vectors of a determined length which characterize the network packets and connections.

[9] rpc,smtp,ssh,pop3,imap,http, ftp,kerberos,irc,etc.

Assuming each field in the connection vector requires 2 bytes to store the data[10], then the connection vector described in table 4.1, requires 30 bytes.

Using the analysis in sections 5.2, 5.4, 5.7, which shows the coherence between the number of generable detectors, the number of resulting holes and the generalization regions, it is shown, that r *must* be close to l to generate a certain number of detectors, to control the number of holes (under/overfitting behavior) and to obtain holes near concentrated self regions.

A large number of holes implies a low detection rate, i.e. attacks are not recognized by the detectors. On the other hand, a limited number of holes results in a high false alarm rate, i.e. unseen normal data is recognized by the detectors and classified as an attack. Furthermore, the complexity of the most known algorithms to generate detectors is presented and the high runtime and space complexity is stressed.

Combining these arguments, it is clear, that the so far proposed AIS intrusion detection approach based, on the negative selection principle is not appropriate and not applicable for these kind of network intrusion detection models. Using a connection vector, which consist of 240 bits or signatures which consist of 80 bits, it is not possible to generate in polynomial time a certain number of detectors and to obtain a linear[11] control on the number of holes.

5.10 Summary

In this chapter the Hamming negative selection with the associated r-chunk matching rule is explored. An algorithm for generating all possible r-chunk detectors is proposed and analyzed. Furthermore a formula for calculating the number of generable r-chunk detectors is presented. It is shown that the number of generable detectors strongly depends on the r-chunk length and the number of self elements. Furthermore, the coherence between the alphabet size and the number of generable detectors is investigated. Additionally, holes (undetectable elements) are motivated and an algorithm for constructing holes is presented. Holes are necessary to generalize beyond the seen training set and therefore must represent unseen self elements. This important fact is explored empirically on different artificial data sets. It was noticed, that a certain r-chunk length (detector specificity) must be reached, that holes occur in regions where self elements are concentrated. The r-chunk length

[10] which is very optimistic estimation, because many fields are contiguous values and require at least 4 bytes

[11] with respect to the numbers of self elements

cannot be arbitrary chosen, as it must capture the semantical representation of the elements. Moreover it was noticed, that different element lengths cannot be arbitrary concatenated, as a proper detector length must capture the semantical representation of all concatenated elements.

In the last part of this chapter we have discussed the appropriateness of the Hamming negative selection as a technique for network intrusion detection. For this purpose previously proposed approaches were presented. We have shown, that these approaches are not useful for real-world security problems. From our point of view it is not possible to characterize a connection vector consisting of IP addresses and ports. Furthermore, the limited information within the (primitive) connection vectors are strongly compressed and the number of all collected connection vectors is strongly reduced. We believe, that this compression was done, as the detector generation process becomes infeasible for a large number of self elements (demonstrated on an example). Moreover, we have presented the time and space complexity of all r-contiguous and r-chunk detector generating algorithms proposed in the literature. Each algorithm has a time or space complexity which is exponential in the r-chunk length r and only feasible for small values of r.

Summarizing all obtained results, we conclude that the Hamming negative selection approach is not appropriate as a technique for network intrusion detection.

Chapter 6

Real-Valued Negative Selection

The principle of generating detectors in the complementary space and using these detectors to classify elements, is adaptable to other shape-spaces. In this chapter we investigate the negative selection defined over the real-valued shape-space. More precisely, we investigate the real-valued negative selection with variable-sized detectors proposed by Ji and Dasgupta [39, 40]. Before we start to investigate the real-valued negative selection with variable-sized detectors, we describe the abstracted immune elements and the functionality of the negative selection principle defined over a real-valued shape-space.

6.1 Generic Real-Valued Negative Selection

The idea to generate detectors in the complementary space for continuous data, was proposed informally by Ebner et al. [18] and formally by González et al. [30, 31]. The generic real-valued negative selection algorithm, operates on a unitary hypercube $[0,1]^n$. A detector $d = (\mathbf{c}_d, r_{ns})$ has a center $\mathbf{c} \in [0,1]^n$ and a non-self recognition radius $r_{ns} \in \mathbb{R}$. Furthermore, every self element $s = (\mathbf{c}_s, r_s)$ has a center and a self radius r_s. The self-radius was proposed to allow other elements to be considered as self elements which lie close to the self-center. If an element lies within a detector (hypersphere), which in effect would be close to the self-center given a certain radius, then it is classified as non-self, otherwise as self. An element[1] $\mathbf{e}$ lies within a detector $d = (\mathbf{c}_d, r_{ns})$, if the Euclidean distance $dist(\mathbf{c}, \mathbf{e}) = \left(\sum_{i=1}^{n}(c_i - e_i)^2\right)^{1/2} \leq r_{ns}$.

[1] n dimensional point

6.2 Real-Valued Negative Selection with Variable-Sized Detectors

Ji and Dasgupta [39, 40] proposed a real-valued negative selection algorithm with variable-sized detectors — termed *V-Detector* (see algorithm 4). The algorithm randomly determines a center $\mathbf{x}$ of a detector (see line 7) which must not lie within already existing detectors (see line 9) or within the hypersphere of a self-element (see line 16). Once such a center $\mathbf{x}$ is found, the radius of the detector is dynamically resized until the boundary of the region comes in contact with the closest self-element (see line 16-17). The detector is added in the detector set, if all conditions previously termed are satisfied and the resized detector radius r is greater than then self-element radius r_s (see line 18). The algorithm terminates if a predefined number of detectors are generated, or a pre-determined proportion of non-self space is covered.

For all our experiments contained in this chapter and in chapter 7, we employed the algorithm proposed by Ji and Dasgupta [39, 40] (see algorithm 4).

The real-valued negative selection was originally proposed to overcome scaling problems[2] inherent in the Hamming shape-space negative selection algorithm.

In the following sections, we investigate termination behavior of algorithm (4) and undertake an analysis and comparison of the classification performance on low- and high-dimensional data sets (see chapter 7). Our investigations reveal that the real-valued negative selection is a technique which is not competitive to statistical anomaly detection techniques on high-dimensional data sets. This fact is theoretically investigated and explained in chapter 8. However, on low-dimensional data sets it produces similar classification results like the real-valued positive selection which we propose in section 6.3.

6.2.1 Algorithm Visualization

As explained above, the V-Detector algorithm randomly generates detectors with a variable-sized radius. In order to assess how well the algorithm generates a set of non-self detectors and terminates, we made use of a simple

[2] runtime and space complexity

Algorithm 4: Generate V-Detector Set

```
   input  : $S$ = Set of self elements, $T_{max}$ = max. number of
            V-Detectors,
            $r_s$ = self radius, $c_0$ = estimated coverage, $MSC$ = max. s elf
   coverage
   output: $D$ = Set of generated V-Detectors
 1 begin
 2     $D \longleftarrow \emptyset$
 3     repeat
 4         $t \longleftarrow 0$
 5         $T \longleftarrow 0$
 6         $r \longleftarrow \infty$
 7         $\mathbf{x} \longleftarrow$ random point from $[0,1]^n$
 8         foreach $d \in D$ do
               // Euclid. distance between detector center $\mathbf{c}_d$
               and $\mathbf{x}$
               // is less than Non-Self radius $r_{ns}$ of detector $d$
 9             if $dist(\mathbf{c}_d, \mathbf{x}) \leq r_{ns}$ then
                   // point $\mathbf{x}$ is covered by a detector
10                 $t \longleftarrow t + 1$
11                 if $t \geq 1/(1 - c_0)$ then
12                     return D
13                 goto 5:
           // find the closest distance to a self element
           margin
14         foreach $s \in S$ do
15             $l \longleftarrow \mathrm{dist}(\mathbf{c}_s, \mathbf{x})$
16             if $l - r_s \leq r$ then
17                 $r \longleftarrow l - r_s$
18         if $r > r_s$ then
               // Add a new detector $d$ to set $D$
19             $D \longleftarrow D \cup \{d = (\mathbf{x}, r)\}$
20         else
21             $T \longleftarrow T + 1$
22         if $T > 1/(1 - MSC)$ then
23             exit
24     until $|D| = T_{max}$
25 end
```

toy problem. We created a simple two-dimensional artificial data set with 9 self elements (see Fig. 6.1(a)). We ran[3] the algorithm with different self-radius lengths r_s and estimated coverage c_0 paramters. Furthermore we set the remaining parameters as proposed in [40] :

$$\begin{aligned} \text{Maximum Self Coverage} \quad & MSC && = 99.99\,\% \\ \text{Maximum Number of Detectors} \quad & T_{max} && = 1000 \end{aligned}$$

The results are visualized in figure 6.1. Figure 6.1(b) shows the generated detectors for the artificial data set for self-radius $r_s = 0.05$ and estimated coverage $c_0 = 99\,\%$. It can be seen that the algorithm generates variable-sized detectors which cover the non-self space with a limited number of overlapping detectors. Two *independent* algorithm runs for $r_s = 0.05$ and $c_0 = 80\,\%$ were also performed (see Fig. 6.1(c), 6.1(d)). It can be seen, that this random detector generation and coverage estimation method varies a great deal with equal parameter settings. To obtain a steady space coverage for each independent algorithm run, the parameter c_0 must be close to $100\,\%$ or by applying a proper coverage estimation — for instance the Monte Carlo Integration technique (see appendix A.3). Consequently, this increases the runtime complexity required to generate detectors and therefore bias the termination behavior[4]. This is now analyzed in the following section.

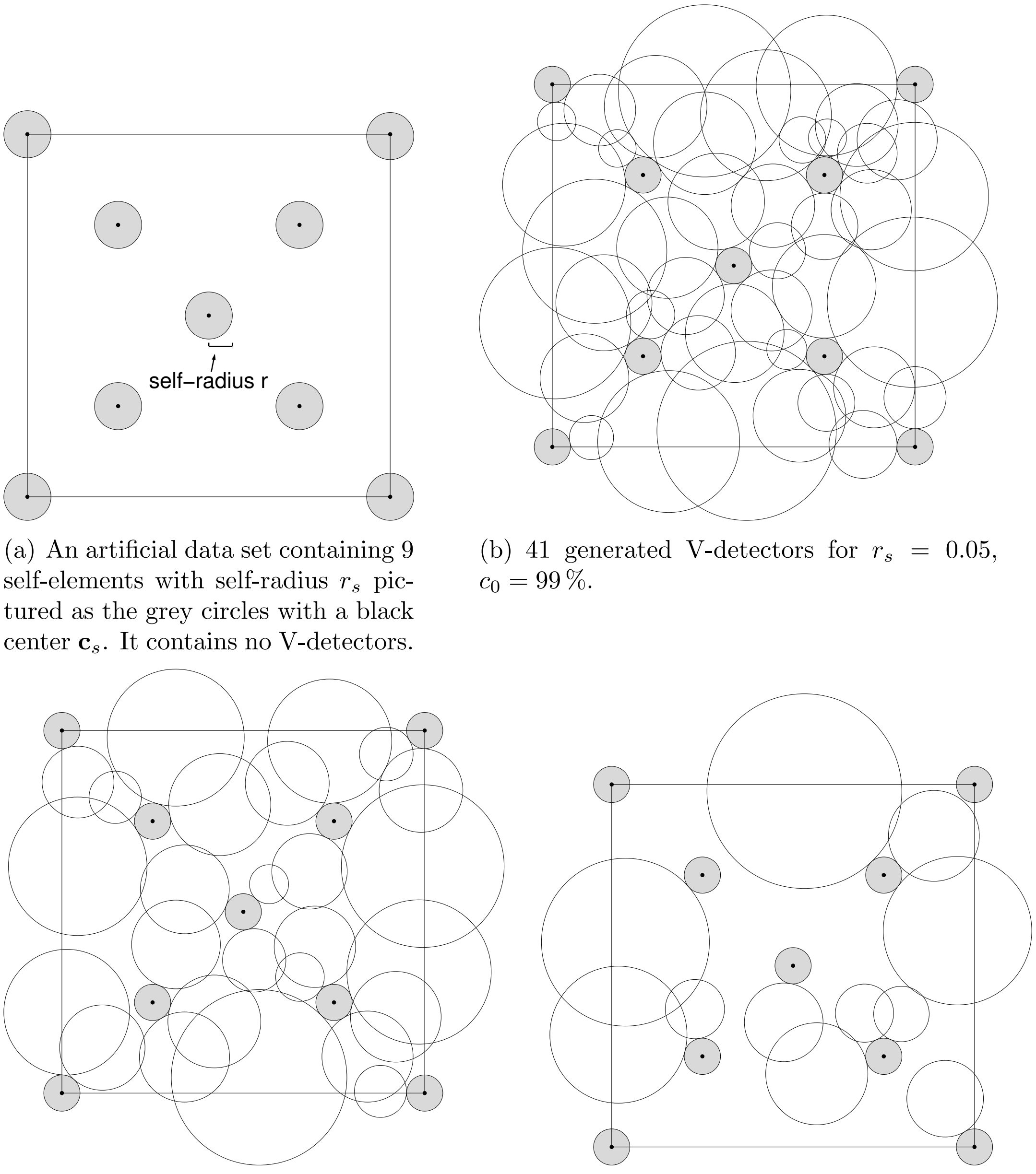
6.2.2 Algorithm Termination

First, it can be seen (algorithm 4), that the termination condition in line 22 is not useful, because T *never* has a value higher than 1. Once increased to 1 (see line 21), T is set to 0 (see line 5) in the same outer repeat loop and therefore, the termination condition in line 22 is never satisfied.

Another algorithm termination is reached (see line 11), when the condition $t \geq 1/(1 - c_0)$ is satisfied. Let $\mathbf{x}$ be a random sample and $\mathbf{x} \in \Delta$ denote that $\mathbf{x}$ is covered by at least one detector. The variable t is only increased when $\mathbf{x} \in \Delta$ (see line 9). When a random sample $\mathbf{x} \notin \Delta$ is chosen, and falls within a self-element circle or an uncovered gap, then t is set 0 (see line 4). Therefore, the termination criteria is guaranteed, when a sample sequence $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_j \in \Delta$ of length j is found, where $j = t/\delta$. The term δ denotes the average number of detectors covering a sample $\mathbf{x}$. The justification behind

[3] Each visualization presented in figures 6.1(b)-6.1(d) is obtained by one algorithm run

[4] In the original work [40], the runtime complexity of the V-Detector algorithm is estimated by $O(|D| \cdot |S|)$ *without* a probabilistic approach. By applying a probabilistic sampling technique, there is always a trade-off between accuracy of the solution and the runtime complexity



(a) An artificial data set containing 9 self-elements with self-radius r_s pictured as the grey circles with a black center $\mathbf{c}_s$. It contains no V-detectors.

(b) 41 generated V-detectors for $r_s = 0.05$, $c_0 = 99\,\%$.

(c) First independent algorithm run which generated 26 V-detectors for $r_s = 0.05$, $c_0 = 80\,\%$.

(d) Second independent algorithm run which generated 11 V-detectors for $r_s = 0.05$, $c_0 = 80\,\%$.

Figure 6.1: The real-valued negative selection algorithm with variable-sized detectors applied on an artificial data set for different estimated coverages.

δ is that a sample $\mathbf{x}$ can be covered by more than one detector, because the detectors can overlap and therefore the variable t can be increased multiple times. The probability of finding a sequence of length j, can be calculated with the geometric distribution and the approach $\mathbf{x}_{j+1} \notin \Delta$.

The probability to find in $j+1$ random sampling trials j successes before the first failure is :

$$P(\mathbf{x}_{j+1} \notin \Delta) \quad = \quad p(1-p)^j \tag{6.1}$$

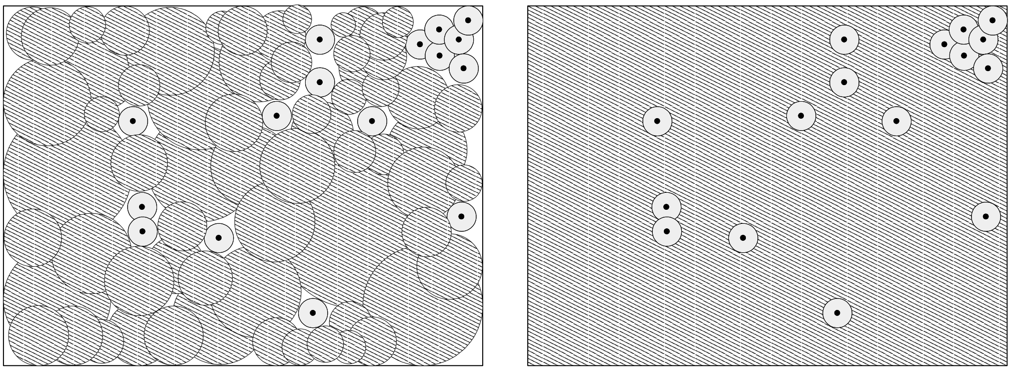
where p is the probability that a random sample $\mathbf{x}$ is covered by at least one detector. Term 6.1 only depends on p and j. The higher the number of self elements or the larger the self-radius, the lesser the probability of finding a sample sequence which guarantees the algorithm termination. Furthermore, the probability is strongly biased by parameter c_0. A higher confidence of the estimated coverage c_0 decreases the probability of finding a termination sample sequence and therefore increases the runtime complexity. This is a typical property of probabilistic Monte Carlo algorithms [46] (see appendix A.3, A.4).

6.3 Real-Valued Positive Selection

The real-valued positive selection (see algorithm 5 and figure 6.2(b)) also referred to as self-detector classification was proposed by Stibor et al. [61] and uses like the real-valued negative selection detectors (hyperspheres) for classifying elements. It is a very trivial algorithm and was only proposed for comparative studies. Each self-element is a self-detector with a self-radius r_s. An unseen element which lies within the self-detector is classified as self, otherwise as non-self. This means, that no detector generation phase is necessary, but the classification decision for each unseen element is computationally expensive, as in the worst-case the distance to each self-detector must be calculated. For very large data sets this result in an infeasible computationally complexity.

Algorithm 5: Generate Self-Detector Set

```
input  : $S$ = Set of self elements, $r_s$ = self-radius
output: $D$ = Set of generated Self-Detectors
begin
    $D \longleftarrow \emptyset$
    foreach $\mathbf{s} \in S$ do
        $d_s \leftarrow (\mathbf{s}, r_s)$
        $D \longleftarrow D \cup d_s$
end
```



(a) Real-valued negative selection principle with variable size detectors for a two-dimensional space. The non-self space which is covered by the detectors is pictured as the shaded area. The self elements are pictured as white circles with a black center

(b) Real-valued positive selection for a two-dimensional space. The covered non-self space is pictured as the shaded area. The self elements are pictured as grey circles with a center and a self-radius r_s

Figure 6.2: Geometric interpretation of real-valued negative/positive selection

6.4 Review of Real-Valued Negative/Positive Selection

Geometrically speaking (compare figure 6.2(a) and 6.2(b)), the real-valued negative selection method first fills the space with detector-circles[5] (shaded area) and considers elements as non-self if they lie within a detector-circle. In contrast, the real-valued positive selection method considers the complete space as non-self, with exception of elements which lie within a self-detector circle.

The main parameter which influences the classification performance and enables the learning capability (generalization) is the self-radius r_s, which is used in both methods. The radius r_s strongly depends on the *probability density function* of the data set, which is unknown *a-priori*. An improper chosen radius r_s consequently results in a weak classification performance. To estimate a proper radius with only the training data from one class, a coherence between estimated probability density function and radius must be found. In section 7.2, we present a method of finding an optimal self radius, by given some elements of the *anomalous* class.

Another problem is how to find an optimal distribution of the detectors, i.e. the minimum number of detectors covering the maximum possible non-self space. This is a hard combinatorial problem, which is solved in [31] with the simulating annealing technique. As a consequence, a vast amount of time is needed to generate and to position the detectors to cover the non-self space.

Following Occam's Razor principle which states : *"Entities should not be multiplied unnecessarily"* or a more useful statement for scientists : *"When you have two competing theories which make exactly the same predictions, the one that is simpler is the better"*, we believe that the negative selection approach provides no benefits, when compared to the positive selection approach.

From our point of view it makes more sense to formulate the classification model based on self deviation, rather than in the complementary non-self space. This is also mentioned in Roberts work [53] :

"it is better to formulate a model of the 'normal' data and test for 'novel' data against this model"

and in Tarassenko et al. work [64] :

[5] hyperspheres for higher dimensions

"We have been exploring an alternative approach in which we attempt to learn a description of normality using the large number of available mammograms which do not show any evidence of mass-like structures. The idea is then to test for novelty against this description in order to try and identify candidate masses in previously unseen images."

In chapter 8 we show the most fundamental limitations of real-valued negative selection for high-dimensional classification problems. The reason for that are not algorithmic complexity problems as with in the Hamming negative selection (section 5.8), but adverse properties of hyperspheres in high-dimensional spaces.

6.5 Summary

In this chapter the real-valued negative selection with variable-sized detectors was explored. In contrast to the Hamming negative selection each element is represented as a n-dimensional point with a center and a radius. The detectors are therefore hyperspheres. When an element lies within a hypersphere, it is classified as an anomalous point, otherwise as a normal point. The variable-sized real-valued negative selection algorithm randomly generates hyperspheres which do not cover any normal points. The algorithm terminates when a certain proportion of the space is covered by the hyperspheres (detectors).

We have investigated the termination behavior of the algorithm and have additionally explored the sampling technique which is applied to estimate the hypersphere coverage. The analysis reveal, that the termination behavior of the algorithm was incorrectly specified. Furthermore, our results have shown that the runtime complexity of the algorithm depends on the probability of finding a suitable hypersphere and the accuracy of the estimated space coverage. Moreover, an algorithm visualization has revealed that the sampling technique to estimate the hypersphere coverage is insufficient.

Inspired by the Occam's Razor principle and the statements of researches in the field of anomaly detection [53, 64, 18], we have proposed a straightforward positive selection approach. Instead of filling the (non-self) space with hyperspheres, each self element is a self-detector. An element which lies within the self-detector is classified as a normal point, otherwise as an anomalous point. In the next chapter the classification performance of the real-valued positive selection, real-valued negative selection and the statistical anomaly detection techniques are compared for low- and high-dimensional data sets.

Chapter 7

Classification Results and Comparative Study

In this chapter the classification performance of real-valued negative selection with variable-sized detectors (algorithm 4) is compared to real-valued positive selection (algorithm 5) and to the statistical techniques described in section 4.1. For quantifying the classification (detection) performance, we first motivate a technique called ROC analysis, which is commonly used in the field of machine learning [24].

7.1 ROC Analysis

ROC (Received Operating Characteristic) analysis is introduced in signal detection theory to describe how well a receiver can distinguish a signal from noise or more generally, depict a tradeoff between hit rates and false alarm rates of classifiers. To motivate the ROC analysis, we give a simple example of a binary classifier and the resulting evaluation problems.

Given a data set of 100 elements, where 99 are non-self and 1 is self. Let c_t be a classifier that always predicts non-self (trivial classifier). The error-rate of c_t is therefore 1 % and it seems to be a very good classifier. Now, the data set again contains 100 elements, but 99 of them are self and 1 is non-self. In this case, the classifier has an error rate of 99 % and is unacceptable. This problem arises, because we do not know the class distribution and the *context* or *skew* which determines the goodness of a classifier. To measure the real performance of classifiers, ROC analysis is an appropriate method. Given a classifier and an example, there are four different outcomes. If the example is non-self and it is classified as non-self, it is counted as a *true positive*; if it is classified as self, it is counted as a *false negative*. If the example is self

		Actual	
		P	N
Classifier Predict	P	TP	FP
	N	FN	TN

(a) General Confusion Matrix

		Actual	
		Non-Self	Self
$\text{C}_{\text{trivial}}$ Predict	Non-Self	99	1
	Self	0	0

(b) Self/Non-Self Confusion Matrix for the trivial classifier c_t

Figure 7.1: ROC analysis Confusion Matrix

and it is classified as self, it is counted as a *true negative*; if it is classified as non-self, it is counted as a *false positive*. Given a classifier and a set of test examples, a 2×2 *confusion matrix* can be constructed representing the dispositions of the set of examples (see Fig. 7.1). Using the matrix values, the *detection rate* (true positive rate) and the *false alarm rate* (false positive rate) of a classifier is calculated as :

$$\begin{aligned} \textit{detection rate } (DR) &= \frac{\textit{nonself correctly classified}}{\textit{total nonself}} = \frac{TP}{TP+FN} \\ \textit{false alarm rate } (FAR) &= \frac{self\ incorrectly\ classified}{total\ self} = \frac{FP}{FP+TN} \end{aligned}$$

The trivial classifier c_t has a detection rate of 100%, but a false alarm rate of also 100%. To compare the classification performance of several classifiers, the detection rate and false alarm rate is plotted on a two-dimensional graph (ROC space), where the detection rate is plotted on the ordinate and the false alarm rate on the abscissa. Informally, one point in the ROC space is better than another if it is northwest (detection rate is higher, false alarm rate is lower, or both) of the first. The point $\widehat{p} = (0, 1)$ represents a perfect classifier (100% detection rate and 0% false alarm rate), where a random classifier[1] yields a point which always lies on the dashed line (see Fig. 7.1). In figure 7.1 three classifiers are depicted. The trivial classifier c_t has the weakest classification performance, c_1 the highest performance.

7.2 Determining Optimal Self-Radius

The ROC analysis provides a technique to evaluate the classification performance of classifiers. To find a self-radius r_s which yields the overall best

[1] toss a coin to predict self or non-self

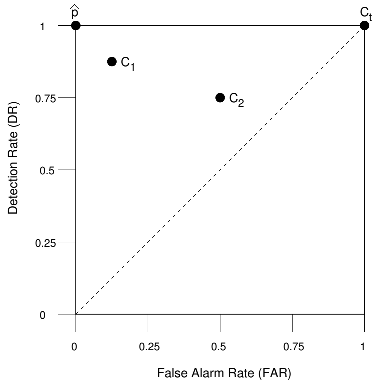


Figure 7.1: A ROC graph showing three classifiers c_1, c_2, c_t, where c_1 has the highest classification performance and c_t the lowest. The diagonal dashed line represents the strategy of randomly guessing a class. The point $\widehat{p} = (0, 1)$ represents a perfect classifier.

balance between detection rate and false alarm rate, r_s is initialized with a small start value (i.e. 0.01) and increased after one training classification run by $r_s = r_s + \Delta i$ until $r_s \geq \max$ (i.e $\max = 1.0$). For every Δi step, the resulting false alarm rate f_i and detection rate d_i yield a point p_i in the ROC space, which results in a ROC curve (see figures 7.2,7.3,7.4). Then a radius r_s is chosen which yields the minimum error, this results in an overall best balance between detection rate and false alarm rate.

$$\text{minimum error} \;=\; \min\left(1 - (d_i - f_i)\right), \quad \forall i \tag{7.1}$$

It has not escaped our notice, that this radius determination requires of course a second class, and is similar to a cross-validation [32]. The cross-validation is a widely used method to estimate how well the classification model was learned from some training data and is going to perform on unseen test data. More concrete it is a method for estimating the prediction of the generalization error and there the cross-validation is used for determining the optimal model parameter, i.e. the parameter values which gives the lowest prediction error. In figures 7.2, 7.3, 7.4 the ROC curves obtained by algorithm 5 are plotted for variable amount of training samples — the data sets are described in section 7.3. One can see the coherence between radius length and the amount of training examples. The smaller the amount of training examples, the larger the radius. The radius length can be considered as a measure of uncertainty, which shrinks if many training examples are given (uncertainty is small) and grows if few training examples are given (uncertainty is high).

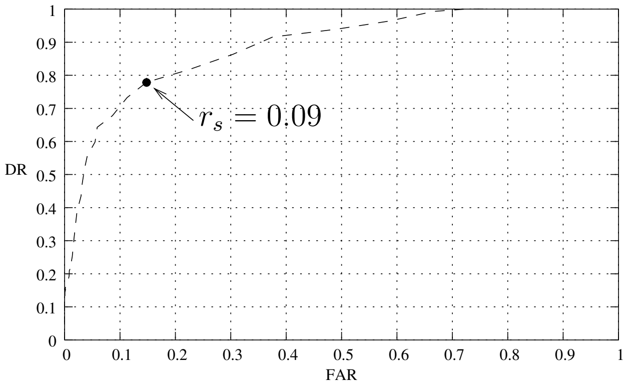


Figure 7.2: ROC curve of Biomedical data set for 25% training data, generated by an increasing self-radius in $\Delta = 0.01$ steps. Self-radius $r_s = 0.09$ results in an overall best balance, with a detection rate of 0.7787, a false alarm rate of 0.1480 and a minimum error of 0.3693

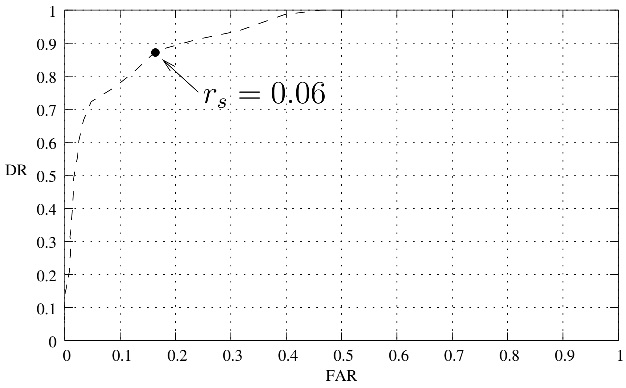


Figure 7.3: ROC curve of Biomedical data set for 50% training data, generated by an increasing self-radius in $\Delta = 0.01$ steps. Self-radius $r_s = 0.06$ results in an overall best balance, with a detection rate of 0.8766, a false alarm rate of 0.1653 and a minimum error of 0.2887

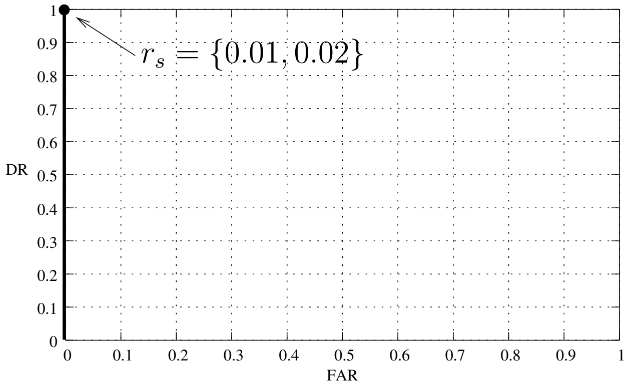


Figure 7.4: ROC curve of Biomedical data set for 100% training data, generated by an increasing self-radius in $\Delta = 0.01$ steps. Self-radii $r_s = \{0.01, 0.02\}$ result in an overall best balance, with a detection rate of 1.0, a false alarm rate of 0.0 and a minimum error of 0.0

7.3 Low-Dimensional Data Sets and Experimental Settings

In this section, we present classification results performed with real-valued positive selection, one-class SVM and Parzen-Window Estimator. The results obtained are compared to results performed with real-valued negative selection (algorithm 4) and were reported in [40].

In line with previous work [40], we perform our experiments on the Iris-Fisher and Biomedical data sets, which can be downloaded from [69]. The Iris Fisher data set contains 3 classes of 50 instances, where each class refers to a type of iris plant. The Biomedical data set contains 209 observations (134 for "normals" and 75 for "carriers") of blood measurements to identify carriers of a rare genetic disorder. Each blood sample consists of four measurements and a label (normal or carrier). Both data sets are normalized in the unitary hypercube $[0, 1]^n$ using the *min-max normalization.* The Biomedical data set contains 15 data vectors which contain undefined points — we omit these data vectors.

In the first experiment, the classifiers are trained with 100% of one class (considered as self), where the remaining classes are considered as non-self. In the test phase, all elements in the data set must be classified, either as self or non-self. As the goal of learning is to be able to classify unseen data, the classifiers are trained in the second experiment with 50% and 25%[2] of randomly drawn elements from one class. In the test phase, all elements in

[2] only Biomedical data set

the data set are presented to the classifier and must be classified. Each run is repeated 100 times and the results are averaged. The experiments were performed for each class of the Iris-Fisher and Biomedical data set. In the Biomedical data set, only non-carriers were considered as self elements.
In order for our results to be comparable to Ji's and Dasgupta's [40] results, we performed the same experiments and used the same radius length $r_s = 0.1$ for the Iris-Fisher data set and radii length $r_s = \{0.05, 0.1\}$ for the Biomedical data set. Additionally, we performed classification runs with radius r_s, which gives the minimum error for 50% Iris Fisher training data ($r_s = 0.15$) and 25% Biomedical training data ($r_s = 0.09$), obtained by Eq. (7.1). By increasing the radius, it is possible to achieve a lower false alarm rate, but this will consequently decrease the detection rate. This is demonstrated using $r_s = 0.13$ on the Biomedical data set.

The one-class SVM experiments were performed with following kernels :

$$\begin{aligned}
\text{Linear} \quad k_0(\mathbf{u},\mathbf{v}) &= (\mathbf{u}\cdot\mathbf{v}) \\
\text{Polynomial} \quad k_1(\mathbf{u},\mathbf{v}) &= ((\mathbf{u}\cdot\mathbf{v}) + \Theta)^d \\
\text{Gaussian radial basis function} \quad k_2(\mathbf{u},\mathbf{v}) &= \exp\left(-\frac{\|\mathbf{u}-\mathbf{v}\|^2}{c}\right) \\
\text{Sigmoidal} \quad k_3(\mathbf{u},\mathbf{v}) &= \tanh(\kappa(\mathbf{u}\cdot\mathbf{v}) + \Theta)
\end{aligned}$$

and defaults parameters $c, \kappa, \Theta \in \mathbb{R}$ and varied parameters $d \in \mathbb{N}$ and $\nu = \{0.1, 0.05, 0.01\}$.

For the experiments we used the one-class SVM implementation LIBSVM 2.6 [37, 10]. LIBSVM is a program, which provides several SVM algorithms for classification and regression, including the described one-class SVM.

The Parzen-Window Estimator experiments were performed with variance parameter $\sigma = 0.67$ for Iris-Fisher and $\sigma = 0.69$ for Biomedical data set. These variance parameters are approximated by using the term $\sigma \approx (4/(2d+1))^{1/(d+4)} \cdot n^{-1/(d+4)}$ — where n is the number of data samples and d the dimension of a data sample — proposed in [57].

7.4 Results

For the Iris-Fisher data set (see table 7.1) the straightforward real-valued positive selection (Self-Detector) method outperforms the real-valued negative selection (V-Detector) in all classification tests for all chosen radii for 100% training data. For 50% training data, the self-detector classification with radius $r_s = 0.1$ outperforms the real-valued negative selection in the detection rate, but it has a higher false positive rate. With radius $r_s = 0.15$, it performs better for both rates, except the false alarm rate in the setosa class which differ by 0.56%.

The moderate classification accuracy of the one-class SVM (see tables 7.2,7.3,7.4,7.5,7.6,7.7) can be explained by the fact that the SVM was trained with too few examples to capture the density. Additionally, it has been observed, that the four different kernels k_0, k_1, k_2, k_3 produce very different classification results. The RBF kernel (k_2) results in the overall highest classification accuracy. Whereas the linear (k_0) and sigmoid (k_3) kernel produce high accuracy for the setosa and virginica class, but very poor results for the versicolor class. Furthermore, one can observe that the polynomial kernel (k_1) produces poor results for even polynomial degrees ($d = \{2, 4\}$), whereas degree $d = 3$ results in higher accuracy compared with the even polynomial degrees.

The Parzen-Window Estimator technique produced the overall poorest classification results compared with the real-valued positive selection, real-valued negative selection and the one-class SVM with RBF kernel. In order to produce higher classification results the parameter σ and p_u were empirically tuned, but without success. The results shown in table 7.8 are the best obtained with the Parzen-Window Estimator technique. The Parzen-Window technique requires a certain number of samples to capture the underlying density function. The Iris-Fisher data set contains a limited number of samples and therefore strongly bias the classification performance of the Parzen-Window technique. We believe that the number of samples is too small to capture the underlying density and therefore this technique results in poor results.

For the Biomedical data set (see table 7.9) the real-valued positive selection when compared with the real-valued negative selection, produces higher detection rates, but also higher false alarm rates. By applying the ROC analysis one can verify that the real-valued positive selection produces an overall higher classification accuracy. Furthermore, it is observed that the one-class SVM produces for the Biomedical data set similar classification results for each tested kernel k_0, k_1, k_2, k_3 — for the Iris-Fisher data set results vary a great deal with kernels k_0, k_1, k_2, k_3. Moreover, the one-class SVM produced overall high classification results for the Biomedical data set. The Parzen-Window Estimator technique produced the overall poorest results compared with the other techniques. As for the Iris-Fisher data set, the parameters were additionally tuned "by hand" for discovering the highest classification accuracy. These best found values and the associated classification results are shown in table 7.14.

Table 7.1: Classification Results with real-valued negative/positive selection for Fisher Iris data set

Training Data	Algorithm	Detection Rate		False Alarm Rate	
		Mean	SD	Mean	SD
Setosa 100 %	V-Detector$^{r=0.1}$	99.98	0.14	0.00	0.00
	Self-Detector$^{r=0.1}$	100	0.00	0.00	0.00
	Self-Detector$^{r=0.15}$	100	0.00	0.00	0.00
Setosa 50 %	V-Detector$^{r=0.1}$	99.97	0.17	1.32	0.95
	Self-Detector$^{r=0.1}$	100	0.00	9.98	3.18
	Self-Detector$^{r=0.15}$	100	0.00	1.88	1.73
Versicolor 100 %	V-Detector$^{r=0.1}$	85.95	2.44	0.00	0.00
	Self-Detector$^{r=0.1}$	98.00	0.00	0.00	0.00
	Self-Detector$^{r=0.15}$	87.00	0.00	0.00	0.00
Versicolor 50 %	V-Detector$^{r=0.1}$	88.3	2.77	8.42	2.12
	Self-Detector$^{r=0.1}$	99.03	0.99	15.72	3.55
	Self-Detector$^{r=0.15}$	92.02	3.06	3.78	3.04
Viginica 100 %	V-Detector$^{r=0.1}$	81.87	2.78	0.00	0.00
	Self-Detector$^{r=0.1}$	99.00	0.00	0.00	0.00
	Self-Detector$^{r=0.15}$	91.00	0.00	0.00	0.00
Viginica 50 %	V-Detector$^{r=0.1}$	93.58	2.33	13.18	3.24
	Self-Detector$^{r=0.1}$	99.09	0.30	26.36	4.23
	Self-Detector$^{r=0.15}$	93.72	2.28	9.98	3.76

Table 7.2: Classification Results with one-class SVM, kernel k_0 (linear) for Iris Fisher data set

Training Data	Algorithm Parameters	Detection Rate		False Alarm Rate	
		Mean	SD	Mean	SD
Setosa 100 %	$k_0, \nu = 0.1$	100	0.00	10.66	0.94
	$k_0, \nu = 0.05$	100	0.00	2.00	0.00
	$k_0, \nu = 0.01$	100	0.00	3.86	0.80
Setosa 50 %	$k_0, \nu = 0.1$	100	0.00	13.02	5.96
	$k_0, \nu = 0.05$	100	0.00	8.68	3.84
	$k_0, \nu = 0.01$	100	0.00	5.12	3.68
Versicolor 100 %	$k_0, \nu = 0.1$	50.00	0.00	12.14	1.45
	$k_0, \nu = 0.05$	50.00	0.00	5.08	1.07
	$k_0, \nu = 0.01$	50.00	0.00	3.88	1.65
Versicolor 50 %	$k_0, \nu = 0.1$	50.25	0.67	13.48	6.13
	$k_0, \nu = 0.05$	50.17	0.58	9.90	4.74
	$k_0, \nu = 0.01$	50.21	0.68	10.00	4.54
Virginica 100 %	$k_0, \nu = 0.1$	100	0.00	8.00	0.00
	$k_0, \nu = 0.05$	97.00	0.00	6.00	0.00
	$k_0, \nu = 0.01$	96.00	0.00	0.08	0.39
Virginica 50 %	$k_0, \nu = 0.1$	98.88	1.20	12.50	4.42
	$k_0, \nu = 0.05$	97.76	1.83	8.60	4.52
	$k_0, \nu = 0.01$	97.95	1.67	6.50	4.72

Table 7.3: Classification Results with one-class SVM, kernel k_1 (polynomial), degree $d = 2$ for Iris Fisher data set

Training Data	Algorithm Parameters	Detection Rate		False Alarm Rate	
		Mean	SD	Mean	SD
Setosa 100 %	$k_1, d = 2, \nu = 0.1$	95.00	0.00	8.76	1.05
	$k_1, d = 2, \nu = 0.05$	93.00	0.00	4.04	0.28
	$k_1, d = 2, \nu = 0.01$	93.00	0.00	0.58	0.91
Setosa 50 %	$k_1, d = 2, \nu = 0.1$	94.28	1.02	12.36	6.22
	$k_1, d = 2, \nu = 0.05$	93.45	0.79	8.34	3.66
	$k_1, d = 2, \nu = 0.01$	93.23	0.66	5.26	4.16
Versicolor 100 %	$k_1, d = 2, \nu = 0.1$	0.00	0.00	12.84	8.99
	$k_1, d = 2, \nu = 0.05$	0.06	0.60	23.42	8.66
	$k_1, d = 2, \nu = 0.01$	11.54	19.15	54.34	15.47
Versicolor 50 %	$k_1, d = 2, \nu = 0.1$	0.02	0.2	24.74	9.89
	$k_1, d = 2, \nu = 0.05$	1.48	3.87	35.44	13.43
	$k_1, d = 2, \nu = 0.01$	22.47	28.41	63.62	14.75
Virginica 100 %	$k_1, d = 2, \nu = 0.1$	49.81	0.39	11.02	1.59
	$k_1, d = 2, \nu = 0.05$	47.00	0.00	6.00	0.00
	$k_1, d = 2, \nu = 0.01$	45.92	1.16	3.02	2.19
Virginica 50 %	$k_1, d = 2, \nu = 0.1$	49.08	1.16	13.68	5.94
	$k_1, d = 2, \nu = 0.05$	47.94	1.94	8.76	5.04
	$k_1, d = 2, \nu = 0.01$	48.10	1.70	10.82	5.40

Table 7.4: Classification Results with one-class SVM, kernel k_1 (polynomial), degree $d = 3$ for Iris Fisher data set

Training Data	Algorithm Parameters	Detection Rate		False Alarm Rate	
		Mean	SD	Mean	SD
Setosa 100 %	$k_1, d = 3, \nu = 0.1$	100	0.00	8.28	0.69
	$k_1, d = 3, \nu = 0.05$	100	0.00	7.88	0.47
	$k_1, d = 3, \nu = 0.01$	100	0.00	3.60	0.80
Setosa 50 %	$k_1, d = 3, \nu = 0.1$	100	0.00	12.18	4.80
	$k_1, d = 3, \nu = 0.05$	100	0.00	9.90	4.75
	$k_1, d = 3, \nu = 0.01$	100	0.00	6.58	3.11
Versicolor 100 %	$k_1, d = 3, \nu = 0.1$	53.96	4.79	49.26	24.11
	$k_1, d = 3, \nu = 0.05$	61.10	9.64	64.52	23.41
	$k_1, d = 3, \nu = 0.01$	69.57	12.46	78.12	18.39
Versicolor 50 %	$k_1, d = 3, \nu = 0.1$	57.86	10.70	55.64	24.18
	$k_1, d = 3, \nu = 0.05$	63.36	13.55	67.68	21.83
	$k_1, d = 3, \nu = 0.01$	69.65	14.66	76.02	16.81
Virginica 100 %	$k_1, d = 3, \nu = 0.1$	99.49	0.50	10.36	2.45
	$k_1, d = 3, \nu = 0.05$	97.26	1.16	6.34	3.08
	$k_1, d = 3, \nu = 0.01$	97.67	1.94	16.58	8.62
Virginica 50 %	$k_1, d = 3, \nu = 0.1$	99.29	1.24	16.48	7.00
	$k_1, d = 3, \nu = 0.05$	98.30	2.25	14.38	8.45
	$k_1, d = 3, \nu = 0.01$	99.23	1.28	30.28	22.42

Table 7.5: Classification Results with one-class SVM, kernel k_1 (polynomial), degree $d = 4$ for Iris Fisher data set

Training Data	Algorithm Parameters	Detection Rate		False Alarm Rate	
		Mean	SD	Mean	SD
Setosa 100 %	$k_1, d = 4, \nu = 0.1$	94.90	0.30	11.60	0.80
	$k_1, d = 4, \nu = 0.05$	93.00	0.00	6.74	1.74
	$k_1, d = 4, \nu = 0.01$	92.99	0.10	2.78	0.98
Setosa 50 %	$k_1, d = 4, \nu = 0.1$	93.98	0.85	11.92	4.41
	$k_1, d = 4, \nu = 0.05$	93.52	0.94	7.86	4.44
	$k_1, d = 4, \nu = 0.01$	93.32	0.93	6.56	4.48
Versicolor 100 %	$k_1, d = 4, \nu = 0.1$	13.39	20.55	48.32	28.25
	$k_1, d = 4, \nu = 0.05$	24.32	28.65	60.50	28.20
	$k_1, d = 4, \nu = 0.01$	34.22	33.96	68.10	23.99
Versicolor 50 %	$k_1, d = 4, \nu = 0.1$	15.64	25.18	51.96	28.26
	$k_1, d = 4, \nu = 0.05$	31.60	34.15	62.84	27.32
	$k_1, d = 4, \nu = 0.01$	52.61	35.51	75.54	17.44
Virginica 100 %	$k_1, d = 4, \nu = 0.1$	49.00	1.60	18.94	12.28
	$k_1, d = 4, \nu = 0.05$	48.11	2.15	21.02	19.58
	$k_1, d = 4, \nu = 0.01$	52.75	8.87	49.90	27.14
Virginica 50 %	$k_1, d = 4, \nu = 0.1$	48.87	2.82	22.92	18.44
	$k_1, d = 4, \nu = 0.05$	52.81	11.12	42.02	28.28
	$k_1, d = 4, \nu = 0.01$	59.16	15.82	58.22	31.23

Table 7.6: Classification Results with one-class SVM, kernel k_2 (RBF) for Iris Fisher data set

Training Data	Algorithm Parameters	Detection Rate		False Alarm Rate	
		Mean	SD	Mean	SD
Setosa 100 %	$k_2, \nu = 0.1$	100	0.00	8.88	0.99
	$k_2, \nu = 0.05$	100	0.00	4.02	0.20
	$k_2, \nu = 0.01$	100	0.00	4.00	0.00
Setosa 50 %	$k_2, \nu = 0.1$	100	0.00	13.82	5.54
	$k_2, \nu = 0.05$	100	0.00	5.72	4.36
	$k_2, \nu = 0.01$	100	0.00	7.24	5.20
Versicolor 100 %	$k_2, \nu = 0.1$	92.00	0.00	12.06	1.56
	$k_2, \nu = 0.05$	90.00	0.00	8.36	1.40
	$k_2, \nu = 0.01$	89.01	0.10	3.44	1.82
Versicolor 50 %	$k_2, \nu = 0.1$	92.46	2.21	14.98	5.21
	$k_2, \nu = 0.05$	90.84	2.71	8.56	5.36
	$k_2, \nu = 0.01$	90.59	2.62	8.34	5.34
Virginica 100 %	$k_2, \nu = 0.1$	83.00	0.00	8.00	0.00
	$k_2, \nu = 0.05$	75.00	0.00	5.36	0.90
	$k_2, \nu = 0.01$	68.00	0.00	2.94	1.00
Virginica 50 %	$k_2, \nu = 0.1$	86.26	7.41	13.76	6.22
	$k_2, \nu = 0.05$	78.58	9.40	9.64	4.22
	$k_2, \nu = 0.01$	78.21	9.33	9.10	5.10

Table 7.7: Classification Results with one-class SVM, kernel k_3 (sigmoid) for Iris Fisher data set

Training Data	Algorithm Parameters	Detection Rate		False Alarm Rate	
		Mean	SD	Mean	SD
Setosa 100 %	$k_3, \nu = 0.1$	100	0.00	11.46	1.47
	$k_3, \nu = 0.05$	100	0.00	3.32	1.82
	$k_3, \nu = 0.01$	100	0.00	0.4	0.80
Setosa 50 %	$k_3, \nu = 0.1$	100	0.00	11.72	5.05
	$k_3, \nu = 0.05$	100	0.00	7.04	3.74
	$k_3, \nu = 0.01$	100	0.00	5.56	4.15
Versicolor 100 %	$k_3, \nu = 0.1$	50.00	0.00	11.86	1.53
	$k_3, \nu = 0.05$	50.00	0.00	4.58	1.75
	$k_3, \nu = 0.01$	50.00	0.00	6.28	2.16
Versicolor 50 %	$k_3, \nu = 0.1$	50.26	0.66	13.94	5.40
	$k_3, \nu = 0.05$	50.31	0.82	10.84	4.31
	$k_3, \nu = 0.01$	50.26	0.78	13.84	5.85
Virginica 100 %	$k_3, \nu = 0.1$	100	0.00	11.80	0.60
	$k_3, \nu = 0.05$	97.00	0.00	6.00	0.00
	$k_3, \nu = 0.01$	96.00	0.00	0.88	0.99
Virginica 50 %	$k_3, \nu = 0.1$	99.07	1.26	13.66	5.62
	$k_3, \nu = 0.05$	98.41	1.83	9.66	4.98
	$k_3, \nu = 0.01$	97.52	1.68	7.32	3.73

Table 7.8: Classification Results with Parzen-Window Estimator for Iris Fisher data set

Training Data	Algorithm Parameters	Detection Rate		False Alarm Rate	
		Mean	SD	Mean	SD
Setosa 100 %	$\sigma = 0.67, p_u = 0.1$	100.00	0.00	4.00	0.00
Setosa 50 %	$\sigma = 0.67, p_u = 0.1$	100.00	0.00	3.84	0.54
Versicolor 100 %	$\sigma = 0.67, p_u = 0.1$	79.99	0.00	2.00	0.00
Versicolor 50 %	$\sigma = 0.67, p_u = 0.1$	79.35	1.57	3.18	1.89
Virginica 100 %	$\sigma = 0.67, p_u = 0.1$	79.99	0.00	9.99	0.00
Virginica 50 %	$\sigma = 0.67, p_u = 0.1$	80.18	2.94	11.60	1.85

Table 7.9: Classification Results with real-valued negative/positive selection for Biomedical data set

Training Data	Algorithm	Detection Rate		False Alarm Rate	
		Mean	SD	Mean	SD
Normals 100 %	V-Detector$^{r=0.05}$	40.51	3.92	0.00	0.00
	V-Detector$^{r=0.1}$	30.61	3.04	0.00	0.00
	Self-Detector$^{r=0.05}$	88.05	0.00	0.00	0.00
	Self-Detector$^{r=0.09}$	67.16	0.00	0.00	0.00
	Self-Detector$^{r=0.1}$	59.70	0.00	0.00	0.00
	Self-Detector$^{r=0.13}$	44.77	0.00	0.00	0.00
Normals 50 %	V-Detector$^{r=0.05}$	42.89	3.83	1.07	0.49
	V-Detector$^{r=0.1}$	32.92	2.35	0.61	0.31
	Self-Detector$^{r=0.05}$	91.17	1.60	23.78	2.57
	Self-Detector$^{r=0.09}$	72.39	2.30	4.60	1.76
	Self-Detector$^{r=0.1}$	66.29	2.49	3.39	1.43
	Self-Detector$^{r=0.13}$	50.70	3.36	1.86	0.88
Normals 25 %	V-Detector$^{r=0.05}$	57.97	5.86	2.63	0.77
	V-Detector$^{r=0.1}$	43.68	4.25	1.24	0.50
	Self-Detector$^{r=0.05}$	93.87	1.65	48.44	3.36
	Self-Detector$^{r=0.09}$	78.10	3.67	14.93	3.48
	Self-Detector$^{r=0.1}$	72.85	3.50	11.10	2.74
	Self-Detector$^{r=0.13}$	59.37	4.76	5.05	2.12

Table 7.10: Classification Results with one-class SVM, kernel k_0 (linear) for Biomedical data set

Training Data	Algorithm Parameters	Detection Rate		False Alarm Rate	
		Mean	SD	Mean	SD
Normals 100 %	$k_0, \nu = 0.1$	74.62	0.00	10.23	0.00
	$k_0, \nu = 0.05$	68.65	0.00	6.06	0.36
	$k_0, \nu = 0.01$	41.79	0.00	1.13	0.39
Normals 50 %	$k_0, \nu = 0.1$	75.08	4.68	10.54	3.01
	$k_0, \nu = 0.05$	65.83	7.76	6.25	2.29
	$k_0, \nu = 0.01$	52.04	11.78	2.60	1.13
Normals 25 %	$k_0, \nu = 0.1$	74.22	8.04	12.25	5.03
	$k_0, \nu = 0.05$	69.29	9.41	8.07	4.51
	$k_0, \nu = 0.01$	63.31	12.32	5.68	3.88

Table 7.11: Classification Results with one-class SVM, kernel k_1 (polynomial) and different degrees d for Biomedical data set

Training Data	Algorithm	Detection Rate		False Alarm Rate	
		Mean	SD	Mean	SD
Normals 100 %	$k_1, d = 2, \nu = 0.1$	73.13	0.00	9.73	0.37
	$k_1, d = 2, \nu = 0.05$	68.65	0.00	5.18	0.39
	$k_1, d = 2, \nu = 0.01$	43.28	0.36	2.29	0.22
	$k_1, d = 3, \nu = 0.1$	71.64	0.00	9.00	0.71
	$k_1, d = 3, \nu = 0.05$	61.37	0.48	5.12	0.39
	$k_1, d = 3, \nu = 0.01$	51.86	0.64	2.43	0.22
	$k_1, d = 4, \nu = 0.1$	71.64	0.00	9.55	0.26
	$k_1, d = 4, \nu = 0.05$	59.83	0.60	5.59	0.45
	$k_1, d = 4, \nu = 0.01$	52.44	2.06	2.39	0.33
Normals 50 %	$k_1, d = 2, \nu = 0.1$	73.70	3.92	10.51	2.94
	$k_1, d = 2, \nu = 0.05$	64.28	6.88	6.28	2.12
	$k_1, d = 2, \nu = 0.01$	54.79	11.59	2.85	1.87
	$k_1, d = 3, \nu = 0.1$	73.61	3.25	11.17	2.97
	$k_1, d = 3, \nu = 0.05$	65.43	6.74	6.43	2.35
	$k_1, d = 3, \nu = 0.01$	59.10	11.05	4.56	2.76
	$k_1, d = 4, \nu = 0.1$	72.37	4.30	11.25	2.96
	$k_1, d = 4, \nu = 0.05$	65.02	6.25	6.37	2.28
	$k_1, d = 4, \nu = 0.01$	53.85	11.57	2.47	2.05
Normals 25 %	$k_1, d = 2, \nu = 0.1$	72.55	6.88	11.77	4.63
	$k_1, d = 2, \nu = 0.05$	69.11	7.98	8.35	4.50
	$k_1, d = 2, \nu = 0.01$	63.13	11.88	5.88	3.86
	$k_1, d = 3, \nu = 0.1$	74.32	6.77	13.06	5.33
	$k_1, d = 3, \nu = 0.05$	69.98	7.18	9.37	4.75
	$k_1, d = 3, \nu = 0.01$	62.79	12.91	5.50	4.27
	$k_1, d = 4, \nu = 0.1$	72.77	7.09	13.34	6.07
	$k_1, d = 4, \nu = 0.05$	69.80	6.81	9.41	4.27
	$k_1, d = 4, \nu = 0.01$	62.82	12.66	6.29	5.36

Table 7.12: Classification Results with one-class SVM, kernel k_2 (RBF) for Biomedical data set

Training Data	Algorithm Parameters	Detection Rate		False Alarm Rate	
		Mean	SD	Mean	SD
Normals 100 %	$k_2, \nu = 0.1$	46.26	0.00	10.22	0.00
	$k_2, \nu = 0.05$	40.29	0.00	5.41	0.50
	$k_2, \nu = 0.01$	22.38	0.00	1.45	0.60
Normals 50 %	$k_2, \nu = 0.1$	53.38	9.88	11.20	3.08
	$k_2, \nu = 0.05$	38.58	10.11	6.53	2.62
	$k_2, \nu = 0.01$	33.43	10.76	3.74	1.98
Normals 25 %	$k_2, \nu = 0.1$	54.94	14.80	12.26	4.31
	$k_2, \nu = 0.05$	44.97	15.65	7.86	4.08
	$k_2, \nu = 0.01$	46.58	14.14	8.31	3.76

Table 7.13: Classification Results with one-class SVM, kernel k_3 (sigmoid) for Biomedical data set

Training Data	Algorithm Parameters	Detection Rate		False Alarm Rate	
		Mean	SD	Mean	SD
Normals 100 %	$k_3, \nu = 0.1$	76.11	0.00	10.23	0.00
	$k_3, \nu = 0.05$	68.65	0.00	4.73	0.07
	$k_3, \nu = 0.01$	43.37	1.37	1.09	0.38
Normals 50 %	$k_3, \nu = 0.1$	75.35	4.36	10.45	3.09
	$k_3, \nu = 0.05$	64.01	8.83	5.74	2.12
	$k_3, \nu = 0.01$	57.74	11.57	4.14	2.42
Normals 25 %	$k_3, \nu = 0.1$	75.16	7.17	13.20	5.05
	$k_3, \nu = 0.05$	68.53	10.61	7.72	4.60
	$k_3, \nu = 0.01$	60.68	13.36	5.65	4.99

Table 7.14: Classification Results with Parzen-Window Estimator for Biomedical data set

Training Data	Algorithm Parameters	Detection Rate		False Alarm Rate	
		Mean	SD	Mean	SD
Normals 100 %	$\sigma = 0.69, p_u = 0.1$	59.70	0.00	12.59	0.00
Normals 50 %	$\sigma = 0.69, p_u = 0.1$	61.80	4.98	12.72	0.85
Normals 25 %	$\sigma = 0.69, p_u = 0.1$	61.73	17.18	12.76	1.77

7.5 High-Dimensional Data Set and Experimental Settings

In this section we explore the anomaly detection effectiveness of the real-valued negative selection on a high-dimensional data set. Furthermore, the real-valued positive selection and the statistical anomaly detection techniques are benchmarked, to obtain a comparative evaluation. For our experiments, we made use of the dataset taken from KDD Cup 1999 [34]. This data set contains a wide variety of network intrusions and normal network traffic. The data set consists of connection-based network traffic data, where each record corresponds to one network connection. A network connection is a sequence of Internet packets sent during a period of time between two IP addresses. A complete record is described as a network connection vector which contains 38 continuous and 3 symbolic fields and an end-label (attack type or normal behavior).

Example 7.1.
```
0,icmp,ecr_i,SF,1032,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,511,511,0.00,0.00,0.00,0.00,1.00,0.00,0.00,
255,243,0.95,0.01,0.95,0.00,0.00,0.00,0.00,0.00,smurf
```

Example 7.2.
```
0,tcp,http,SF,239,968,0,0,0,0,0,1,0,0,0,0,0,0,0,0,
0,0,3,3,0.00,0.00,0.00,0.00,1.00,0.00,0.00,3,239,
1.00,0.00,0.33,0.03,0.00,0.00,0.00,0.00,normal
```

Example 7.1 shows a connection vector which characterizes a Denial of Service (short DoS) attack. A DoS attack is an attack on a computer system, or network, that causes a loss of service to users by consuming the bandwidth of the victim network or overloading the computational resources of the victim system. As a concrete example 7.1 characterizes a smurf DoS attack which uses spoofed broadcast icmp messages to flood a target system. In contrast, example 7.2 shows a connection vector which characterizes a "normal" access to a HTTP server. The complete KDD dataset contains 3925650 abnormal $(80,14\%)$ and 972780 normal $(19,86\%)$ connection vectors and has a total size of ca. 700 mb. The abnormal samples are partitioned in four categories :

- DOS ($\approx 98,92\,\%$) : denial-of-service, e.g. syn flood.
- R2L ($\approx 0,0286\,\%$) : unauthorized access from a remote machine, e.g. guessing password.
- U2R ($\approx 0,0013\,\%$) : unauthorized access to local superuser (root) privileges, e.g., various "buffer overflow" attacks.
- probing ($\approx 1,05\,\%$) : surveillance and other probing, e.g., port scanning.

Due to the high runtime complexity of the Parzen-Window method and the real-valued positive selection, our experiments were performed on a reduced dataset. More precisely, we randomly created 20 subsets $S_1, \ldots, S_{20}$ from the complete KDD dataset. Each subset S_i contains randomly determined $1\,\%$ of normal and $1\,\%$ of anomalous data from the whole KDD dataset. There are 39256 anomalous and 9727 normal connection vectors in each subset. Furthermore, each discriminative symbolic string is mapped onto a natural number, i.e. icmp → 0, tcp → 1, udp → 2, and so on. The dataset is then normalized in the unitary hypercube $[0, 1]^{41}$ using the *min-max normalization.*

Each classification method is trained from subset S_i with normal samples *only.* The test run is performed on the whole subset S_i (normal and anomalous samples). After performing all 20 classification runs for each subset $S_1, \ldots, S_{20}$, the mean detection rate, mean false alarm rate and the standard deviations were recorded and are presented in table 7.15.

As the real-valued negative selection is the only method which has a random behavior[3] each run of the algorithm was repeated 20 times for each subset S_i.

The parameters for the real-valued negative selection were chosen as outlined in [40] ($MSC = 99.99\,\%$, $T_{max} = 1000$, $c_0 = 99\,\%$). Initial experiments with real-valued negative selection were performed with self-radius $r_s = 0.1$ and $r_s = 0.05$. For this radius, the algorithm produces very poor classification results. Therefore, several "empirical radius searching" runs were

[3] generates detectors randomly

performed to find an effective self-radius. The radius lengths shown in table 7.15 resulted in the best classification performance. These radius lengths are also used for the positive selection algorithm. For the one-class SVM we used the RBF kernel (k_2), as it shows a robust and high overall classification accuracy in the Iris-Fisher and Biomedical data sets experiments. For the Parzen-Window Estimator we used the proposed variance parameter $\sigma = 0.01$ [73].

7.6 Results

Table 7.15: Classification Results for KDD dataset

Algorithm	Detection Rate		False Alarm Rate		# Detectors or # Support Vectors	
	Mean	SD	Mean	SD	Mean	SD
V-Detector$^{r_s=0.000005}$	2.66	8.35	0.00	0.00	1.37	0.52
V-Detector$^{r_s=0.00001}$	2.40	7.12	0.00	0.00	1.36	0.51
V-Detector$^{r_s=0.00005}$	1.75	6.05	0.00	0.00	1.39	0.56
V-Detector$^{r_s=0.0001}$	1.58	5.73	0.00	0.00	1.33	0.50
V-Detector$^{r_s=0.05}$	1.21	4.59	0.00	0.00	1.48	0.59
V-Detector$^{r_s=0.1}$	0.65	3.46	0.00	0.00	1.59	0.67
Self-Detector$^{r_s=0.000005}$	100	0.00	0.00	0.00	9727	0
Self-Detector$^{r_s=0.00001}$	100	0.00	0.00	0.00	9727	0
Self-Detector$^{r_s=0.00005}$	100	0.00	0.00	0.00	9727	0
Self-Detector$^{r_s=0.0001}$	100	0.00	0.00	0.00	9727	0
Self-Detector$^{r_s=0.05}$	100	0.00	0.00	0.00	9727	0
Self-Detector$^{r_s=0.1}$	99.99	0.02	0.00	0.00	9727	0
ocSVM$^{k_2,\nu=0.005}$	99.78	0.03	0.05	0.02	55.70	1.56
ocSVM$^{k_2,\nu=0.01}$	99.82	0.02	0.99	0.02	103.40	1.50
ocSVM$^{k_2,\nu=0.05}$	99.87	0.02	4.95	0.03	491.15	1.27
Parzen-Window$^{p_u=0.005,\sigma=0.01}$	99.93	0.02	0.00	0.00	—	—
Parzen-Window$^{p_u=0.01,\sigma=0.01}$	99.93	0.02	0.00	0.00	—	—
Parzen-Window$^{p_u=0.05,\sigma=0.01}$	99.93	0.02	0.00	0.00	—	—

In table 7.15 one can see that real-valued positive selection (Self-Detector) method yields the highest detection rate and the lowest false alarm rate. A benefit of this method is that no training phase is required, and a nearly zero standard deviation of the detection rate for each threshold r_s is achieved. However, this method is computationally very expensive, due to the fact that the Euclidean distance is calculated from a sample to each self-element. The Parzen-Window method yields likewise, a high detection rate and a low false alarm rate. This method also requires no training phase and has a very low standard deviation of the detection rate. However, this method is computationally expensive[4], because each training sample has to calculate the class conditionally probability for a test sample. The one-class SVM achieves similar high detection rates and low false alarm rates. Through the application of the default radial basis kernel, the test data is nearly optimally separable in high-dimensional feature space. This is shown by the fraction of outliers compared to the false alarm rate. For $\nu = 5\%$ outliers, the false alarm rate is nearly 5%. For $\nu = 0.5\%$ outliers, the false alarm rate is 0.5%. The main advantage of the one-class SVM, in comparison with the Parzen-Window method, is the low computational complexity to classify new elements. The one-class SVM considers only a subset of the training samples — the support vectors — to classify new elements. Results reveal that the real-valued negative selection with variable-sized detectors is not competitive to the statistical techniques and to the real-valued positive selection for this high-dimensional classification problem. It has a very low detection rate and a very high standard deviation — the standard deviation is far higher than the mean, and therefore vary a great deal in the classification performance. Though the V-Detector parameter c_0 is 99 %, the estimated coverage method (see line 11) seems problematic in high-dimensional spaces. In the experiments performed, the algorithm terminates due to the estimated coverage with approximately 1.4 generated detectors. This “curios” phenomenon of such a low number of detectors and the low detection rate is explored in the following chapter.

7.7 Summary

For the low-dimensional data sets the real-valued positive and real-valued negative selection produced better classification results (high detection rate, low false alarm rate) than the statistical techniques. The Parzen-Window Estimator technique produced the overall poorest results for the low-dimensional data set. The one-class SVM produced slightly better classification results than the Parzen-Window Estimator, however the results vary a great deal with different kernels. The RBF kernel produced the overall best results compared with the other kernels. For the high-dimensional data set the statistical techniques and the real-valued positive selection produced high detection rates and low false alarm rates. However, the real-valued positive

[4] exponential operation and several arithmetic operations

selection and the Parzen-Window Estimator have limited applicability due to the high runtime complexity to classify an element. The one-class SVM overcomes this complexity problems by means of estimating the quantiles of the probability distribution, i.e. its support. The one-class SVM performs very well on the high-dimensional data set. The real-valued negative selection produced very poor results (extreme low detection rates). These poor results are investigated further in the following chapter.

Chapter 8

Limitation of Real-Valued Negative Selection in Higher Dimensions

In this chapter, we explore the poor classification results produced by the real-valued negative selection on the high-dimensional data set. Therefore properties of hyperspheres in high dimensions are studied. Unfortunately hyperspheres lose their familiar properties in high dimensional spaces and evolve from the pattern classification point of view adverse and undesirable properties.

8.1 Volume of Hyperspheres

The volume of a n-dimensional hypersphere with radius r can be calculated as follows :

$$V(n,r) = r^n \cdot \frac{\pi^{n/2}}{\Gamma\left(\frac{n}{2}+1\right)}$$

where

$\Gamma(n+1) = n! \quad \text{for} \quad n \in \mathbb{N} \quad \text{and}$
$\Gamma(n+\frac{1}{2}) = \dfrac{1 \cdot 3 \cdot 5 \cdot 7 \cdot \ldots \cdot (2n-1)}{2n}\sqrt{\pi}$ for half-integer arguments.

We briefly show the construction idea[1] behind the the volume calculation of hyperspheres. For an in-depth description see [43], where the complete construction and a proof is shown.

[1] taken from [43]

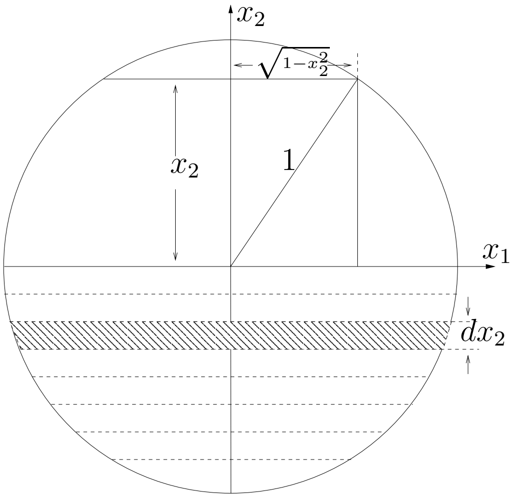


Figure 8.1: Unit circle dissected in infinite stripes (taken from [43])

The volume $V(n)$ of a n-dimensional *unit sphere* can be constructed inductively

$$\begin{aligned}
V(2) &= \pi \\
V(3) &= \frac{4}{3}\pi \\
&\vdots \\
V(n) &= \begin{cases} \dfrac{\pi^{n/2}}{(n/2)!} & , n \text{ even} \\[2ex] \dfrac{2^n \pi^{(n-1)/2}\,((n-1)/2)!}{n!} & , n \text{ odd} \end{cases}
\end{aligned}$$

Given a 2-dimensional unit circle

$$C^2 = \{(x_1, x_2) \in \mathbb{R}^2 \mid x_1^2 + x_2^2 \leq 1\}$$

The volume $V(C^2)$ can be calculated as a summation of infinite area stripes (see Fig. 8.1)

$$\begin{aligned}
V(C^2) &= 2 \cdot \int_{-1}^{1} \sqrt{1 - x_2^2}\, dx_2 \\
&= 2 \cdot \int_{0}^{\pi} \sqrt{1 - \cos^2(t)}\ \sin(t)\, dt
\end{aligned}$$

$$
\begin{aligned}
&= 2 \cdot \int_0^\pi \sin^2(t)\,dt \\
&= \int_0^\pi dt = \pi
\end{aligned}
$$

$V(C^2) \rightarrow V(C^3)$

$$
\begin{aligned}
V(C^3) &= \int_{-1}^{1} \pi \left(\sqrt{1-x_3^2}\right)^2 dx_3 \\
&= \pi \int_0^\pi (1-x_3^2)\,dx3 \\
&= \frac{4}{3}\pi
\end{aligned}
$$

$\vdots$

$V(C^{n-1}) \rightarrow V(C^n)$

$$
\begin{aligned}
V(C^n) &= V(C^{n-1}) \cdot \int_{-1}^{1} (1-x_n^2)^{(n-1)/2}\,dx_n \\
&= \frac{\pi^{n/2}}{\Gamma(\frac{n}{2}+1)}
\end{aligned}
$$

Proposition 8.1. *The volume of a n-dimensional hypersphere with radius r is*

$$
V(n,r) = r^n \cdot \frac{\pi^{n/2}}{\Gamma\left(\frac{n}{2}+1\right)} \tag{8.1}
$$

Proof. see [43] □

8.2 Curse of Dimensionality

The phenomenon “curse of dimensionality” was first mentioned by Bellman [6] during his study of optimizing a function of a few dozen variables in an exhaustive search space. For example, given a function defined on an unitary hypercube of dimension n, in each dimension 10 discrete points are considered for evaluating the function. In dimension $n = 2$, this results in 100 evaluations, whereas in dimension $n = 10$, 10^{10} function evaluations are required. In general, an exponential number of $(1/\epsilon)^n$ function evaluations are required to obtain an optimization error of ϵ and therefore is computationally infeasible, even for moderate n.

This simple example shows how problems like function optimization which are computational feasible in lower dimensions, transform to computationally infeasible problems in higher dimensions. A similar phenomenon (but not from the perspective of computational complexity) undergoes hyperspheres in high-dimensional spaces, where they lose their familiar properties. In high-dimensional shape-spaces $\mathbb{R}^n$, i.e. $n > 3$, hyperspheres have undesirable properties. These properties (the following corollaries) can be derived from proposition 8.1.

Corollary 8.1. *The volume of hyperspheres converges to* 0 *for* $n \to \infty$*.*

$$\lim_{n\to\infty} V(n,r) = 0$$

Proof.

$$\lim_{n\to\infty}\left(r^n \cdot \frac{\pi^{n/2}}{\underbrace{\Gamma\left(\frac{n}{2}+1\right)}_{\approx\sqrt{2\pi}e^{-n}\,n^{n+\frac{1}{2}}}} \right) = \frac{1}{\sqrt{2\pi}} \lim_{n\to\infty}\left(\frac{(\overbrace{r\,e\,\sqrt{\pi}}^{c})^n}{n^{n+\frac{1}{2}}} \right) = \frac{1}{\sqrt{2\pi}} \lim_{n\to\infty}\left(\frac{c^n}{n^{n+\frac{1}{2}}}\right) = 0$$

□

Corollary 8.2. *The fraction of the volume which lies at values between* $r-\epsilon$ *and* r*, where* $0 < \epsilon < r$ *is*

$$V_{fraction}(n,r,\epsilon) = 1 - \left(1 - \frac{\epsilon}{r}\right)^n$$

Proof.

$$1 - \frac{V(n,r-\epsilon)}{V(n,r)} = 1 - \left(\frac{\frac{(r-\epsilon)^n\cdot\pi^{n/2}}{\Gamma\left(\frac{n}{2}+1\right)}}{\frac{r^n\cdot\pi^{n/2}}{\Gamma\left(\frac{n}{2}+1\right)}} \right) = 1 - \left(1 - \frac{\epsilon}{r}\right)^n$$

□

Corollary 8.1 implies that the higher the dimension the smaller the volume of a hypersphere for fixed radii. This property is investigated in more detail, in the following section 8.3.

Corollary 8.2 reveals that in high-dimensional spaces, points which are uniformly randomly distributed inside the hypersphere, are almost concentrated in a thin shell close to the surface or, in other words, at very high dimensions the entire volume of a sphere is concentrated immediately below the surface.

Example 8.1. *Given a hypersphere with radius $r = 1$, $\epsilon = 0.1$ and $n = 50$ and k points which are uniformly randomly distributed inside the hypersphere. Approximately $1 - \left(1 - \frac{0.1}{1}\right)^{50} \approx 99,5\,\%$ of the k points lie within the thin ϵ-shell close to the surface.*

8.3 Volume Extrema

By keeping the radius fixed and differentiating the volume $V(n, r)$ with respect to n, one obtains the dimension[2] where the volume is maximal :

$$\frac{\partial}{\partial n}\left(\frac{r^n \cdot \pi^{n/2}}{\Gamma\left(\frac{n}{2}+1\right)}\right) = \frac{r^n \ln(r)\, \pi^{n/2}}{\Gamma\left(\frac{n}{2}+1\right)} + \frac{r^n \pi^{n/2} \ln(\pi)}{2\,\Gamma\left(\frac{n}{2}+1\right)} - \frac{r^n \pi^{n/2}\, \Psi\left(\frac{n}{2}+1\right)}{2\,\Gamma\left(\frac{n}{2}+1\right)} \tag{8.2}$$

$$\text{where} \quad \Psi(n) = \frac{\partial}{\partial n} \ln \Gamma(n)$$

Vice versa, keeping the dimension fixed and differentiate term (8.1) with respect to r, it is not solvable in roots, i.e. no extrema exists :

$$\frac{\partial}{\partial r}\left(\frac{r^n \cdot \pi^{n/2}}{\Gamma\left(\frac{n}{2}+1\right)}\right) = \frac{r^n\, n\, \pi^{n/2}}{r\,\Gamma\left(\frac{n}{2}+1\right)} \tag{8.3}$$

For instance, a hypersphere with radius $r = 1$ reaches its maximum volume in dimension 5 and loses volume in lower and higher dimensions. In figure 8.2 this fact is visualized for different radius lengths $r = \{0.9, 1.0, 1.1, 1.2\}$. One can see, that for each plotted radius the associated hypersphere reaches a maximal volume in a certain dimension and loses volume asymptotically in higher and lower dimensions.

[2] The dimension is obviously a nonnegative integer, however we consider term 8.2 analytically as a real-valued function

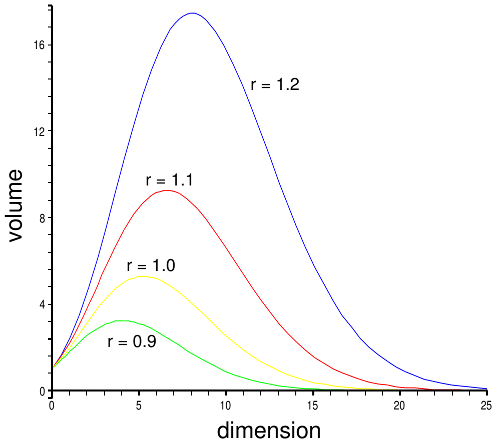


Figure 8.2: Hypersphere volume plot for radius lengths $r = \{0.9, 1.0, 1.1, 1.2\}$ and dimension $n = 0, \ldots, 25$. Obviously, n is a nonnegative integer, but the graph is drawn treating n as continuously varying.

In table 8.1 the dimension where a hypersphere reaches its maximum volume for different radius lengths is presented. For radius lengths $r = 0.05$ and $r = 0.1$, surprisingly the maximum volume lies in negative real-valued numbers. Hence, a volume maximization for such small radius lengths is not feasible, as the dimension is a nonnegative integer.

Table 8.1: Dimension where a hypersphere reaches the maximum volume for radius lengths $r = \{0.05, 0.1, 0.2, \ldots, 1.0\}$. Obviously, n is a nonnegative integer, however the results are obtained by a real-valued function and transformed to integer values with the floor function.

Radius r	0.05	0.1	0.2	0.3	0.4
Dimension $\lfloor n \rfloor$	-9.17 $\cdot 10^7$	-88.94	1.59	1.12	1.0

Radius r	0.5	0.6	0.7	0.8	0.9	1.0
Dimension $\lfloor n \rfloor$	1.03	1.20	1.53	2.14	3.23	5.27

8.4 Results and Observations

The results and observations presented in sections 8.1, 8.2 and 8.3 indicate that high-dimensional real-valued shape-spaces strongly bias the volume (recognition space) of hyperspheres. A hypersphere, for example with radius $r = 1$ has a high volume in relation to its radius length, up to dimension

15 (see Fig. 8.2). In higher dimensions ($n > 15$), for $r = 1$ the volume is nearly 0. This means that the recognition space — or in the context of real-valued negative selection the covered space — is nearly 0. In contrast, a radius that is too large ($r > 2$) in high dimensional spaces ($n > 10$) implies an exponential volume. This exponential volume problem, is the reason for the poor classification results shown in section 7.6. This is discussed in the next section.

8.5 Empty Space Phenomenon

Investigating the 41-dimensional KDD Cup data set, one can statistically verify, that the whole normalized non-anomalous class is concentrated at one place inside the unitary hypercube $\mathcal{U} = [0, 1]^{41}$. In [68], this characteristic is called "empty space phenomenon" and arises in any data set that does *not* grow exponentially with the dimension of the space. This phenomenon also occurs in our experiments performed with the real-valued negative selection. In table 7.15 it can be seen, that the real-valued negative selection algorithm terminated when (on average) 1.4 detectors were generated. By generating only one detector (hypersphere) with, for example, a radius $r = 3$ and a detector center not necessarily lying inside $\mathcal{U}$, the volume of that hypersphere amounts to $5.11 \cdot 10^{10}$. The unitary hypercube $\mathcal{U} = [0, 1]^{41}$ has a total volume of 1, however most of the volume of a hypercube is concentrated in the large corners, which themselves become very long "spikes". This can be verified by comparing the ratio (see term 8.4) of the distance $\sqrt{n}$ from the center of the hypercube to one of the edges to the perpendicular distance $a/2$ to one of the edges (see Fig. 8.3).

$$\frac{(\sum_{i=1}^{n}(a/2)^2)^{\frac{1}{2}}}{a/2} = \sqrt{n} \qquad \text{where } n \text{ is the dimension} \tag{8.4}$$

For $n \to \infty$, the term (8.4) goes to ∞ and therefore the volume is concentrated in very long "spikes" of $\mathcal{U}$.

As a consequence, the hypersphere covers only those (high-volume) spikes which are lying within the $V_{fraction}$ proportion of the hypersphere. Hence, the real-valued negative selection algorithm terminates with only a very small number of large radii detectors (hyperspheres) which are covering a limited number of spikes. As a result a large proportion of the volume of the hypercube is not lying within the hyperspheres — it lies in the remaining (high-volume) spikes, though the hypersphere volume is far higher than the hypercube volume.

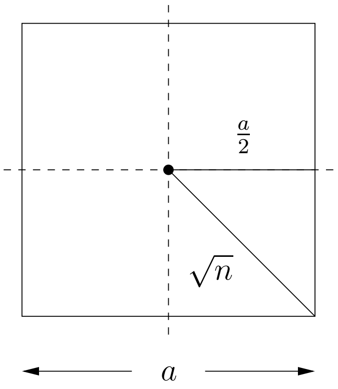


Figure 8.3: Distance ratio $\frac{\sqrt{n}}{a/2}$ between a line from center to a corner and a perpendicular line from center to an edge

The real-valued negative selection is a technique which is not well suitable for high-dimensional data sets, because it makes more sense to formulate a classification model with regard to the given training elements, instead of complementary space. The complementary (anomalous) space is exponentially large when compared to the "normal" space in high dimensions. The real-valued negative selection technique, attempts to cover this high-dimensional space with hyperspheres, but as we have shown, these have adverse properties in such high-dimensional spaces.

8.6 Summary

In this chapter, we have shown that hyperspheres have undesirable properties in high dimensions — the volume tends to zero by keeping the radius fixed, and nearly all uniformly randomly distributed points are close to the hypersphere surface. We have presented these hypersphere properties and have given an explanation for poor classification results reported in section 7.6. Moreover, we have now explained the limitations of the real-valued negative selection for high-dimensional classification problems. The hypersphere properties we have discussed are valid observations for all high-dimensional classification problems where hyperspheres are applied as recognition regions and as a result, these adverse hypersphere properties bias all (artificial immune system) algorithms, which use hyperspheres as recognition units.

Chapter 9

Conclusions

The immune system is a complex system which protects the body against intruders like viruses, fungi, parasites and bacteria. By reading this sentence is seems to be obvious to develop an intrusion detection system which is inspired by the outstanding adaptive detection principles of an immune system. However, as mentioned by Vapnik [67], this is not necessarily the best way for creating an artificial learning machine (or in our context an intrusion detection system).

In this thesis, we have investigated the negative selection, as an immune inspired paradigm when applied for anomaly detection and (network) intrusion detection problems. Roughly speaking, in the natural immune system the negative selection is a process which eliminates self-reactive lymphocytes and ensures that *only* those lymphocytes enter the blood stream which do not recognize self proteins. These lymphocytes recognize almost all non-self proteins which do not belong to the body. This principle was first abstracted and algorithmically formulated by Forrest et al. [27] for detecting data manipulations caused by computer viruses. The basic idea was to generate a number of detectors (lymphocytes) in the complementary space[1] and then to apply these detectors to classify new (unseen) data as self (no data manipulation) or non-self (data manipulation). In following works, different algorithms for generating detectors were proposed [15, 71, 2, 60] and the scope of the negative selection was extended to general anomaly detection [12, 30, 31, 39, 40, 29] and (network) intrusion detection problems [1, 35, 30, 5, 58, 72]. However, in several works [41, 18, 28] different problems in negative selection were mentioned, but not closer investigated from the perspective of a pattern classification problem. This investigation was done in this thesis.

We have described the immunological principles of negative selection and

[1] space which contains no seen self elements

positive selection. Furthermore, it is described how the immune system is able to recognize a nearly unlimited number of antigens with a limited number of antibodies. For developing immune-inspired algorithms, a proper representation of the immune elements and an abstraction of the immune principles must be formulated. Work in theoretical immunology has proposed different immune elements representations and affinity metrics and provided much of the foundations for the development of artificial immune systems. In artificial immune systems, a framework is commonly used, as it guides in a 3-step formalization to a solution for a given problem.

We have described this 3-step formalization and have presented two shape-spaces (Hamming and real-valued) and different affinity metrics for the negative selection. In our first exploration the Hamming negative selection with the associated r-chunk matching rule was investigated with respect to the number of generable detectors and the number of resulting holes (undetectable elements). These undetectable elements are required to generalize beyond the training set. In our second exploration, the generalization regions in Hamming negative selection were empirically investigated and the complexity of several detector generating algorithms were reported. Results and observations from this explorations were summarized and discussed in the context of the appropriateness as a network intrusion detection technique. To summarize, statements and hypothesis reported in [41, 28] are supported and verified. The Hamming negative selection seems to be a very appealing approach for detecting manipulations in data, however the complexity to generate and to store a sufficient number of detectors is infeasibly high when applied to real-world (network) intrusion detection problems. This complexity problem can be reduced by making the detectors more generic, however this induces an infeasible number of holes. Generic detectors induce holes in regions, where no self elements are concentrated, and this consequently results in a misclassification of unseen elements.

Another explored variant of the negative selection, is the real-valued negative selection. The real-valued negative selection is applied for (real-valued) anomaly detection problems. Detectors (lymphocytes) are represented as hyperspheres with a center and a radius. In the detector generation phase, the complementary space is filled with hyperspheres. A new (unseen) element is classified as an anomaly, when it lies within the hyperspheres, otherwise it is a non-anomalous element. For comparative studies, a new non-complementary approach called real-valued positive selection was proposed. In the real-valued positive selection each seen self element is a detection hypersphere (called Self-Detector). A new (unseen) element is classified as a non-anomalous element, when it lies within the (self-)hyperspheres, otherwise it is an anomaly. For comparative studies well established statistical anomaly detection techniques — Parzen-Window and one class SVM — were presented. The classification performance was evaluated by means of ROC analysis for a low-dimensional anomaly detection problem and for a high-dimensional anomaly detection problem. The comparative studies revealed, that the real-valued positive selection produced high detection rates and low false alarm rates for the low- and high-dimensional data sets. However, it was limited applicable for a large number of self elements, as in the worst case the distance to each (self-)hypersphere must be calculated. The Parzen-Window technique produced the poorest (low detection rates and high false alarm rates) classification results for the low-dimensional data sets, but performed well on the high-dimensional data set. However, the Parzen-Window technique is likewise, computationally expensive, because each self element has to calculate the class conditionally probability for an unseen test element. The one-class SVM produced slightly worse results for the low-dimensional data sets when compared with the real-valued positive selection. However, these results vary a great deal for all tested kernels. The RBF kernel produced the overall best results compared with the other tested kernels. For the high-dimensional data set, the one-class SVM performs very well. The runtime complexity for classifying an unseen test element is low compared with the Parzen-Window and real-valued positive selection, as it considers only a subset of the self elements (the support vectors). The detection and false alarm rate is similar to the Parzen-Window and the real-valued positive selection. The real-valued negative selection produced slighty worse classification results than the real-valued positive selection for the low-dimensional data sets. *However*, on the high-dimensional data set the real-valued negative selection performed extremely poor compared with the other benchmarked anomaly detection techniques. This poor classification results on the high-dimensional data set were explained and discussed in the last part of this thesis. To summarize, hyperspheres have undesirable properties in high dimensions — the volume tends to zero by keeping the radius fixed, and the entire volume of a hypersphere is concentrated immediately below the surface. This properties induce general fundamental limitations of the real-valued negative selection for high-dimensional anomaly detection problems. We finish this thesis with the following insights : the negative selection is a very intuitive and attractive immune inspired principle, but it is not appropriate and not applicable for real-world anomaly detection and (network) intrusion detection problems.

9.1 Future Work

From the point of view of the author, the negative selection was thoroughly explored and has no potential for becoming a robust and useful intrusion detection and anomaly detection technique. We therefore believe that future work in this direction is not meaningful. Of course, for some toy problems the negative selection reveals an average classification performance, however the negative selection is surely not a computational efficient and robust technique. A computational efficient technique must be able to handle real-world problems. A robust technique is one, which is applicable on a manifold problem domain and produces good, (statistical) stable, competitive and sound results.

We believe that intrusion detection and anomaly detection problems could be better solved with statistical approaches for instance with the one-class SVM [55] or the minium enclosing hypersphere [65] approach, rather than with negative selection. These statistical techniques are computational efficient [56], robust [56] and additionally are anchored in a statistical learning framework [11, 9]. This implies that statistical techniques not only allow good and competitive classification results, but also sound theoretical results. Citing again Vladimir Vapnik : "*Nothing is more practical than a good theory.*", we believe that computational efficient and robust techniques for intrusion detection and anomaly detection problems are rich available in the field of statistical learning and therefore future work should point in this direction.

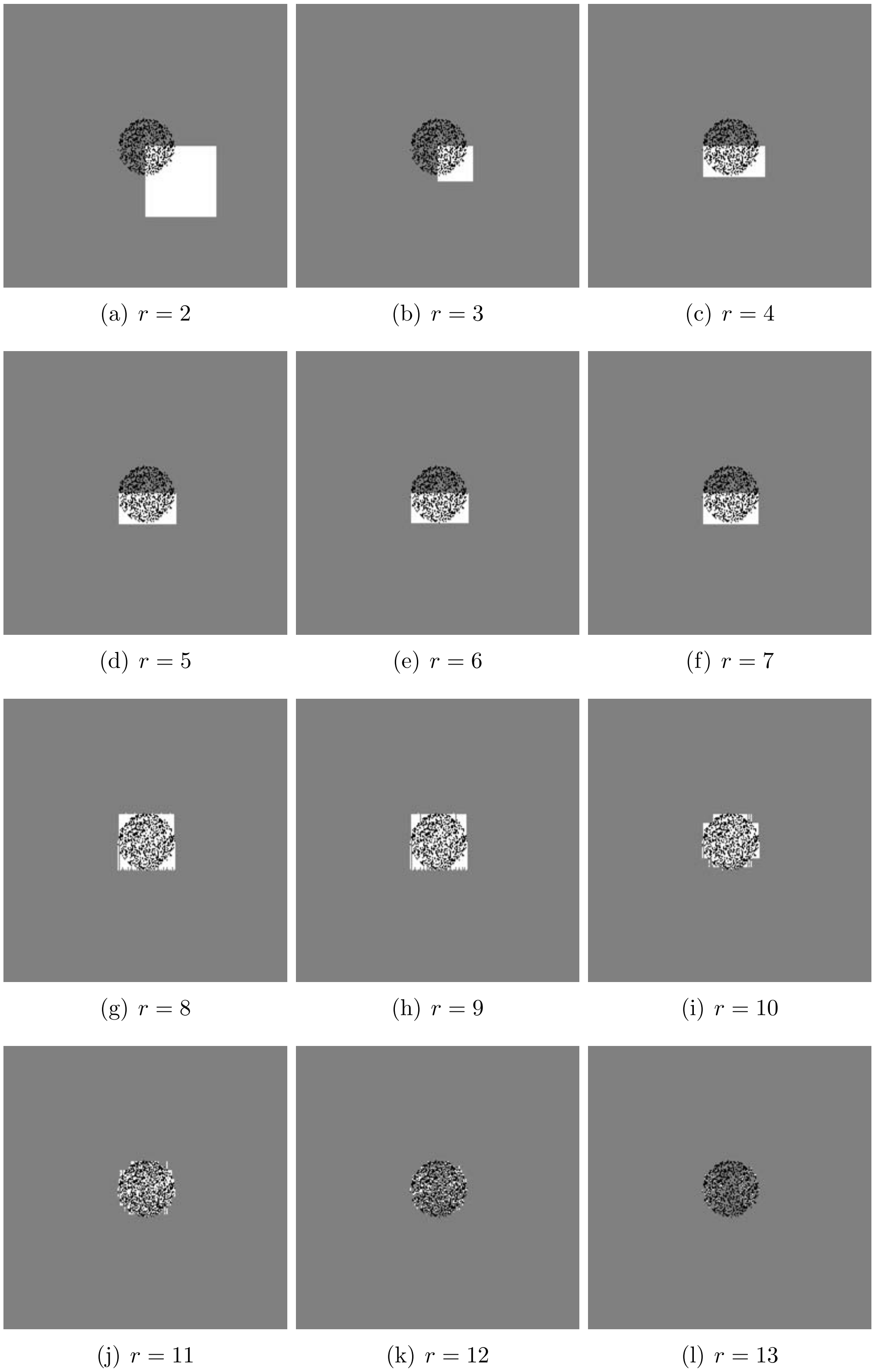
9.2 Epilogue

The author would like to emphasize here as a final statement, that this negative results obtained, of course are *only* related to the investigated negative selection. In the field of artificial immune systems exists a lot of immune-inspired algorithms and techniques which have a great potential in their application areas. Artificial immune systems are a young and exciting research field and the author is convinced that immune-inspired algorithms and techniques will become well established problem solving methods in the near future, as for instance genetic algorithms and genetic programming approaches.

Appendix A

Appendix

A.1 Figures of Generalization Regions Experiment



(a) $r = 2$

(b) $r = 3$

(c) $r = 4$

(d) $r = 5$

(e) $r = 6$

(f) $r = 7$

(g) $r = 8$

(h) $r = 9$

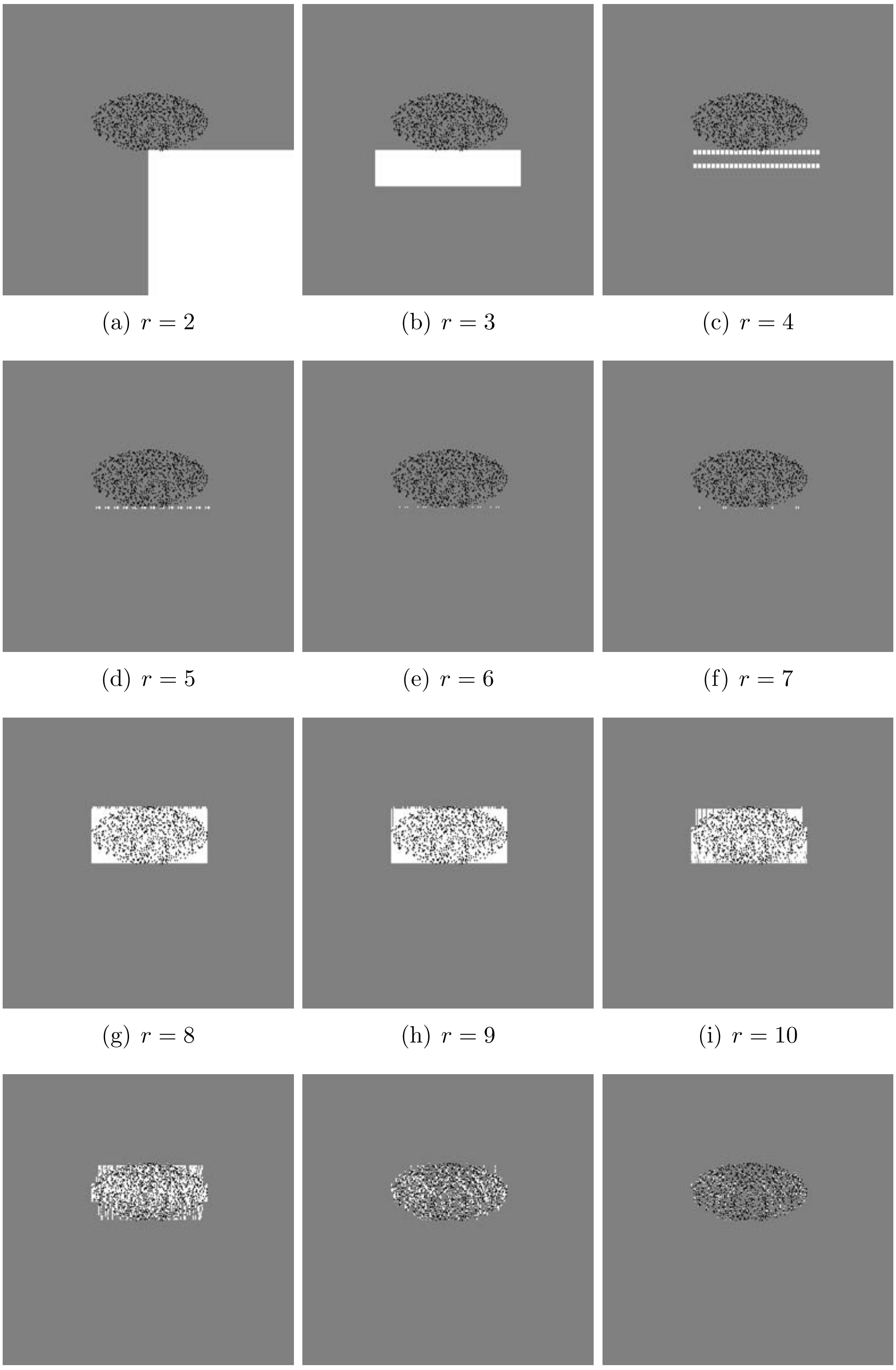
(i) $r = 10$

(j) $r = 11$

(k) $r = 12$

(l) $r = 13$

Figure A.1: 1000 random (self) points distributed inside a sphere with center $(0.5, 0.5)$ and radius 0.1. The grey shaded area is covered by the generated r-chunk detectors, the white area are holes. The black points are self elements.



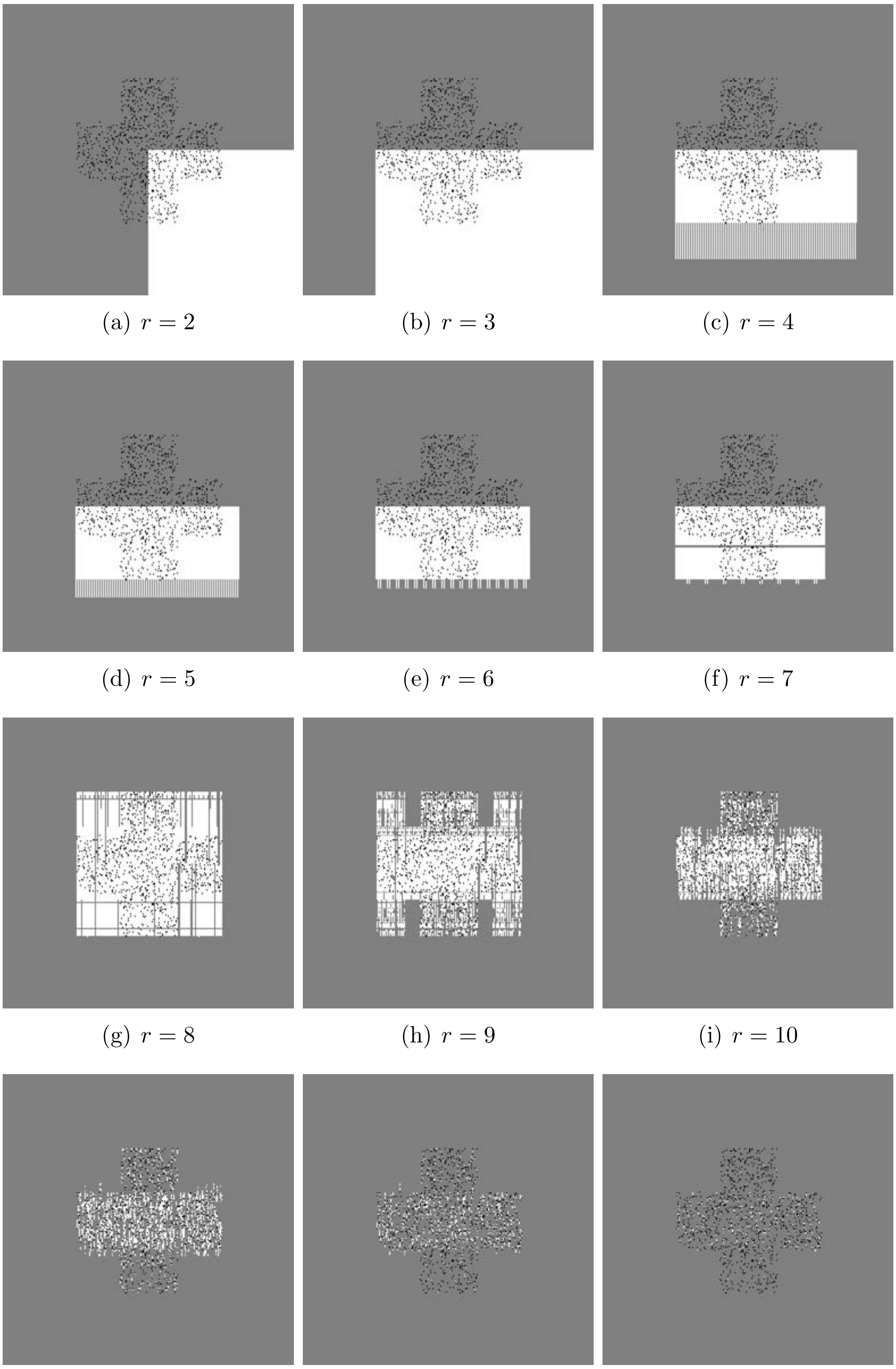
(a) $r = 2$ (b) $r = 3$ (c) $r = 4$

(d) $r = 5$ (e) $r = 6$ (f) $r = 7$

(g) $r = 8$ (h) $r = 9$ (i) $r = 10$

(j) $r = 11$ (k) $r = 12$ (l) $r = 13$

Figure A.2: 1000 random (self) points distributed inside an ellipse with center $(0.5, 0.5)$, height 0.4 and width 0.2. The grey shaded area is covered by the generated r-chunk detectors, the white area are holes. The black points are self elements.



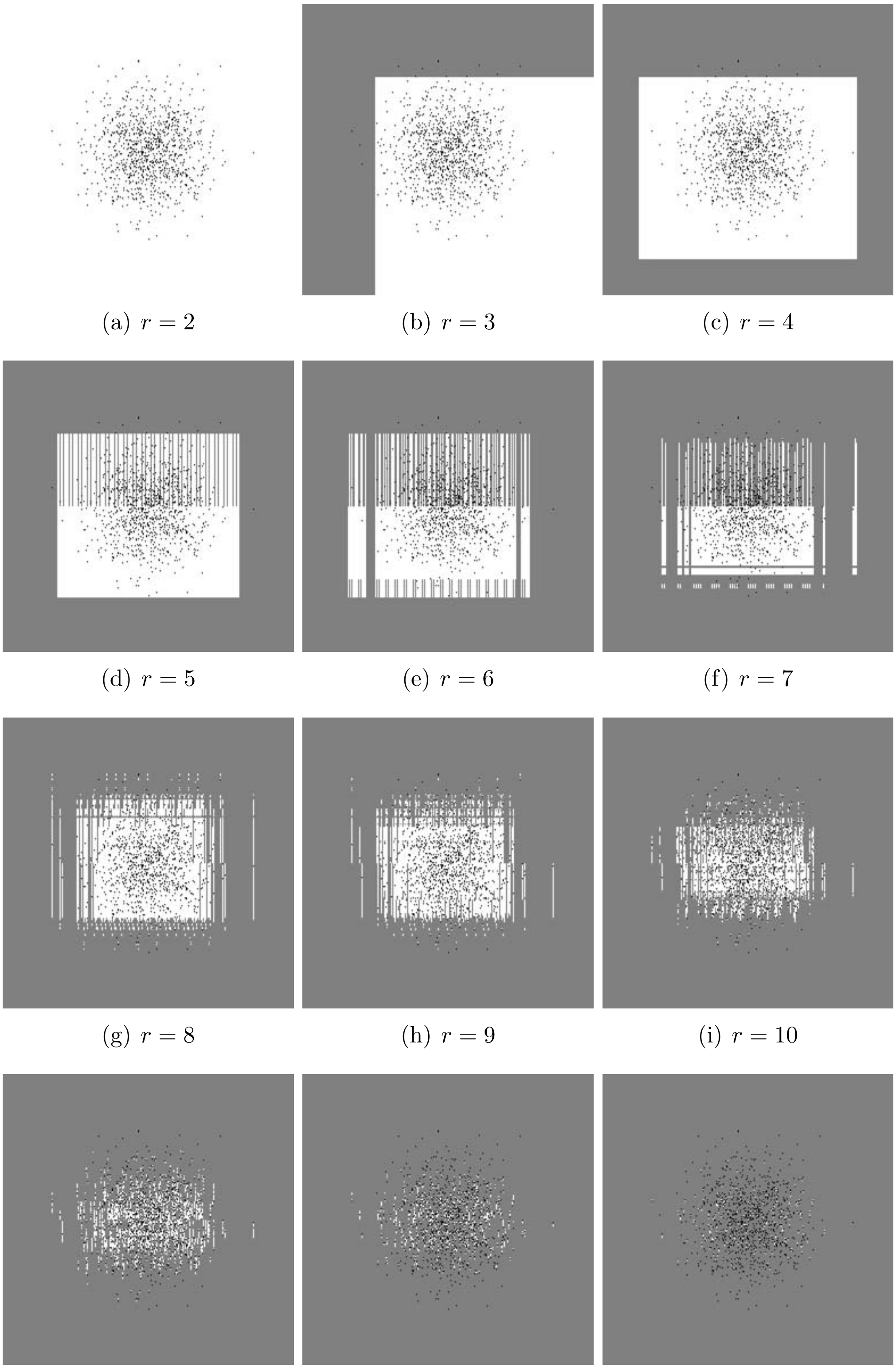
(a) $r = 2$ (b) $r = 3$ (c) $r = 4$

(d) $r = 5$ (e) $r = 6$ (f) $r = 7$

(g) $r = 8$ (h) $r = 9$ (i) $r = 10$

(j) $r = 11$ (k) $r = 12$ (l) $r = 13$

Figure A.3: 1000 random (self) points distributed inside two rectangles with x, y coordinates $(0.4, 0.25)$, height 0.2, width 0.5 and coordinates $(0.25, 0.4)$, height 0.5, width 0.2. The grey shaded area is covered by the generated r-chunk detectors, the white area are holes. The black points are self elements.



(a) $r = 2$

(b) $r = 3$

(c) $r = 4$

(d) $r = 5$

(e) $r = 6$

(f) $r = 7$

(g) $r = 8$

(h) $r = 9$

(i) $r = 10$

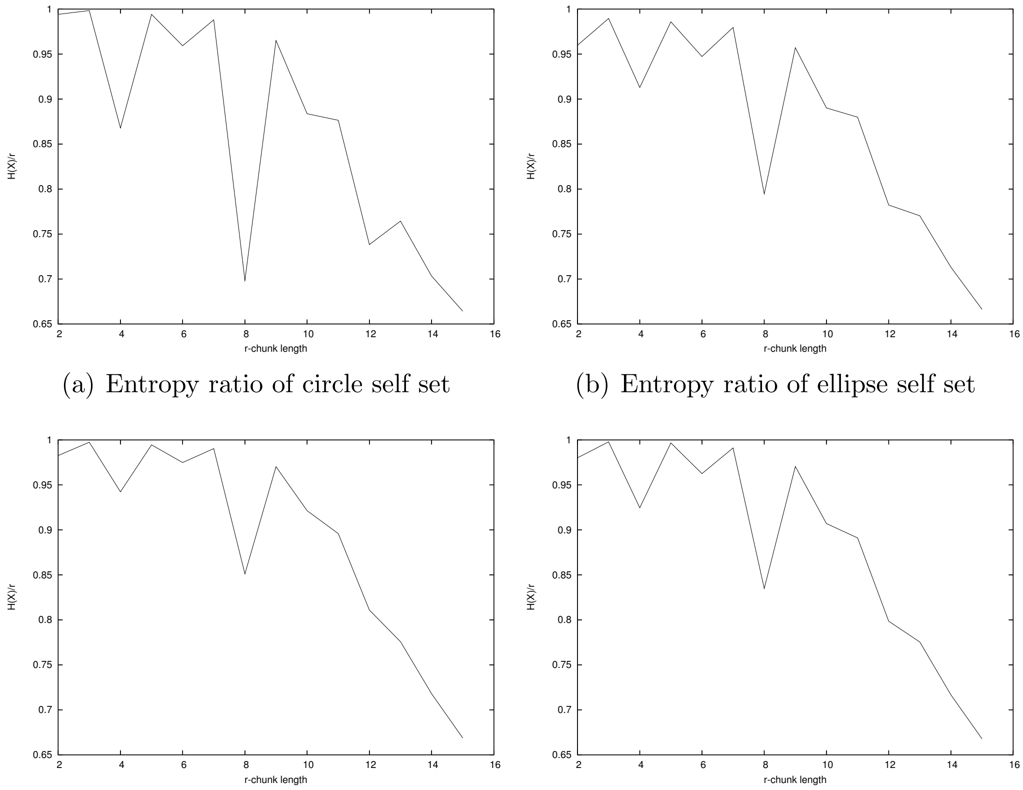
(j) $r = 11$

(k) $r = 12$

(l) $r = 13$

Figure A.4: 1000 random (self) points generated by a Gaussian distribution with mean $\mu = 0.5$ and variance $\sigma = 0.1$. The grey shaded area is covered by the generated r-chunk detectors, the white area are holes. The black points are self elements.

A.2 Figures of Entropy Experiment



(a) Entropy ratio of circle self set

(b) Entropy ratio of ellipse self set

(c) Entropy ratio of rectangle self set

(d) Entropy ratio of Gaussian self set

Figure A.5: Coherence between entropy ratio $H(X)/r$ of self set S and r-chunk lengths $r = \{2, 3, \dots, 15\}$

A.3 Monte Carlo Integration

Monte Carlo Integration is a method to integrate a function over a complicated domain, where analytical expressions are very difficult to be applied – e.g. the calculation of the volume of overlapping hyperspheres in higher dimensions. Given integrals of the form $I = \int_{\mathcal{X}} h(\mathbf{x}) f(\mathbf{x}) d\mathbf{x}$, where $h(\mathbf{x})$ and $f(\mathbf{x})$ are functions for which $h(\mathbf{x})f(\mathbf{x})$ is integrable over the space $\mathcal{X}$, and $f(\mathbf{x})$ is a non-negative valued, integrable function satisfying $\int_{\mathcal{X}} f(\mathbf{x}) d\mathbf{x} = 1$. The Monte Carlo integration picks N random points $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$, over $\mathcal{X}$

and approximates the integral as

$$I \approx \frac{1}{N} \sum_{n=1}^{N} h(\mathbf{x}_n) \tag{A.1}$$

The absolute error of this method is *independent* of the dimension of the space $\mathcal{X}$ and decreases as $1/\sqrt{N}$ [26]. By applying this integration method, two fundamental questions arises :

- How many observations should one collect to ensure a specified statistical accuracy ?
- Given N observations from a Monte Carlo Experiment, how accurate is the estimated solution ?

Both question are answered and discussed in [26]. Using the Chebyshev's inequality and specifying a *confidence level* $1 - \delta$, one can determine the smallest sample size N that guarantees an integration error no larger than ϵ. In [26] this specification is called the (ϵ, δ) *absolute error criterion* and leads to the worst-case sample size

$$N := \lceil 1/4\delta\epsilon^2 \rceil \tag{A.2}$$

Term A.2 shows that there is always a tradeoff between the accuracy of the solution and the sample size. In the following section this tradeoff is demonstrated on a Monte Carlo algorithm.

A.4 Monte Carlo Hyperspheres Volume Integration

Using equations $(A.1)$ and $(A.2)$ a straightforward algorithm can be developed which estimates the total space (volume) covered by the hyperspheres inside the unitary hypercube $[0, 1]^n$. It can be seen, that the accuracy of the integrated space depends on the absolute error of the estimated volume and the confidence level. A higher confidence level δ or a smaller absolute error ϵ bias the required sample size and therefore the algorithm runtime complexity.

Algorithm 6: Monte Carlo Hyperspheres Volume Integration

```
   input : $H$ = set of hyperspheres, $\epsilon$ = absolute error of the estimated
           volume, $\delta$ = confidence level
   output: total volume of $H$
1  begin
2      inside $\longleftarrow$ 0
       // calculate required worst-case
       // sample size $N$
3      $N \longleftarrow \lceil 1/4\delta\epsilon^2 \rceil$
4      for $i \leftarrow 1$ to $N$ do
5          $\mathbf{x} \longleftarrow$ random point from $[0,1]^n$
6          foreach $h \in H$ do
7              if $dist(\mathbf{c}_h, \mathbf{x}) \leq r_h$ then
8                  inside $\longleftarrow$ inside + 1
9                  goto 5:
10     return (inside/$N$)
11 end
```

Bibliography

[1] Uwe Aickelin, Julie Greensmith, and Jamie Twycross. Immune system approaches to intrusion detection – a review. In *Proceedings of the 3nd International Conference on Artificial Immune Systems (ICARIS)*, volume 3239 of *Lecture Notes in Computer Science*, pages 316–329. Springer-Verlag, 2004.

[2] Modupe Ayara, Jonathan Timmis, Rogerio de Lemos, Leandro N. de Castro, and Ross Duncan. Negative selection: How to generate detectors. In *Proceedings of the 1nd International Conference on Artificial Immune Systems (ICARIS)*, pages 89–98. University of Kent at Canterbury Printing Unit, 2002.

[3] Rebecca Bace and Peter Mell. *Intrusion Detection Systems.* National Institute of Standards and Technology (NIST), 2001. http://csrc.nist.gov/publications/nistpubs/800-31/sp800-31.pdf.

[4] Justin Balthrop, Fernando Esponda, Stephanie Forrest, and Matthew Glickman. Coverage and generalization in an artificial immune system. In *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 3–10, New York, 9-13 July 2002. Morgan Kaufmann Publishers.

[5] Justin Balthrop, Stephanie Forrest, and Matthew Glickman. Revisiting lisys: Parameters and normal behavior. In *Congress On Evolutionary Computation – CEC 2002*, pages 1045–1050. IEEE Press, 2002.

[6] Richard Bellman. *Adaptive Control Processes: A Guided Tour.* Princeton University Press, 1961.

[7] Christopher M. Bishop. Novelty detection and neural network validation. In *IEE Proceedings: Vision, Image and Signal Processing*, volume 141, pages 217–222, 1994.

[8] Christopher M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[9] Olivier Bousquet and Stéphane Boucheron Gábor Lugosi. *Introduction to Statistical Learning Theory*, volume 3176 of *Lecture Notes in Artificial Intelligence*, pages 169–207. Springer-Verlag, 2004.

[10] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a Library for Support Vector Machines (http://www.csie.ntu.edu.tw/∼cjlin/papers/libsvm.pdf)*, December 2004.

[11] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods.* Cambridge University Press, 2000.

[12] Dipankar Dasgupta and Stephanie Forrest. Novelty detection in time series data using ideas from immunology. In *Proceedings of the 5th International Conference on Intelligent Systems*, 1996.

[13] Leandro N. de Castro and Jonathan Timmis. *Artificial Immune Systems: A New Computational Intelligence Approach.* Springer Verlag, 2002.

[14] Patrick D'haeseleer. An immunological approach to change detection: theoretical results. In *Proceedings of the 9th IEEE Computer Security Foundations Workshop*, pages 18–26. IEEE Computer Society, IEEE Computer Society Press, 1996.

[15] Patrick D'haeseleer, Stephanie Forrest, and Paul Helman. An immunological approach to change detection: algorithms, analysis, and implications. In *Proceedings of the 1996 IEEE Symposium on Research in Security and Privacy*, pages 110–119. IEEE Computer Society, IEEE Computer Society Press, May 1996.

[16] David Dittrich. The stacheldraht distributed denial of service attack tool, 1999. http://staff.washington.edu/dittrich.

[17] Richard 0. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification.* Wiley-Interscience, second edition, 2001.

[18] Marc Ebner, Hans-Georg Breunig, and Jürgen Albert. On the use of negative selection in an artificial immune system. In *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 957–964. Morgan Kaufmann Publishers, 2002.

[19] Claudia Eckert. *IT-Sicherheit, Konzepte-Verfahren-Protokolle.* Oldenburg Verlag, 3. edition, 2004.

[20] Victor H. Engelhard. How cells process antigens. *Scientific American*, pages 54–61, August 1994.

[21] Fernando Esponda, Elena S. Ackley, Stephanie Forrest, and Paul Helman. On-line negative databases. In *Proceedings of the 3nd International Conference on Artificial Immune Systems (ICARIS)*, volume 3239 of *Lecture Notes in Computer Science*, pages 175–188. Springer-Verlag, 2004.

[22] Fernando Esponda, Stephanie Forrest, and Paul Helman. The crossover closure and partial match detection. In *Proceedings of the 2nd International Conference on Artificial Immune Systems (ICARIS)*, volume 2787 of *Lecture Notes in Computer Science*, pages 249–260. Springer-Verlag, 2003.

[23] Fernando Esponda, Stephanie Forrest, and Paul Helman. A formal framework for positive and negative detection schemes. *IEEE Transactions on Systems, Man and Cybernetics Part B: Cybernetics*, 34(1):357–373, 2004.

[24] Tom Fawcett. ROC graphs: Notes and practical considerations for data mining researchers. Technical Report HPL-2003-4, Hewlett Packard Laboratories, January 17 2003.

[25] William Feller. *An Introduction to Probability Theory and its Applications*, volume 1. John Wiley & Sons, 3. edition, 1968.

[26] George S. Fishman. *Monte Carlo Concepts, Algorithms, and Applications.* Springer, 1995.

[27] Stephanie Forrest, Alan S. Perelson, L. Allen, and R. Cherukuri. Self-nonself discrimination in a computer. In *Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy.* IEEE Computer Society Press, 1994.

[28] Alex Alves Freitas and Jon Timmis. Revisiting the foundations of artificial immune systems: A problem-oriented perspective. In *Proceedings of the 2nd International Conference on Artificial Immune Systems (ICARIS)*, volume 2787 of *Lecture Notes in Computer Science*, pages 229–241. Springer-Verlag, 2003.

[29] Fabio González, Dipankar Dasgupta, and Jonatan Gómez. The effect of binary matching rules in negative selection. In *Genetic and Evolutionary Computation – GECCO-2003*, volume 2723 of *Lecture Notes in Computer Science*, pages 195–206, Chicago, 12-16 July 2003. Springer-Verlag.

[30] Fabio González, Dipankar Dasgupta, and Robert Kozma. Combining negative selection and classification techniques for anomaly detection. In *Congress on Evolutionary Computation*, pages 705–710. IEEE, May 2002.

[31] Fabio González, Dipankar Dasgupta, and Luis Fernando Nio. A randomized real-valued negative selection algorithm. In *Proceedings of the 2nd International Conference on Artificial Immune Systems (ICARIS)*, volume 2787 of *Lecture Notes in Computer Science*, pages 261–272, Edinburgh, UK, 2003. Springer-Verlag.

[32] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning, Data Mining, Inference and Prediction.* Springer, 2001.

[33] R. Heady, G. Luger, A. Maccabe, and M. Servilla. The architecture of a network level intrusion system. Technical report, Computer Science Department, University of New Mexico, August 1990.

[34] Hettich, S. and Bay, S. D. KDD Cup 1999 Data, 1999. http://kdd.ics.uci.edu.

[35] Steven Hofmeyr and Stephanie Forrest. Architecture for an artificial immune system. *Evolutionary Computation*, 8(4):443–473, 2000.

[36] Steven Andrew Hofmeyr. *An Immunological Model of Distributed Detection and its Application to Computer Security.* PhD thesis, University of New Mexico, 1999.

[37] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. *A Pratical Guide to Support Vector Classification (http://www.csie.ntu.edu.tw/∼cjlin/papers/guide/guide.pdf)*, July 2003.

[38] Charles A. Janeway, Paul Travers, Mark Walport, and Mark Shlomchik. *Immunologie.* Spektrum Akademischer Verlag, 5. edition, 2002.

[39] Zhou Ji and Dipankar Dasgupta. Augmented negative selection algorithm with variable-coverage detectors. In *Congress on Evolutionary Computation*, pages 1081–1088. IEEE, 2004.

[40] Zhou Ji and Dipankar Dasgupta. Real-valued negative selection algorithm with variable-sized detectors. In *Genetic and Evolutionary Computation – GECCO-2004, Part I*, volume 3102 of *Lecture Notes in Computer Science*, pages 287–298, Seattle, WA, USA, 26-30 June 2004. Springer-Verlag.

[41] Jungwon Kim and Peter J. Bentley. An evaluating of negative selection in an artificial immune system for network intrusion detection. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001*, pages 1330–1337, 2001.

[42] Jack Koziol. *Intrusion Detection with Snort.* Sams, 2003.

[43] Max Leppmeier. *Kugelpackungen von Kepler bis heute.* Vieweg Verlag, 1997.

[44] David J. C. MacKay. *Information Theory, Inference, and Learning Algorithms.* Cambridge University Press, 2003.

[45] Stephen Marsland. Novelty detection in learning systems. *Neural Computing Surveys*, 3, 2003.

[46] Michael Mitzenmacher and Eli Upfal. *Probability and Computing, Randomized Algorithms and Probabilistic Analysis.* Cambridge University Press, 2005.

[47] Klaus Mosegaard and Malcolm Sambridge. Monte Carlo analysis of inverse problems. *Inverse Problems*, 18:29–54, 2002.

[48] Biswanath Mukherjee, L. Todd Heberlein, and Karl N. Levitt. Intrusion detection system. *IEEE Network*, May/June 1994.

[49] Jerome K. Percus, Ora E. Percus, and Alan S. Perelson. Predicting the size of the T-cell receptor and antibody combining region from consideration of efficient self-nonself discrimination. *Proceedings of National Academy of Sciences USA*, 90:1691–1695, 1993.

[50] A. S. Perelson and G.F. Oster. Theoretical studies of clonal selection: minimal antibody repertoire size and reliability of self-nonself discrimination. In *J. Theor. Biol.*, volume 81, pages 645–670, 1979.

[51] A. S. Perelson and G. Weisbuch. Immunology for physicists. *Reviews of Modern Physics*, 69(4):1219–1267, 1997.

[52] Martin Torsen Ranang. An artificial immune system approach to preserving security in computer networks. Master's thesis, Norges Teknisk-Naturvitenskapelige Universitet, 2002.

[53] Stephen Roberts. Extreme value statistics for novelty detection in biomedical signal processing. *IEE Proceedings Science, Technology & Measurement*, 147(6):363–367, 2000.

[54] Glenn W. Rowe. *Theoretical Models in Biology: The Origin of Life, the Immune System, and the Brain.* Oxford Science Publications, 1997.

[55] Bernhard Schölkopf, John C. Platt, Shawe-Taylor, Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. Estimating the support of a high-dimensional distribution. Technical Report MSR-TR-99-87, Microsoft Research (MSR), November 1999.

[56] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis.* Cambridge University Press, 2004.

[57] B. W. Silverman. *Density Estimation for Statistics and Data Analysis.* Chapman and Hall, 1986.

[58] Shantanu Singh. Anomaly detection using negative selection based on the r-contiguous matching rule. In *Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS)*, pages 99–106. Unversity of Kent at Canterbury Printing Unit, 2002.

[59] Timothy K. Starr, Stephen C. Jameson, and Kristin A. Hogquist. Positive and negative selection of T cells. *Annual Review of Immunology*, 21:139–176, April 2003.

[60] Thomas Stibor, Kpatcha M. Bayarou, and Claudia Eckert. An investigation of R-chunk detector generation on higher alphabets. In *Proceedings of Genetic and Evolutionary Computation Conference – GECCO-2004*, volume 3102 of *Lecture Notes in Computer Science*, pages 299–307. Springer-Verlag, 2004.

[61] Thomas Stibor, Philipp H. Mohr, Jonathan Timmis, and Claudia Eckert. Is negative selection appropriate for anomaly detection ? In *Proceedings of Genetic and Evolutionary Computation Conference – GECCO-2005*, pages 321–328. ACM Press, 2005.

[62] Thomas Stibor, Jonathan Timmis, and Claudia Eckert. A comparative study of real-valued negative selection to statistical anomaly detection techniques. In *Proceedings of 4th International Conference on Artificial Immune Systems*, volume 3627 of *Lecture Notes in Computer Science*, pages 262–275. Springer-Verlag, 2005.

[63] Thomas Stibor, Jonathan Timmis, and Claudia Eckert. On the appropriateness of negative selection defined over hamming shape-space as a network intrusion detection system. In *Congress On Evolutionary Computation – CEC 2005*, pages 995–1002. IEEE Press, 2005.

[64] L. Tarassenko, P. Hayton, N. Cerneaz, and M. Brady. Novelty detection for the identification of masses in mammograms. In *Proceedings of the 4th IEE International Conference on Artificial Neural Networks*, pages 442–447, 1995.

[65] David M. J. Tax. *One-class classification.* PhD thesis, Technische Universiteit Delft, 2001.

[66] David M. J. Tax and Robert P. W. Duin. Data domain description using support vectors. In *European Symposium on Artificial Neural Networks – ESANN*, pages 251–256, 1999.

[67] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, second edition, 1999.

[68] Michel Verleysen. Learning high-dimensional data. *Limitations and Future Trends in Neural Computation*, 186:141–162, 2003.

[69] Pantelis Vlachos. StatLib. http://lib.stat.cmu.edu.

[70] Slawomir T. Wierzchoń. Discriminative power of the receptors activated by k-contiguous bits rule. *Journal of Computer Science and Technology*, 1(3):1–13, 2000.

[71] Slawomir T. Wierzchoń. Generating optimal repertoire of antibody strings in an artificial immune system. In *Intelligent Information Systems*, pages 119–133. Springer Verlag, 2000.

[72] Zejun Wu and Yiwen Liang. Self-regulating method for model library based artificial immune systems. In *Proceedings of 4th International Conference on Artificial Immune Systems*, volume 3627 of *Lecture Notes in Computer Science*, pages 353–365. Springer-Verlag, 2005.

[73] Dit-Yan Yeung and Calvin Chow. Parzen-window network intrusion detectors. In *Proceedings of the Sixteenth International Conference on Pattern Recognition*, pages 385–388, 2002.

Curriculum Vitae

Name:	Thomas STIBOR
Date and place of birth:	07th April, 1976 in Ostrov, Czech Republic
Marital status:	single, no children
Nationality:	German
Place of residence:	Havelstrasse 22 D - 64295 Darmstadt Mobil: +49 163 5290202 E-mail:stibor@sec.informatik.tu-darmstadt.de WWW: http://www.sec.informatik.tu-darmstadt.de/ en/mitarbeiter/stibor/index.html

Education:

Mar/06	**Degree :** Dr. rer. nat. (PhD), Darmstadt University of Technology. PhD thesis title : “On the Appropriateness of Negative Selection for Anomaly Detection and Network Intrusion Detection”, Supervisor : Prof. Dr. Claudia Eckert, Examiner : Prof. Dr. Werner Dilger, Dr. Jonathan Timmis
Oct/04-Mar/05	Visiting Researcher, University of Canterbury at Kent, United Kingdom (Research Group of Dr. Jonathan Timmis)
Sept/02	PhD Student, Research Group IT-Security of Prof. Dr. Claudia Eckert, Departement of Computer Science, Darmstadt University of Technology, Germany
Mar/02	Research Assistant, Fraunhofer Institute for Secure Information Technology, Security and Smartcard Technology, Darmstadt, Germany
Feb/02	**Degree :** Diplom (M.Sc.) in Computer Science, University of Frankfurt/Main, Germany. Master thesis title : “Zeichen von Kurven” (in German), [english title : „Drawing Curves“] Supervisor : Prof. Dr. Torben Hagerup
Oct/99	**Degree :** Vordiplom (B.A) in Computer Science, University of Frankfurt/Main, Germany
Oct/97	Study of Computer Science and Meteorology (minor) at University of Frankfurt/Main, Germany
Jul/96	Military service : tank artillery unit Idar-Oberstein, Germany
Jul/87-Jun/96	**„A“ Levels,** Frankfurt Grammar School (Germany)

Research interests:

Artificial Immune Systems (Immune inspired principles for anomaly detection problems)
Machine Learning (Supervised/Unsupervised learning problems, SVM learning)
IT-Security (Security models and intrusion detection)