A Comparative Study of Real-Valued Negative Selection to Statistical Anomaly Detection Techniques

Thomas Stibor[1] and Jonathan Timmis[2] and Claudia Eckert[1]

[1] Department of Computer Science
Darmstadt University of Technology
{*stibor,eckert*}*@sec.informatik.tu-darmstadt.de*
[2] Departments of Electronics and Computer Science
University of York, Heslington, York
jt517@ohm.york.ac.uk

Abstract. The (randomized) real-valued negative selection algorithm is an anomaly detection approach, inspired by the negative selection immune system principle. The algorithm was proposed to overcome scaling problems inherent in the hamming shape-space negative selection algorithm. In this paper, we investigate termination behavior of the real-valued negative selection algorithm with variable-sized detectors on an artificial data set. We then undertake an analysis and comparison of the classification performance on the high-dimensional KDD data set of the real-valued negative selection, a real-valued positive selection and statistical anomaly detection techniques. Results reveal that in terms of detection rate, real-valued negative selection with variable-sized detectors is not competitive to statistical anomaly detection techniques on the KDD data set. In addition, we suggest that the termination guarantee of the real-valued negative selection with variable-sized detectors is very sensitive to several parameters.

1 Introduction

The field of Artificial Immune Systems (AIS) has seen the development of many algorithms. One of the major algorithms developed within AIS is the negative selection algorithm, first proposed by Forrest et al. [1] and then subsequently developed over the years [2,3,4,5]. This paper investigates the real-valued negative selection algorithm with variable-sized detectors [6] and its applicability to network intrusion traffic. The negative selection algorithm is oft cited for its potential use in intrusion detection problems due to its ability to generate a set of *detectors* from a single class of data (usually the *normal* network traffic), that is capable of identifying possible intrusions. However, there remains little work in the literature regarding the application of the negative selection algorithm with variable-sized detectors to network intrusion detection. This paper undertakes a comparative study between the negative selection with variable-sized detectors, another simple AIS algorithm, positive selection, and two well established

statistical techniques. Our investigations reveal, that whilst appealing, negative selection with variable-sized detectors does not appear to perform as well as the more established techniques. The paper is organized as follows: Section 2 provides a simple overview of anomaly detection. Then, the immune negative selection principle and basic negative selection algorithm are briefly explained in section 3. Section 3.1 provides a review of the real-valued negative selection algorithm with variable-sized detectors. This is followed by a simple real-valued positive selection algorithm in section 3.2. Through the use of an artificial data set, in section 4 we explore the termination behavior of the real-valued negative selection algorithm. For comparative purposes, two novelty detection techniques are described in section 5. This is then followed by an analysis of the classification performance of the negative selection and is compared to the positive selection and to statistical novelty detection techniques in section 6.

2 Anomaly Detection

Anomaly detection, also referred to as novelty detection [7], outlier detection [7] or one-class learning [8,9], is a classification technique, which is used for classifying data 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. More formally, the task is to find a functional mapping $f : \mathbb{R}^N \rightarrow \{\mathcal{C}_0, \mathcal{C}_1\}$, using training data samples generated i.i.d.[3] according to an unknown probability distribution $P(\mathbf{x}, y)$

$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n) \in \mathbb{R}^N \times Y, \qquad Y = \{\mathcal{C}_0, \mathcal{C}_1\}$$

such that f will correctly classify unseen examples $(\mathbf{x}, y)$. In the worst case, the training set contains *only* normal samples $(\mathbf{x}, y \in \mathcal{C}_0)$ and the challenge is to detect abnormal samples $(\mathbf{x}, y \in \mathcal{C}_1)$ with the function f which was trained[4] with only normal samples.

3 Negative Selection Principle

The negative selection principle is a process that takes place in the thymus gland, which helps to filter self reactive lymphocytes away from entering the lymphatic system. This principle inspired Forrest et al. [1] to propose a negative selection algorithm to detect data manipulation caused by computer viruses. The basic idea was to generate a number of detectors in the complementary space and then

[3] independently drawn and identically distributed

[4] the parameters are determined, based on the seen training samples

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.

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.

3.1 Real-Valued Negative Selection

The idea to generate detectors in the complementary space for continuous data, was proposed informally by Ebner et al. [10] and formally by Gonzalez et al. [4,5]. The 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 introduced 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[5] $\mathbf{e}$ lies within a detector $d = (\mathbf{c}_d, r_{ns})$, if the Euclidean distance $dist(\mathbf{c}, \mathbf{e}) = \left(\sum_{i=1}^{n}(c_i - e_i)^2\right)^{1/2} < r_{ns}$. Ji and Dasgupta [6] proposed a real-valued negative selection algorithm with variable-sized detectors (termed *V-Detector*) — the algorithm is presented in the appendix and illustrated in figures 1(b), 1(c), 1(d). The algorithm randomly determines a center of a detector which must not lie within the hypersphere of a self-element. The radius is dynamically resized until the boundary of the region comes in contact with a self-element. 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 paper, we employed the algorithm proposed by Ji and Dasgupta [6].

3.2 Real-Valued Positive Selection

The real-valued positive selection algorithm was informally described by Ebner et al. [10] and formally by Stibor et al. [11]. The main difference to the negative selection is that *no* non-self detectors exists. Instead, each self element contains a self-detector which classifies unseen elements. An 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, in contrast to the real-valued negative selection.

[5] n dimensional point

4 Investigating the Real-Valued Negative Selection Algorithm with Variable-Sized Detectors

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 toy problem. We created a simple two-dimensional artificial data set with 9 self elements (see Fig. 1(a)). We ran the algorithm using the same parameters as [6] :

$$\text{Maximum Self Coverage } MSC = 99.99\,\%$$
$$\text{Maximum Number of Detectors } T_{max} = 1000$$

The results are visualized in figure 1. Figure 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 noted 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. 1(c), 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 $100\,\%$. Consequently, this increases the runtime complexity required to generate detectors. This is now analyzed in the following section.

4.1 Algorithm Termination

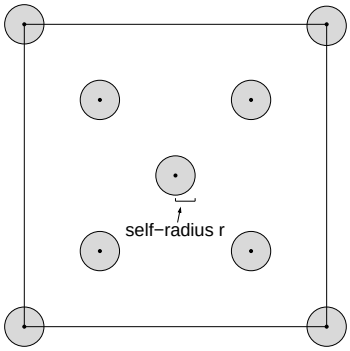
First, it can be seen (algorithm 1), that the termination condition in line 22 is not useful, because T *never* has a value higher than 1. Once increased to 1 (see line 21), T is set to 0 (see line 5) in the same outer repeat loop and therefore, the termination condition is line 22 is never satisfied.

Another algorithm termination is reached (see line 11), when the condition $t \geq 1/(1 - c_0)$ is satisfied. Let $\mathbf{x} \in \Delta$ 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 — 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 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 $\mathbf{x}_{j+1} \notin \Delta$.

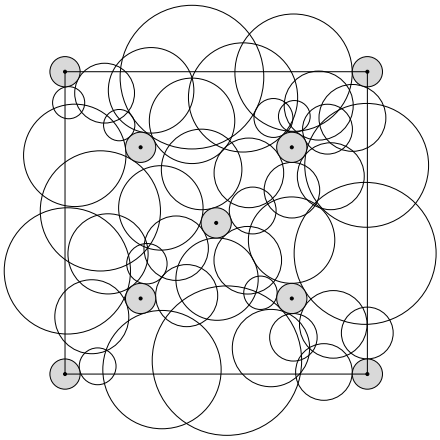
The probability to find in $j+1$ random sampling trials j successes before the first failure is :

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

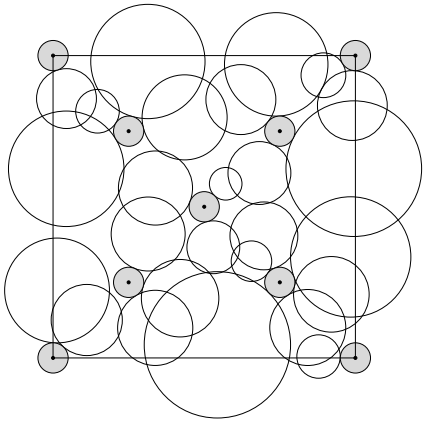
Term 1 only depends on p and j. The higher the number of self elements or the larger the self-radius, the lesser the probability of finding a sample sequence



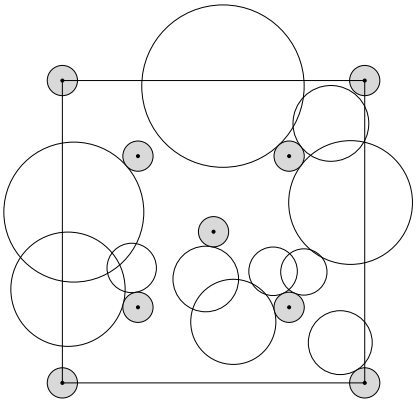
(a) An artificial data set containing 9 self-elements with self-radius r_s pictured as the grey circles with a black center $\mathbf{c}_s$. It contains no V-detectors.



(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 $r_s = 0.05$, $c_0 = 80\,\%$.

Fig. 1. The real-valued negative selection algorithm with variable-sized detectors applied on an artificial data set for different estimated coverages.

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. In the work [6], the runtime complexity of the V-Detector algorithm is estimated by $O(|D| \cdot |S|)$ *without* a probabilistic approach. As the detectors are generated randomly, we suggest a probabilistic runtime complexity estimation.

A final point to note is that the simple random generation and coverage estimation method employed, induces the steady space coverage problem. This is explored with a high-dimensional dataset and discussed in section 6.1.

5 Statistical Novelty Detection

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 [12]. This classification decision is based on the Bayes theorem and can be written as :

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

where $P(C_k)$ is the *prior* probability of a sample belonging to each of the classes C_k 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. 2), which is constant over some large region of the input space [13]. 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 $\mathcal{C}_1$.

5.1 Parzen-Window Estimators

Parzen-Window is a nonparametric method for estimating density functions [14]. 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

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

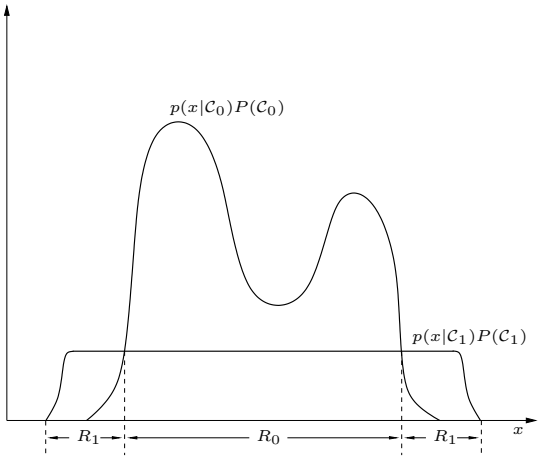


Fig. 2. Bayesian decision for determining whether an input sample belongs to class $\mathcal{C}_0$ (falling in region $\mathcal{R}_0$) or $\mathcal{C}_1$ (falling in region $\mathcal{R}_1$) modeled with class-conditional density functions.

where K is a kernel function which must satisfies the condition

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

and h the window width (also called smoothing parameter). For our experiments we choose 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 control the degree of smoothness of the estimated density function. In our experiments (see section 6), we used the proposed variance parameter $\sigma = 0.01$ [15].

Through the combination of the Parzen-Window method and the Bayes classification method, it is possible to 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[6] $p_u(\mathbf{x})$ is *a-priori* modeled. An unseen sample which falls in region $\mathcal{R}_0$ is classified as normal, otherwise it is said to falls in the region $\mathcal{R}_1$ and is classified as an anomalous sample.

5.2 One-Class Support Vector Machine

In many applications it is sufficient to estimate the support of the probability distribution, as opposed to the full density. A one-class Support Vector Machine

[6] the threshold

(termed one-class SVM) avoids estimating the full density. Instead, it estimates quantiles of the multivariate distribution, i.e. its support. The one-class SVM maps the input data into a higher-dimensional feature space $\mathcal{F}$ via a nonlinear 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 (the hyperplane has maximum distance to the origin, see Fig. 3). 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 $\mathbf{x}$, the value $f(\mathbf{x})$ is determined by evaluating which side of the hyperplane it falls on, in feature space.

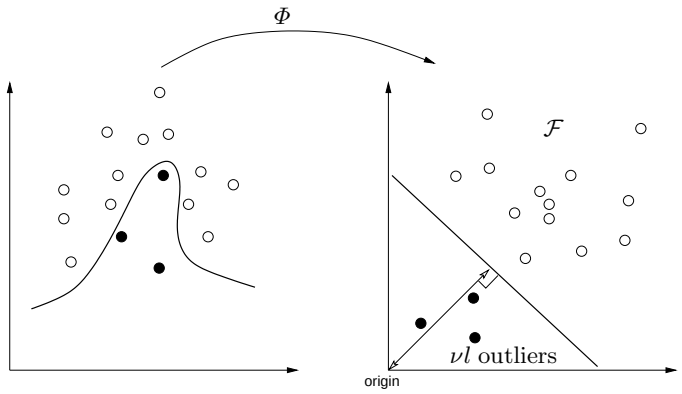


Fig. 3. Map the training data into a high-dimensional feature space $\mathcal{F}$ via Φ. Construct a separating hyperplane with maximum distance to the origin, with the constrains that νl outliers lie between the origin and the hyperplane.

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

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

where $\alpha_{1\ldots l}$ are Lagrange multipliers, k the kernel function and $\mathbf{x}_{1\ldots l}$ the training samples.
By solving this optimization problem, one obtains the decision function

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

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 a hyperplane.
For our experiments, we used the one-class SVM implementation LIBSVM 2.6 [16]. LIBSVM is a program, which provides several SVM algorithms for classification and regression, including the one-class SVM implementation proposed by Schölkopf et al. [8]. The default kernel (radial basis function) and the default values of the parameters for the one-class SVM are used.

6 Classification Results and Comparative Study

We wished to explore the effectiveness of all of these approaches on network intrusion detection problems. For our experiments we made use of the dataset from taken from KDD Cup 1999 [17]. 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 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 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 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 1 characterizes a smurf DoS attack which uses spoofed broadcast icmp messages to flood a target system. In contrast, example 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 have 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 on to a natural number, i.e. icmp $\rightarrow 0$, tcp $\rightarrow 1$, udp $\rightarrow 2$, 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 1. The detection rate and false alarm rate is calculated as follows :

$$\text{detection rate} = \frac{\text{anomalous sample correctly classified}}{\text{total anomalous samples}} = \frac{\text{TP}}{\text{TP+FN}}$$

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

The abbreviations *TP,FP,TN,FN* are used in the ROC[7] [18] analysis to evaluate the performance of classification algorithms. Given a classifier and a sample, there are four different outcomes. If the sample is anomalous and it is classified as anomalous, it is counted as a *true positive* (TP); if it is classified as normal, it is counted as a *false negative* (FN). If the sample is normal and it is classified as normal, it is counted as a *true negative* (TN); if it is classified as anomalous, it is counted as a *false positive* (FP).

As the real-valued negative selection is the only method which has a random behavior[8] 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 [6] ($MSC = 99.99\,\%$, $T_{max} = 1000$, $c_0 = 99\,\%$). 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 performed to find an effective self-radius. The radius lengths shown in table 1 resulted in the best classification performance. These radius lengths are also used for the positive selection algorithm.

[7] Received Operating Characteristic
[8] generates detectors randomly

Table 1. 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$^{\nu=0.005}$	99.78	0.03	0.05	0.02	55.70	1.56
ocSVM$^{\nu=0.01}$	99.82	0.02	0.99	0.02	103.40	1.50
ocSVM$^{\nu=0.05}$	99.87	0.02	4.95	0.03	491.15	1.27
Parzen-Window$^{u=0.005}$	99.93	0.02	0.00	0.00	—	—
Parzen-Window$^{u=0.01}$	99.93	0.02	0.00	0.00	—	—
Parzen-Window$^{u=0.05}$	99.93	0.02	0.00	0.00	—	—

6.1 Discussion

In table 1 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 hight 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[9], 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

[9] exponential operation and several arithmetic operations

statistical techniques and to the Self-Detector method presented in this paper. It has a very low detection rate and a very high standard deviation — the standard deviation is far higher than the mean. 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.

7 Conclusion

In this paper we have briefly introduced the anomaly detection problem and have described two statistical anomaly detection approaches — the Parzen-Window and one-class SVM technique. It has been observed that immune system performs an anomaly detection, in part, through a process negative selection. This process eliminates self reactive lymphocytes and also ensures that all possible (including unseen) antigens are recognizable. This negative selection process motivated computer scientists to develop immune inspired algorithms which work in a similar way. On such algorithm, the real-valued negative selection algorithm, which employs variable-sized detectors, is such an anomaly detection approach. We have investigated the termination behavior of the real-valued negative selection algorithm with variable-sized detectors on an artificial data set. The investigations reveal that the algorithm termination behavior is sensitive to several parameters. A high confidence of the estimated detector coverage is necessary for obtaining a steady space coverage, but consequently increases the time complexity for the generation of detectors significantly. We then explored the performance of the algorithm on a high-dimensional data set for anomaly detection, and compared it to the real-valued positive selection and to two statistical anomaly detection techniques. The classification results revealed that the real-valued positive selection outperformed the other classification methods for this data set experiment. However, real-valued positive selection is limited due to the high complexity involved. The Parzen-Window method, likewise achieved a high classification performance, but has same complexity problems as the real-valued positive selection. The one-class SVM achieved a good classification performance and has an acceptable runtime complexity. The real-valued negative selection with variable-sized detectors has poor classification performance on the high-dimensional KDD data set.

It is difficult to conclude that the real-valued negative selection is in general not appropriate on high-dimensional data sets. However this work revealed several problems of the V-Detector algorithm which where not mentioned before. Nevertheless it may appear that the negative selection principle would seem to be a technique that is not appropriate for real-world anomaly detection problems [11,19].

References

1. Forrest S., Perelson A.S., Allen L., Cherukuri R.: Self-nonself discrimination in a computer. In: Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press (1994)
2. D'haeseleer, P.: An immunological approach to change detection: Theoretical results. In: Proc. 9th IEEE Computer Security Foundations Workshop. (1996) 18–26
3. Hofmeyr S. A., Forrest S., D'haeseleer P.: An immunological approach to distributed network intrusion detection. In: First International Workshop on the Recent Advances in Intrusion Detection. (1998)
4. González, F., Dasgupta, D., Kozma, R.: Combining negative selection and classification techniques for anomaly detection. In: Congress on Evolutionary Computation, IEEE (2002) 705–710
5. González, F., Dasgupta, D., Niño, L.F.: A randomized real-valued negative selection algorithm. In Timmis, J., Bentley, P.J., Hart, E., eds.: Proceedings of the 2nd International Conference on Artificial Immune Systems (ICARIS). LNCS, Edinburgh, UK, Springer-Verlag (2003) 261–272
6. Ji, Z., Dasgupta, D.: Real-valued negative selection algorithm with variable-sized detectors. In: Genetic and Evolutionary Computation – GECCO-2004, Part I. Volume 3102 of LNCS., Seattle, WA, USA, Springer-Verlag (2004) 287–298
7. Marsland, S.: Novelty detection in learning systems. Neural Computing Surveys **3** (2003)
8. Schölkopf, B., Platt, J.C., Shawe-Taylor, S.T., Smola, A.J., Williamson, W.: Estimating the support of a high-dimensional distribution. Technical Report MSR-TR-99-87, Microsoft Research (MSR) (1999)
9. Müller, K.R., Mika, S., Rätsch, G., Tsuda, K., Schölkopf, B.: An introduction to kernel-based learning algorithms. Transactions on Neural Networks **12** (2001) 181–201
10. Ebner, M., Breunig, H.G., Albert, J.: On the use of negative selection in an artificial immune system. In: GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, New York, Morgan Kaufmann Publishers (2002) 957–964
11. Stibor, T., Mohr, P., Timmis, J., Eckert, C.: Is negative selection appropriate for anomaly detection ? In: Genetic and Evolutionary Computation – GECCO. (to appear) (2005)
12. Duda, R.., Hart, P.E., Stork, D.G.: Pattern Classification. Second edn. Wiley-Interscience (2001)
13. Bishop C.M.: Novelty detection and neural network validation. In: IEE Proceedings: Vision, Image and Signal Processing. Volume 141. (1994) 217–222
14. Silverman B.W.: Density Estimation for Statistics and Data Analysis. Chapman and Hall (1986)
15. Yeung, D.Y., Chow, C.: Parzen-window network intrusion detectors. In: Proc. of the Sixteenth International Conference on Pattern Recognition. (2002) 385–388
16. Chang, C.C., Lin, C.J.: LIBSVM: a Library for Support Vector Machines (http://www.csie.ntu.edu.tw/∼cjlin/papers/libsvm.pdf). (2004)
17. Hettich, S. and Bay, S. D.: KDD Cup 1999 Data (1999) http://kdd.ics.uci.edu.
18. Fawcett, T.: ROC graphs: Notes and practical considerations for data mining researchers. Technical Report HPL-2003-4, Hewlett Packard Laboratories (2003)
19. Stibor, T., Timmis, J., Eckert, C.: On the appropriateness of negative selection defined over hamming shape-space as a network intrusion detection system. In: Proceedings of the 2005 IEEE Congress on Evolutionary Computation. (to appear), Edinburgh, UK, IEEE Press (2005)

Appendix

Algorithm 1: 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. self coverage

output: D = Set of generated V-Detectors

```
1  begin
2      D ⟵ ∅
3      repeat
4          t ⟵ 0
5          T ⟵ 0
6          r ⟵ ∞
7          x ⟵ random point from [0,1]^n
8          foreach d ∈ D do
               // Euclid. distance between detector center c_d and x
               // is lesser than Non-Self radius r_ns of detector d
9              if dist(c_d, x) ≤ r_ns then
                   // point x is covered by a detector
10                 t ⟵ t + 1
11                 if t ≥ 1/(1 − c_0) then
12                     return D
13                 goto 5:
           // find the closest distance to a self element margin
14         foreach s ∈ S do
15             l ⟵ dist(c_s, x)
16             if l − r_s ≤ r then
17                 r ⟵ l − r_s
18         if r > r_s then
               // Add a new detector d to set D
19             D ⟵ D ∪ {d = (x, r)}
20         else
21             T ⟵ T + 1
22         if T > 1/(1 − MSC) then
23             exit
24     until |D| = T_max
25 end
```