# Is Negative Selection Appropriate for Anomaly Detection ?

Thomas Stibor
Department of Computer Science
Darmstadt University of Technology
stibor@sec.informatik.tu-darmstadt.de

Philipp Mohr
Computing Laboratory
University of Kent
phm4@kent.ac.uk

Jonathan Timmis
Computing Laboratory
University of Kent
J.Timmis@kent.ac.uk

## ABSTRACT

Negative selection algorithms for hamming and real-valued shape-spaces are reviewed. Problems are identified with the use of these shape-spaces, and the negative selection algorithm in general, when applied to anomaly detection. A straightforward self detector classification principle is proposed and its classification performance is compared to a real-valued negative selection algorithm and to a one-class support vector machine. Earlier work suggests that real-value negative selection requires a single class to learn from. The investigations presented in this paper reveal, however, that when applied to anomaly detection, the real-valued negative selection and self detector classification techniques require positive and negative examples to achieve a high classification accuracy. Whereas, one-class SVMs only require examples from a single class.

## Categories and Subject Descriptors

I.5.2 [**Pattern Recognition**]: Design Methodology—*Classifier design and evaluation, Pattern analysis*

## General Terms

Algorithms

## Keywords

Artificial Immune Systems, Negative Selection, Positive Selection, Anomaly Detection, One-Class SVM

## 1. INTRODUCTION

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 examples 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 and the task is to estimate a function $f : \mathbb{R}^N \rightarrow \{0, 1\} \ni Y$, using training data pairs generated i.i.d.[1] according to an unknown probability distribution $P(X, Y)$

$$(X_1, Y_1), \ldots, (X_n, Y_n) \in \mathbb{R}^N \times Y, \quad Y = \{0, 1\}$$

such that $f$ will correctly classify unseen examples $(X, Y)$. If the training data *only* consists of examples from one class and the test data contains examples from two or more classes, the classification task is called *anomaly detection* or *novelty detection*. An example of anomaly detection is machine fault recognition, where only training data containing normal behavior is available, as it is difficult or impossible to obtain abnormal behavior.

From the viewpoint of *pattern classification* and *anomaly detection*, the immune system is a highly dynamic system which recognizes and classifies many different and unseen molecules at any given time and therefore has been a rich source of inspiration for pattern classification [28, 25] and anomaly detection [4, 22, 18].

This paper is organized as follows: In section 2, we explain how the immune system discriminates foreign cells from self cells and is able to recognize possible antigens with a limited number of antibodies. In-depth immune specific details are omitted, since they are not relevant to this work. In section 3 and 4, negative selection over two different shape-spaces is presented and resulting problems are discussed. In section 5, we propose a straightforward classification method inspired by the positive selection immune system principle. In section 6 we discuss resulting problems and open questions in the real-valued negative selection algorithm. In section 7, the Support Vector Machine (SVM) and the modified one-class SVM variant are introduced. In section 8, we compare our proposed algorithm to the real-valued negative selection algorithm proposed by Ji and Dasgupta [18] and to a one-class SVM.

## 2. IMMUNE SYSTEM

The main task of the immune system is to defend the body against disease caused by pathogens. Pathogens are foreign substances like viruses, fungi, parasites and bacteria

[1] independently drawn and identically distributed


*GECCO'05,* June 25–29, 2005, Washington, DC, USA.

which attack the body and can lead to death in the worst case. 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[2], 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[3] and the somatic hypermutation[4] achieves a potential repertoire (complete lymphocytes diversity) of about $10^{14}$ B-lymphocytes and $10^{18}$ T-lymphocytes [17]. Experiments show that each day approximately $10^7$ new lymphocytes are generated [5]. Since at any given time, only $10^8$ different lymphocytes are available, and these are turned over a rate of $10^7$ per day, it will take 10 days to generate a complete new lymphocyte repertoire and on average it takes 3.5 days to generate a lymphocyte which matches a specific antigen [5].
As explained above, the negative selection process eliminates self reactive lymphocytes. In addition, the immune system also performs a *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 (major histocompatibility complex) molecules which are able to present intruded viral peptides on the cells surface. The MHC presented information consists of non-self peptides, but also of self peptides (called self-MHC molecules). The process of positive selection assures that those lymphocytes are selected, whose antibodies are capable of recognizing and binding with self-MHC molecules associated with non-self peptides. One can say that positive selection allows the immune system to recognize self.

# 3. NEGATIVE SELECTION PRINCIPLE

The negative selection principle is a mechanism of the immune system to protect the body against self reactive lymphocytes. This principle inspired Forrest et al. [22] to propose a negative selection algorithm to detect data manipulation caused by computer viruses. The basic idea is to generate a number of detectors in the complementary space and then to apply these detectors to classify new (unseen) data as self (no data manipulation) or non-self (data manipulation). The negative selection algorithm proposed by Forrest et al. is summarized in the following steps.

**Algorithm 1**
Given a shape-space $U$, self set $S$ and non-self set $N$, where

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

1. Define self as a set $S$ of elements of length $l$ 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 $S$.

This original algorithm has some drawbacks, more thoroughly discussed in [11]. To summarize, the algorithm is inefficient, since a vast number of randomly generated detectors need to be discarded, before the required number of suitable once are obtained — this is a simple random search. And second, the algorithm is defined over a shape-space associated with an affinity function which induces additional problems, discussed in the following section.

## 3.1 Shape-Space and Affinity

The notion of *shape-space* was introduced by Perelson and Oster [21] 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. In this paper, the Hamming shape-space and real-valued shape-space are considered, as they are the most commonly used for negative selection. A detailed overview of other shape-shapes and affinity functions used in artifical immune systems is provided in [6].

## 3.2 Hamming Shape-Space and R-chunk Matching

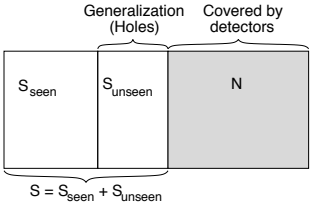
The Hamming shape-space $U_l^\Sigma$ is built out of all elements of length $l$ over an finite alphabet $\Sigma$. In the original negative selection algorithm proposed by Forrest et al. [22] it is defined over the binary alphabet $\Sigma = \{0, 1\}$. The *r-contiguous* [20] matching rule was applied to determine the affinity between a detector and an element. Informally, two elements, with the same length, match if at least $r$ contiguous characters are identical. In succeeding works [2, 12] the performance of different matching rules over the binary alphabet are compared and the *r-chunk* [12] matching rule achieved the highest matching performance compared to the other matching rules over the binary alphabet. The r-chunk matching rule is an improved variant of the r-contiguous matching rule and is defined as follows :

Given a shape-space $U_l^\Sigma$, which contains all elements of length $l$ over an alphabet $\Sigma$ and a shape-space $D_r^\Sigma$.

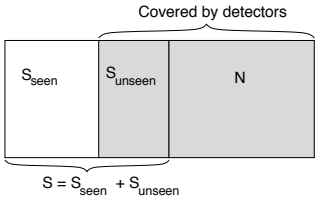
*Definition 1.* An element $e \in U_l^\Sigma$ with $e = e_1 e_2 \dots e_l$ and detector $d \in \mathbb{N} \times D_r^\Sigma$ with $d = (p, d_1 d_2 \dots d_r)$, for $r \leq l$, $p \leq l - r + 1$ *match with r-chunk rule* if $e_i = d_i$ *for* $i = p, \dots, p + r - 1$.

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

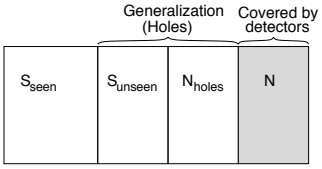
[2]building blocks for antigens and molecules
[3]Variability in antibodies caused by differences in the exact crossover point during the joining of gen segments
[4]The occurrence of a high level of mutation in the variable regions of antibody building blocks




(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 elements.




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

The Hamming shape-space and the affinity functions r-contiguous and r-chunk are very appealing from a biological perspective, as they allow for a straightforward mathematical analysis [7, 9] and provide a simple abstraction of the binding between antibody and antigen [20].

## 3.3 Generalization by Holes

All matching rules, including the r-chunk rule investigated in [12], cause undetectable elements (termed holes). Holes are elements of $N$ or self elements not seen during the training phase. For these elements no detectors can be generated 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. 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. 1(b)). In contrast, a large number of holes implies that there are a large number of unseen self elements and non-self elements which are not covered by the detector set. Therefore the detector set *underfits* (see Fig. 1(c)). Balthrop et al. [2] proposed a method (termed crossover-closure) to find holes for the r-chunk matching rule by given parameters $l, r$ and $S$. We summarize it algorithmically (see algorithm 2) and show (see Fig. 2) an illustrative example of the construction for a given set $S = \{01010, 01110, 10001, 11010, 11011\}, l = 5, r = 3$.

**Algorithm 2**
Given a set $S = \{S_1, S_2, \ldots, S_n\}$ of self strings, element length $l$ and matching length $r$ :

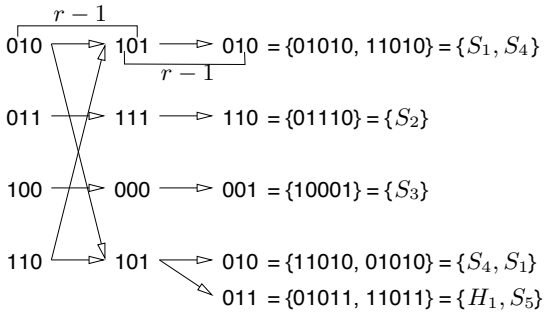



**Figure 2: Construction to find holes for $r = 3$ and $l = 5$ for the r-chunk matching rule for $S = \{01010, 01110, 10001, 11010, 11011\}$. For element $H_1 = \{01011\}$ it is not possible to generate a detector which recognizes $H_1$ as non-self**

1. Cut $S_i$ in $l - r + 1$ substrings $S_{i,j} := S_i[j, \ldots, r-1+j]$ for $j = 1, \ldots, l - r + 1$ , $i = 1, \ldots, n$.
2. 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 $j = 1, \ldots, n - 1$.
3. Traverse and shuffle coincident substrings $S_{i,1}, \ldots, S_{i,l-r+1}$ for $i = 1, \ldots, n$ to obtain the set of self strings $S$ and the set $H$ of undetectable elements.

Algorithm 2 shows, that holes arise in commonly occurring distinct self strings. Therefore, the number of holes increases with the number of self strings and can be controlled with parameter $r$. Esponda et al. [9] presented formulas to approximate the number of holes by given $|S|, l$ and $r$. Using Esponda's formulas, the approximation can be simplified as one term :

$$H(|S|, l, r) \quad = \quad T_1 \cdot T_2 - |S| \tag{1}$$

where

$$T_1 \quad = \quad 2^r - 2^r\left(1 - \tfrac{1}{2^r}\right)^{|S|}$$

$$T_2 \quad = \quad \left[2 - \tfrac{1}{2}\left(1 - \tfrac{1}{2^{r+1}}\right)^{4|S|} - 2\left(\left(1 - \tfrac{1}{2^{r+1}}\right)^{2|S|} - \left(1 - \tfrac{1}{2^{r+1}}\right)^{3|S|}\right)\right]^{(l-r)}$$

Using the assumption that $|S| < |N|$, the number of holes increases exponentially for $r \to l, \ldots, 1$. This implies, that the r-chunk matching rule will underfit exponentially. To obtain a linear under/overfitting behavior, $r$ must be close to $l$. However this makes the detector generation computationally infeasible, since all proposed detector generation algorithms[5] [1, 26] have a runtime complexity, which is exponential in $r$. Therefore, the hamming shape-space and the r-chunk matching rule are *only* appropriate and applicable for anomaly detection problems for small values of $l$ (e.g. $0 < l \leq 32$).

# 4. REAL-VALUED NEGATIVE SELECTION ALGORITHM

The principle of generating detectors in the complementary space and using these detectors to classify elements is adaptable to other shape-spaces. Gonzalez et al. [13, 14]

[5] r-contiguous and r-chunk

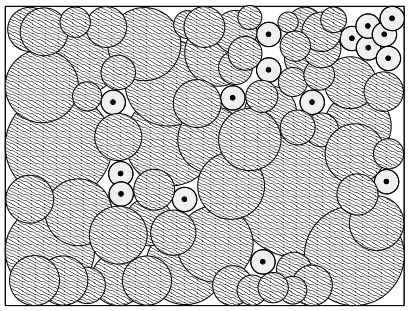


**Figure 3: Real-Valued Negative Selection principle with variable size detectors for a two-dimensional space. The non-self space which is covered by the detectors is pictured as the shaded area. The self elements are pictured as white circles with a black center**

propose a negative selection algorithm, which operates on a unitary hypercube $[0,1]^n$. A detector $d = (c, r_{ns})$ has a center $c \in [0,1]^n$ and a non-self recognition radius $r_{ns} \in \mathbb{R}$. Furthermore, every self element $s = (c, r_s)$ has a center and a self radius $r_s$. The motivation behind the self-radius is to consider other elements as self, which are close to the self-center. If an element lies within a detector (hypersphere), then it is classified as non-self, otherwise as self. An element $e$ lies within a detector $d = (c, r_{ns})$, if the Euclidean distance $dist(c,e) = \left(\sum_{i=1}^{n}(c_i - e_i)^2\right)^{1/2} < r_{ns}$.

## 4.1 Variable-Sized Real-Valued Negative Selection Algorithm

Ji and Dasgupta [18] proposed a real-valued negative selection algorithm with variable size detectors (termed *V-Detector*), where the center of a detector is positioned randomly and must not lie within the hypersphere of a self-element. The radius is dynamically resized to the next self-element margin (see Fig. 3). The algorithm terminates, if a predefined number of detectors is generated or a predetermined proportion of non-self space is covered. The classification performance of this algorithm is tested on different data sets, for example Iris Fisher and Biomedical [27], and is compared to other negative selection algorithms.

# 5. REAL-VALUED SELF DETECTOR CLASSIFICATION

In this section, we propose a straightforward classification method, termed Self Detector Classification, which is inspired by the *positive selection* immune system principle and the 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"*.

The self detector classification operates on a unitary hypercube $[0,1]^n$. Self-elements are considered as self-detectors with an *a-priori* given self-radius[6] $r_s$, which is determined in the training phase by means of the ROC analysis. An element is classified as self, if it lies within a self-detector, otherwise as non-self. The self-detector classification method is

[6]which is also necessary in the real-valued negative selection

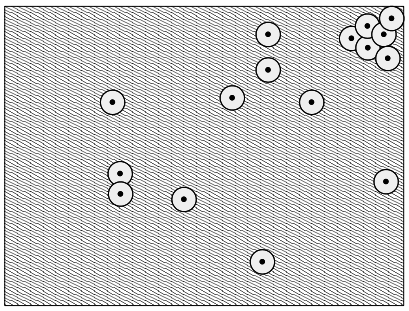


**Figure 4: Real-Valued Self-Detector Classification for a two-dimensional space. The covered non-self space is pictured as the shaded area. The self elements are pictured as grey circles with a center and a self-radius $r_s$**

| | | Actual P | Actual N |
|---|---|---|---|
| Classifier Predict | P | TP | FP |
| | N | FN | TN |

**Figure 5: General Confusion Matrix**

illustrated geometrically in figure 4 and summarized in the following steps :

**Algorithm 3**

1. For training examples $\hat{S} \subseteq S$ generate self detectors $d_{\hat{S}}$ with $r_s = 0$.
2. Perform $(\Delta i)^{-1}$ training classification runs over $\hat{U} \subseteq U$ and find $r_s$, which yields the minimum error (see Eq. (2))
3. Classify new examples $x \in U$ as self, if Euclidean $dist(d_{\hat{S}}, x) < r_s$, otherwise as non-self.

In the following section we motivate the use of ROC analysis and propose a method to find an optimal self-detector radius.

## 5.1 ROC Analysis

ROC (Received Operating Characteristic) analysis is used 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. Given a different data set which 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 inapplicable. 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 the 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 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 $2 \times 2$ *confusion matrix* can be constructed representing the dispositions of the set of examples (see Fig. 5). Using the matrix values, the *detection rate* (true positive rate) and the *false alarm rate* (false positive rate) of a classifier is calculated as :

$$\text{detection rate} = \frac{\text{non-self correctly classified}}{\text{total non-self}} = \frac{\text{TP}}{\text{TP+FN}}$$

$$\text{false alarm rate} = \frac{\text{self incorrectly classified}}{\text{total self}} = \frac{\text{FP}}{\text{FP+TN}}$$

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. A point at $(0, 1)$ represents a perfect classifier (100% detection rate and 0% false alarm rate).

## 5.2 Determine Optimal Self-Radius

The ROC analysis provides a technique to evaluate the classification performance of classifiers. To find a self-detector radius $r_s$ which yields the overall best balance between detection rate and false alarm rate, $r_s$ is initialized with a small start value (e.g. 0.01) and increased after one training classification run by $r_s = r_s + \Delta i$ until $r_s \geq \max$ (e.g. $\max = 1.0$). For every $\Delta i$ 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. 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.

$$\text{minimum error} = \min\left(1 - (d_i - f_i)\right), \quad \forall i \qquad (2)$$

# 6. REVIEW OF REAL-VALUED NEGATIVE SELECTION

From a geometrical point of view (compare figure 3 and 4), the real-valued negative selection method first fills the space with detector-circles[7] (shaded area) and considers elements as non-self if they lie within a detector-circle. In contrast, the self-detector classification 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. From the paper [18], it is not clear how to determine the self-radius $r_s$, with a training set which contains only self elements, without any techniques (probabilistic or deterministic) to obtain information about the other class. In [18] the self-radii $r_s = 0.1$ and $r_s = 0.05$ are used without any explanation. 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 by only been given training data from one class, a coherence between estimated probability density function and radius must be found.

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 [14] 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. One must raise the question of whether this real-valued negative selection algorithm is an appropriate approach for anomaly detection. From our point of view, it is an approach which requires two classes in the learning phase in order to determine the self-radius. In addition, it has a high runtime complexity to find the optimal distribution of the detectors.

# 7. SUPPORT VECTOR MACHINE

For comparative purposes, we briefly introduce the principle of Support Vector Machines, but omit some mathematical details. For an in-depth introduction see [19, 23]. A Support Vector Machine (SVM) is a machine learning paradigm for a two-class classification problem. In this paradigm, the input data is mapped into a higher-dimensional feature space through an *a-priori* chosen non-linear mapping[8], where a linear decision region is constructed. This decision region when mapped back into the original feature space can take a non-linear form. Loosely speaking, the SVM algorithm looks for the separating hyperplane with the largest margin, where the hyperplane only depends on a subset of training examples, called *support vectors*. More precisely, given training data $\{x_i, y_i\}$, $i = 1, \ldots, l$, $x_i \in \mathbb{R}^n$ , $y_i \in \{\pm 1\}$ and a nonlinear mapping in the feature space $\mathcal{F}$

$$\begin{aligned} \Phi : \mathbb{R}^N &\rightarrow \mathcal{F} \\ x &\mapsto \Phi(x) \end{aligned}$$

SVM classifiers are based on the class of hyperplanes :

$$f(x) = (w \cdot x) + b \qquad w \in \mathbb{R}^n, \ b \in \mathbb{R}$$

corresponding to decision functions :

$$f(x) = \text{sign}((w \cdot x) + b)$$

Perfect separating hyperplanes, which classify without training error in $\mathcal{F}$ are :

$$y_i((w \cdot \Phi(x_i)) + b) \geq 1, \quad i = 1, \ldots, l$$

Introducing Lagrange multipliers $\alpha_i \geq 0$, $i = 1, \ldots, l$ this can be formulated as the dual quadratic optimization problem :

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \\ \text{subject to} \quad & \alpha_i \geq 0, \ i = 1, \ldots, l \\ & \sum_{i=1}^{l} \alpha_i \ y_i = 0 \end{aligned}$$

[7] hyperspheres for higher dimensions

[8] the kernel function $k$

And one obtains the nonlinear decision function :

$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^{l} y_i \alpha_i (\Phi(x) \cdot \Phi(x_i)) + b\right) = \operatorname{sgn}\left(\sum_{i=1}^{l} y_i \alpha_i k(x, x_i) + b\right)$$

To avoid overfitting effects, slack-variables are used to relax the hard-margin constraints:

$$y_i((w \cdot \Phi(x_i)) + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, l$$

With the slack-variables a classifier can be constructed, which generalizes well.

## 7.1 One-Class Support Vector Machine

Schölkopf et al. [24] propose a method (termed one-class SVM) to adapt the SVM methodology to a one-class learning system, that means *only* examples from one class are required. The one-class SVM first maps the input data into a higher-dimensional feature space via a kernel function and treats the origin as the only member of the second class. In addition, a fraction of "outliers" are allowed, which lie between the origin and the hyperplane, while the hyperplane has maximum distance to the origin. 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 and therefore, for a new point $x$, the value $f(x)$ is determined by evaluating which side of the hyperplane it falls on in feature space.
More precisely, it can be formulated also as a dual optimization problem :

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j k(x_i, x_j) \\ \text{subject to} \quad & 0 \leq \alpha_i \leq 1/(\nu l),\ i = 1, \ldots, l \\ & \sum_{i=1}^{l} \alpha_i = 1 \end{aligned}$$

Obtaining the decision function

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

which will be positive for most examples $x_i$ in the training set. The value of $\rho$ can be recovered by exploiting the fact, that for any Lagrange multipliers $\alpha_i$, the corresponding pattern $x_i$ satisfies

$$\rho = (w \cdot \Phi(x_i)) = \sum_j \alpha_j k(x_j, x_i)$$

For our experiments, we used the one-class SVM implementation LIBSVM 2.6 [3, 16]. LIBSVM is a program, which provides several SVM algorithms for classification and regression, including Schölkopfs proposed one-class SVM. The default kernel (radial basis function), $\nu = 0.05$ and the default values of the parameters for the one-class SVM are used.

# 8. CLASSIFICATION RESULTS AND COMPARISON STUDY

## 8.1 Data Sets and Experimental Settings

In line with previous work [18], we perform our experiments on the Iris-Fisher and Biomedical data sets, which can be downloaded from [27]. 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 set 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%[9] of randomly drawn elements from one class. In the test phase, all elements in 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 [18] 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 perform 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. (2). 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.

## 8.2 Results

It is difficult to analyze the classification results done in [18] from an anomaly detection point of view, because it is not clear how the self-radius is determined. Furthermore, a one-class classifier, when compared to a two-class classifier, has a smaller amount of information available, in order to build up the mapping function and therefore a comparison is difficult as well. As shown in Table 1, the self-detector classification method outperforms the real-valued negative selection 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%. Similar results are obtained for the Biomedical data set (see Table 2), but with the exception that the false positive rate is higher. With radius $r_s = 0.09$ the overall classification performance is higher. The ROC analysis shows, that our results outperform negative selec-

[9] only Biomedical data set

**Table 1: Classification Results 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 |
| | ocSVM | 100 | 0.00 | 4.00 | 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 |
| | ocSVM | 100 | 0.00 | 5.56 | 4.88 |
| | 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 |
| | ocSVM | 90.00 | 0.00 | 8.38 | 1.60 |
| | 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 |
| | ocSVM | 90.57 | 2.92 | 7.72 | 3.84 |
| | 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 |
| | ocSVM | 75.00 | 0.00 | 5.28 | 0.96 |
| | 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 |
| | ocSVM | 78.42 | 8.35 | 9.36 | 4.04 |
| | 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 2: Classification Results 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 |
| | ocSVM | 40.30 | 0.00 | 5.43 | 0.59 |
| | 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 |
| | ocSVM | 40.27 | 12.22 | 6.24 | 2.53 |
| | 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 |
| | ocSVM | 45.18 | 14.73 | 8.36 | 4.16 |
| | 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 |

tion on overall classification accuracy. The low classification accuracy of the one-class SVM can be explained by the fact, that the SVM was trained with too fews examples. In our experiments the value $\nu = 0.05$ characterizes the fraction of support vectors and outliers. This value was derived from Theorem 7 (see [24], page 10), which gives some confidence in a probabilistic sense, but depends on the number of training examples.

# 9. CONCLUSION

We have briefly introduced pattern classification and anomaly detection and have shown the connection to artificial immune systems. Additionally, we described the fact, that the immune system is able to detect all possible non-self proteins with a limited number of antibodies by performing a continual turnover of new and old lymphocytes and negative selection. This principle motivated computer scientists to develop immune inspired algorithms, which work in a similar way. We have shown that the negative selection algorithm, defined over the hamming shape-space, is not well suited for real-world anomaly detection problems. The generated detector set underfits exponentially for small values of $r$. To avoid this underfitting behavior, the matching threshold value $r$ must lie near $l$, but this makes the detector generation infeasible, since all proposed detector generating algorithms have a runtime complexity which is exponential in $r$. Moreover, we have critically compared the real-valued negative selection algorithm with variable-sized detectors to a proposed straightforward self-detector classification method, inspired by the immune system principle of positive selection. Our results reveal, that the classification performance of both methods depend crucially on the self-radius. From our point of view a proper self-radius can only be determined, when examples from the anomalous class are available. In addition we compared the classification performance of both approaches and obtain results which question the appropriateness of the real-valued negative selection approach. The real-valued negative selection was proposed to overcome the complexity problems of the hamming shape-space. It is currently an open question, how this approach works in high dimensional spaces, where for instance the one-class SVM provides good results.

# 10. ADDITIONAL AUTHORS

Additional authors: Claudia Eckert (Darmstadt University of Technology, Hochschulstr. 10, 64289 Darmstadt, Germany) email: `eckert@sec.informatik.tu-darmstadt.de`.

# 11. REFERENCES

[1] M. Ayara, J. Timmis, L. de Lemos, R. de Castro, and R. Duncan. Negative selection: How to generate detectors. In *1st Int. Conf. on Artificial Immune Systems*, pages 89–98, September 2002.

[2] J. Balthrop, F. Esponda, S. Forrest, and M. Glickman. Coverage and generalization in an artificial immune system. In *GECCO 2002: Proc. of the Genetic and Evolutionary Computation Conf.*, pages 3–10, New York, 9-13 July 2002. Morgan Kaufmann Publishers.

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

[4] D. Dasgupta, S. Forrest. Novelty Detection in Time Series Data Using Ideas from Immunology. In *Proc. of the 5th Int. Conf. on Intelligent Systems*. IEEE Computer Society Press, 1996.

[5] D. G. Osmond. The turn-over of B-cell populations. *Immunology Today*, 14(1):34–37, 1993.

[6] L. N. de Castro and J. Timmis. *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer-Verlag, 2002.

[7] P. D'haeseleer. An immunological approach to change detection: Theoretical results. In *Proc. 9th IEEE Computer Security Foundations Workshop*, pages 18–26, 1996.

[8] Elad Yom-Tov. An Introduction to Pattern Classification. In *Advanced Lectures on Machine Learning*, volume 3176 of *LNCS*, pages 1–20. Springer-Verlag, 2004.

[9] F. Esponda, S. Forrest, and P. 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.

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

[11] A. Freitas and J. Timmis. Revisiting the Foundations of Artificial Immune Systems: A Problem Oriented Perspective. In *Proc. of the 2nd Int. Conf. on Artificial Immune Systems*, volume 2787 of *LNCS*, pages 229–241. Springer, September 2003.

[12] F. González, D. Dasgupta, and G. Gomez. The effect of binary matching rules in negative selection. In *Genetic and Evolutionary Computation – GECCO-2003*, volume 2723 of *LNCS*, pages 195–206, Chicago, 12-16 July 2003. Springer-Verlag.

[13] F. González, D. Dasgupta, and R. Kozma. Combining negative selection and classification techniques for anomaly detection. In *Congress on Evolutionary Computation*, pages 705–710. IEEE, May 2002.

[14] F. A. González, D. Dasgupta, and L. F. Niño. A Randomized Real-Valued Negative Selection Algorithm. In *Proc. of the 2nd Int. Conf. on Artificial Immune Systems (ICARIS)*, volume 2787 of *LNCS*, pages 261–272, Edinburgh, UK, 2003. Springer-Verlag.

[15] S. A. Hofmeyer. An Interpretative Introduction to the Immune System, 2000.

[16] C.-W. Hsu, C.-C. Chang, and C.-J. Lin. *A Pratical Guide to Support Vector Classification (http://www.csie.ntu.edu.tw/∼cjlin/papers/guide/guide.pdf)*, July 2003.

[17] C. A. Janeway, P. Travers, M. Walport, and M. Shlomchik. *Immunologie*. Spektrum Akademischer Verlag, 2002.

[18] Z. Ji and D. Dasgupta. Real-valued negative selection algorithm with variable-sized detectors. In *Genetic and Evolutionary Computation – GECCO-2004, Part I*, volume 3102 of *LNCS*, pages 287–298, Seattle, WA, USA, 26-30 June 2004. Springer-Verlag.

[19] K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. An Introduction to Kernel-Based Learning Algorithms. *Transactions on Neural Networks*, 12(2):181–201, 2001.

[20] J. K. Percus, O. E. Percus, and A. S. Perelson. Predicting the Size of the T-Cell Receptor and Antibody Combining Region from Consideration of Efficient Self-Nonself Discrimination. *Proc. of National Academy of Sciences USA*, 90:1691–1695, 1993.

[21] A. S. Perelson and G. 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.

[22] S. Forrest, A.S. Perelson, L. Allen, R. Cherukuri. Self-Nonself Discrimination in a Computer. In *Proc. of the 1994 IEEE Symposium on Research in Security and Privacy*. IEEE Computer Society Press, 1994.

[23] B. Schölkopf. Statistical learning and kernel methods. Technical Report MSR-TR-2000-23, Microsoft Research (MSR), Feb. 2000.

[24] B. Schölkopf, J. C. Platt, S.-T. Shawe-Taylor, A. J. Smola, and W. Williamson. Estimating the support of a high-dimensional distribution. Technical Report MSR-TR-99-87, Microsoft Research (MSR), Nov. 1999.

[25] A. Secker, A. A. Freitas, and J. Timmis. AISEC: An artificial immune system for e-mail classification. In *Proc. of the 2003 Congress on Evolutionary Computation CEC2003*, pages 131–138, Canberra, 8-12 Dec. 2003. IEEE Press.

[26] T. Stibor, K. M. Bayarou, and C. Eckert. An investigation of R-chunk detector generation on higher alphabets. In *Genetic and Evolutionary Computation – GECCO-2004, Part I*, volume 3102 of *LNCS*, pages 299–307, Seattle, WA, USA, 26-30 June 2004. Springer-Verlag.

[27] P. Vlachos. StatLib. http://lib.stat.cmu.edu.

[28] A. Watkins and L. Boggess. A new classifier based on resource limited artificial immune systems. In *Proc. of the 2002 Congress on Evolutionary Computation CEC2002*, pages 1546–1551. IEEE Press, 2002.