Comments on Real-Valued Negative Selection vs. Real-Valued Positive Selection and One-Class SVM

Thomas Stibor and Jonathan Timmis

***Abstract*— Real-valued negative selection (RVNS) is an immune-inspired technique for anomaly detection problems. It has been claimed that this technique is a competitive approach, comparable to statistical anomaly detection approaches such as one-class Support Vector Machine. Moreover, it has been claimed that the complementary approach to RVNS, termed real-valued positive selection, is not a realistic solution. We investigate these claims and show that these claims can not be sufficiently supported.**

I. Introduction

An early (and popular) immune-inspired algorithm for anomaly detection is negative selection. During negative selection in the natural immune system, self-reactive T-Lymphocytes, which carry antibodies on their surface, are eliminated by a controlled death. As a result, only self-tolerant T-Lymphocytes survive the negative selection process and are then released into the blood stream. Roughly speaking, the negative selection is a process in which self-tolerant lymphocytes are generated, thus allowing the immune system to discriminate self proteins from foreign proteins (termed non-self).

In an artificial immune system (AIS), principles and processes of the natural immune system are abstracted and are applied for solving information processing problems, such as anomaly detection problems. More specifically, in real-valued negative selection (RVNS), T-Lymphocytes are abstracted by means of hyperspheres. In the training phase, the hyperspheres (also called detectors) are (randomly) distributed in an unitary hypercube $\mathbb{H} = [0,1]^n$ of dimension n, such that each self element — also represented as a hypersphere — is not covered by any detector. In the testing phase an element $\mathbf{p} \in \mathbb{H}$ is classified as a self element, if $\mathbf{p}$ is not covered by any detector and otherwise as an anomaly (non-self element). In the field of machine learning, this type of hypersphere detection is known as instance-based learning, i.e. it is a threshold nearest neighbor classifier.

Instead of covering $\mathbb{H}$ completely with detectors, Ebner et al. [1] and subsequently Stibor et al. [2], [3] discussed a different hypersphere detection form. More specifically, no T-Lymphocyte detectors exist at all. Instead, only the given self elements are abstracted by hyperspheres and therefore no training phase is required because $\mathbb{H}$ is *not* (randomly) filled with detectors. In the testing phase, $\mathbf{p}$ is classified as a self element if $\mathbf{p}$ is covered by a hypersphere and otherwise as an anomalous element. This type of hypersphere detection is likewise a simple instance-based learning method and is termed real-valued positive selection (RVPS) [2].

Both methods are evidently straightforward instance based learning techniques which utilize a geometric distance function to decide whether $\mathbf{p}$ is a self element or a non-self element. Stibor et al. [2], [4] have demonstrated that RVPS achieves similar or even better classification results to RVNS for low and high-dimensional anomaly detection problems. Whereas, Ji and Dasgupta [5], [6], [7] argue that it is *only* RVNS that is a promising and competitive to statistical anomaly detection methods approach. As a result, some confusion exists in the AIS community, with the prevalent conjecture that RVNS and RVPS are straightforward methods, which are not competitive to well established statistical techniques.

In this paper we discuss, compare and benchmark RVNS, RVPS and one-class Support Vector Machine (SVM). The paper is organized as follows: In section II the anomaly detection problem is outlined to contextualize the work. In section III, RVNS and RVPS are explained. For a comparative study, the classification principle of the one-class SVM is explained in section IV. In section V the learning capability of RVPS is explored in detail. Experiments and results on a high-dimensional digit recognition problem are shown and discussed in section VI.

II. Anomaly Detection

Anomaly detection, also referred to as one-class learning is an in-balanced two-class pattern classification problem. In (supervised) pattern classification, the goal is to find a functional mapping between input data $\mathbf{x} \in \mathbb{R}^n$ to a class label Y so that $Y = f(\mathbf{x})$. 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 $\mathcal{C}_0, \mathcal{C}_1$ and the task is to find a suitable function f with parameters $\mathbf{p}$ such that $f_{\mathbf{p}} : \mathbb{R}^n \rightarrow \{\mathcal{C}_0, \mathcal{C}_1\}$, will correctly classify

Thomas Stibor is with the Department of Computer Science, Darmstadt University of Technology, 64289 Darmstadt, Germany (email: stibor@sec.informatik.tu-darmstadt.de).

Jonathan Timmis is with the Department of Computer Science and Department of Electronics, University of York, Heslington, York, YO10 5DD, United Kingdom (email: jtimmis@cs.york.ac.uk).

unseen samples. More specifically $\mathbf{p}$ is determined by using training data pairs generated i.i.d.[1] according to an unknown probability distribution

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

and f is (usually) chosen *a-priori* (called model selection). When the training data consists *only* of examples from a single class ($\mathbf{x}, y \in \mathcal{C}_0$) or a single class and a strongly under-represented second anomalous class ($\mathbf{x}, y \in \{\mathcal{C}_0, \mathcal{C}_1\}$) and the test data contains examples 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 data from 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. 1).

III. Real-Valued Negative and Positive Selection

The RVNS algorithm operates on a unitary hypercube $\mathbb{H} = [0,1]^n$ of dimension n. A detector $d = (\mathbf{c}_d, r_{ns})$ has a center $\mathbf{c}_d \in \mathbb{H}$ and a non-self recognition radius $r_{ns} \in \mathbb{R}$. Furthermore, every self element[2] $s = (\mathbf{c}_s, r_s)$ has a center and a self radius r_s. The self-radius allows generalization, i.e. classification beyond the training set. An unlabeled element $\mathbf{e} \in \mathbb{H}$ is classified as non-self, if it lies within d and otherwise as self. An element $\mathbf{e}$ lies within a detector $d = (\mathbf{c}_d, r_{ns})$, if the Euclidean distance $dist(\mathbf{c}_d, \mathbf{e}) = \|\mathbf{c}_d - \mathbf{e}\|_2 < r_{ns}$. Generating detectors in the RVNS algorithm (called V-detector) work as follows:

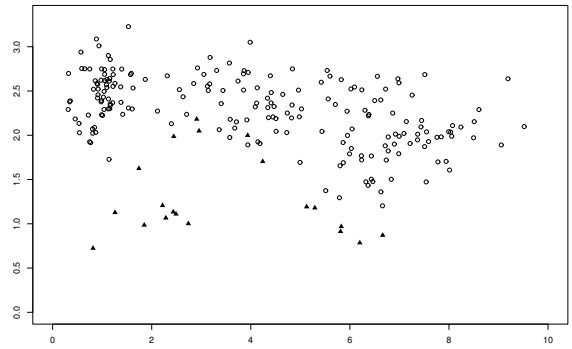
- Input: self elements s with radius r_s, $c_0 \in [0,1]$ and maximum number of detectors $D_{\max}$. Output: detector set $\mathcal{D}$.
- While covered proportion c_0 of $\mathbb{H}$ with detector set $\mathcal{D}$ is not reached and $|\mathcal{D}| < D_{\max}$, do
  - Sample randomly a candidate detector point $\mathbf{c}_d \in \mathbb{H}$.
  - If $\mathbf{c}_d$ is not covered by any s and by any already found detectors in $\mathcal{D}$, then resize radius to $r_{ns} := dist(\mathbf{c}_d, \mathbf{c}_s) - r_s$ where $\mathbf{c}_s$ is the center of the closest self element.
  - If $r_{ns} > r_s$ then put $d = (\mathbf{c}_d, r_{ns})$ into detector set $\mathcal{D}$.

Having generated the detector set $\mathcal{D}$ (training phase), classification of unlabeled elements is performed as described above.

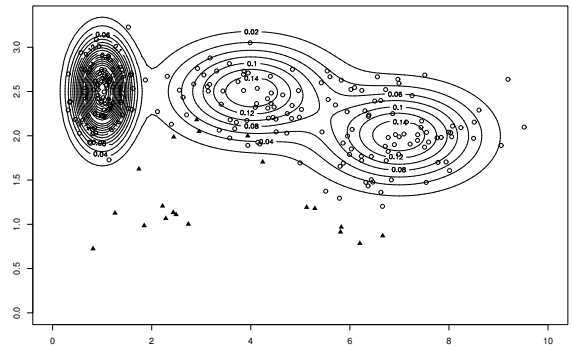
Instead of "filling" $\mathbb{H}$ with detectors with different radii until a coverage of proportion c_0 is reached, one can simply consider each self element as a detector $\widehat{d} = (\mathbf{c}_d, r_s)$. That means no training is required, and classification is performed in an identical manner, except that an unlabeled $\mathbf{e}$ is classified as self if it lies within $\widehat{d}$ and otherwise as non-self. Both straightforward concepts — RVNS and RVPS — are illustrated in figure 2.

[1] Independently drawn and identically distributed.

[2] Points sampled from the "normal" class.



(a) A "typical" anomaly detection problem with two given classes ($\mathcal{C}_0$ = circles and $\mathcal{C}_1$ = triangles). The number of anomalous examples is strongly under-represented (20 examples with class label $\mathcal{C}_1$) compared with 200 examples of normal data (class label $\mathcal{C}_0$).

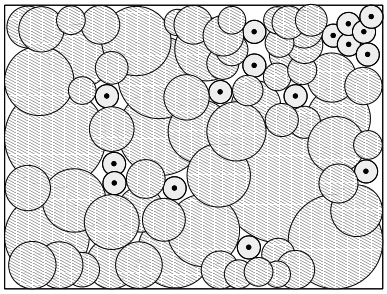


(b) The underlying probability distribution of class $\mathcal{C}_0$ is depicted as a density plot. One can see that the anomalous data is not generated by the probability distribution of class $\mathcal{C}_0$.

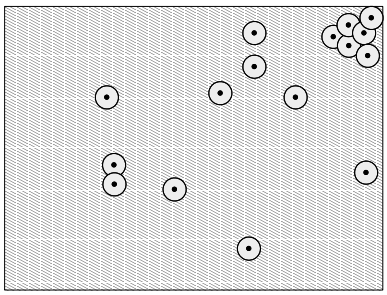
Fig. 1. Anomaly detection problem.

IV. One-Class Support Vector Machine for Anomaly Detection

The Support Vector Machine (SVM) [8], [9] has its foundation in mathematics (computational geometry, optimization and statistics) and is based on the transduction principle [10]. Classical statistical approaches (such as maximum likelihood estimation or density estimation techniques) are based on induction and deduction principle, i.e. first estimating the learning parameters at all (infinite) points of the domain (induction) and then predicting the values at points of interest (deduction). This inference paradigm can be described as a two step movement, namely: from general to particular (deduc-



(a) RVNS principle in a 2-dimensional space. The non-self space which is covered by the detectors is pictured as the shaded area. The self elements are pictured as light grey circles with a black center.



(b) RVPS is simply the RVNS concept, but without filling the 2-dimensional space with detectors. Instead the self elements, i.e. the light grey circles are used as detectors.

Fig. 2. Visual comparison of Real-Valued Negative Selection (RVNS) and Real-Valued Positive Selection (RVPS).

tion) and back from particular to general (induction). Loosely speaking, one has to first solve a difficult intermediate problem, to then solve a relatively simple problem. As a consequence, this two-step movement is an ill-posed problem and suffers under the curse of dimensionality, because one does not have enough learning information to estimate the learning parameters at all (infinite) points of the domain [10]. In the transduction principle, which is implemented by SVM, one estimates the learning parameters at a given finite number of points of interests, and therefore, has enough learning information to overcome these problems. This type of inference is called transductive inference (moving from particular to particular) and is additionally anchored in a robust statistical learning framework.

Based on the computational geometry properties and SVM optimization steps, Schölkopf et al. [11] proposed a one-class SVM, by considering the origin[3] as the only

[3]Instead of assuming that the origin acts as a prior where normal data is not concentrated, Campbell and Bennett [12] extended Schölkopf's et al. problem formulation in terms of finding a surface (a level set $f(\mathbf{x}) = 0$) in input space $\mathcal{X}$ which wraps around the normal data. To be more precise, this can be formulated as $f(\mathbf{x}) = \sum_{i=1}^{l} \alpha_i k(\mathbf{x}, \mathbf{x}_i) + b$ and corresponds in $\mathcal{F}$ to a hyperplane which is pulled onto the mapped datapoints with the restriction that the margin always remains positive or zero.

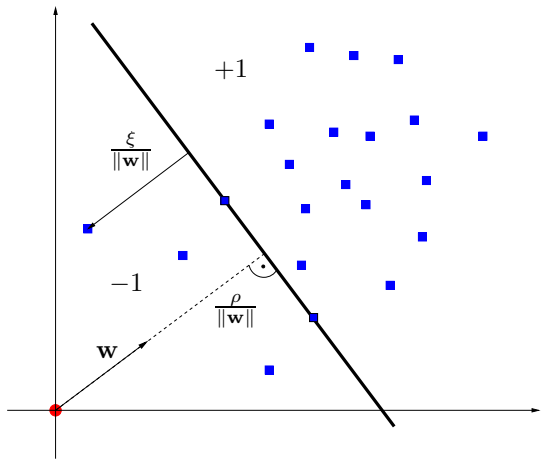


Fig. 3. One-class SVM separates in high-dimensional feature space $\mathcal{F}$, the points from the origin (circled point) with a maximum distance, and allows $\nu \cdot l$ many "outliers" which lie between the origin and the hyperplane, i.e. the -1 side. As a result, most of the points will lie on the $+1$ side of the hyperplane and the outliers on the -1 side. The hyperplane is parametrized by normal vector $\mathbf{w}$, offset ρ and has a distance of $\rho/\|\mathbf{w}\|$ to the origin. Each outlier on the -1 side, has a distance of $\xi_i/\|\mathbf{w}\|$ to the hyperplane.

member of the -1 class. To be more precise, given training examples

$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_l, y_l) \in \mathbb{R}^n \times \{+1\}$$

the input data is mapped via $\Phi(\cdot)$ into a (infinite) high-dimensional feature space $\mathcal{F}$. In $\mathcal{F}$ 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. 3). More specifically, the normal vector of the hyperplane is determined by solving the primal quadratic optimization problem

$$\underset{\mathbf{w},\boldsymbol{\xi},\rho}{\text{minimize}} \quad \tfrac{1}{2}\|\mathbf{w}\|^2 + \tfrac{1}{\nu l}\textstyle\sum_i \xi_i - \rho \qquad (1)$$

$$\text{subject to} \quad \langle \mathbf{w}, \Phi(\mathbf{x}_i)\rangle \geq \rho - \xi_i, \xi_i > 0,\ i = 1, \ldots, l. \qquad (2)$$

Reformulating (1) and (2) to a dual optimization problem in terms of a kernel function $k(\cdot, \cdot)$, one obtains

$$\underset{\boldsymbol{\alpha}\in\mathbb{R}^l}{\text{maximize}} \quad \tfrac{1}{2}\textstyle\sum_{i,j=1}^{l} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \qquad (3)$$

$$\text{subject to } 0 \leq \alpha_i \leq \tfrac{1}{\nu l}, i = 1, \ldots, l \text{ and } \textstyle\sum_{i=1}^{l} \alpha_i = 1. \qquad (4)$$

By solving the dual optimization problem, one obtains the decision function

$$f(\mathbf{x}) \quad = \quad \text{sgn}\left(\sum_{i=1}^{l} \alpha_i k(\mathbf{x}, \mathbf{x}_i) - \rho\right) \qquad (5)$$

which will be positive in region where most of training data is concentrated[4] and negative elsewhere. The value of ρ can be recovered by exploiting the fact, that for α_i

[4]A common way to define anomalies is to assume that anomalies are not concentrated [13], [14] and hence the problem is to construct closed class boundaries of concentrated normal data.

with $0 < \alpha_i \leq 1/(\nu l)$ the corresponding example $\mathbf{x}_i$ is a support vector which satisfies

$$\rho = (\mathbf{w} \cdot \Phi(\mathbf{x}_i)) = \sum_j \alpha_j k(\mathbf{x}_j, \mathbf{x}_i). \quad (6)$$

A. Discussion on one-class SVM

SVMs are anchored in a statistical learning framework [15] and therefore enable a thorough understanding of the generalization capability of the learning algorithm. In contrast, biological inspired approaches such as RVNS or (in the early days) neural networks closely mimic processes of the immune system or the brain. However as mentioned by Vapnik [10], techniques and algorithms which are based on solid mathematical backgrounds are (vastly) superior to techniques and algorithms which copy too naively biological processes:

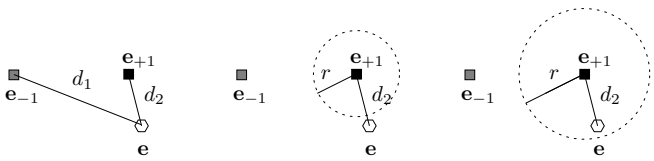
> *"Of course it is very interesting to know how humans can learn. However, this is not necessarily the best way for creating an artificial learning machine. It has been noted that the study of birds flying was not very useful for constructing the airplane."*

The main attractiveness and superiority of one-class SVM — when comparing to RVNS and RVPS — is the capability to deal with high-dimensional classification problems without suffering the curse of dimensionality, foundations in statistical learning theory, and feasible algorithmic complexity. Solving optimization problems (3) and (4), i.e. finding the optimal separating hyperplane with maximal margin is directly linked to minimizing the generalization error. For the one-class SVM Schölkopf et al. [11] proved a boundary of the generalization error, i.e. the expected misclassification error for unseen data. More specifically, they showed that new samples generated according to a probability distribution P will lie predominately in the $+1$ region under the assumption that training examples are generated by P. Such important results are missing for RVNS (and RVPS).

B. Discussion on RVNS and one-class SVM

RVNS is a straightforward anomaly detection technique when comparing to one-class SVM. Important theoretical results for RVNS (and RVPS) as provided by Schölkopf et al. [11] for the one-class SVM and for the SVM in general by Vapnik [15] are currently missing. Instead, authors dealing with RVNS compared the performance of negative selection algorithm version 1 to negative selection algorithm version n, rather than looking for theoretical results. Furthermore, the result on the runtime complexity of RVNS as shown in [16] is incorrect, because they do not consider any "randomness" in their algorithmic investigation (see [3] for more details).

Moreover there seems to exist some confusion regarding the precise estimation of the covered non-self space in RVNS. In [17] authors proposed a Monte Carlo integration to estimate the covered non-self space [17] and subsequently proposed a "naive" integration technique [18], [16] which is imprecise. For the sake of completeness, we have to mention that a variant of the Monte Carlo integration was *rediscovered* again [19].



(a) The 1-NN algorithm assigns $\mathbf{e}$ to class label $+1$, because $d_1 > d_2$, i.e. $\mathbf{e}_{+1}$ is the closest neighbor.

(b) The threshold 1-NN algorithm assigns $\mathbf{e}$ to class label -1, because the threshold radius $r < d_2$.

(c) The threshold radius $r > d_2$ and therefore $\mathbf{e}$ is assign to class label $+1$.

Fig. 4. Illustrations of class assignments of 1-NN and threshold 1-NN algorithm (aka RVPS).

C. Comments on RVPS

We would like to emphasize in this section some misunderstanding on RVPS. Stibor et al. [2] proposed the straightforward RVPS for a comparative study to RVNS and *not* as a state of the art classification technique which is comparable to sound and robust methods from the field of (statistical) machine learning. It is interesting to note, that the authors who proposed RVNS *never* compared the method to RVPS [18], [16], [20], [19], [6], [5]. Moreover we would like to clarify here that using hyperspheres as classifier shapes is not a new concept. This method is known as a "hypersphere classifier" and was proposed more than 40 years ago [21], [22]. Another hypersphere learning approach — with dynamically growing, shrinking and pruning hyperspheres — is proposed by Batchelor [23] and subsequently as a special case of a neural network known as Restricted Coulomb Energy classifier [24].

V. RVPS is a Threshold Nearest Neighbor Classifier

The nearest neighbor (NN) decision rule classifies samples based on the geometrical closeness to the training examples. Given an unlabeled element $\mathbf{e} \in \mathbb{R}^d$, $d \in \mathbb{N}$, the 1-NN algorithm search for the closest training example $\mathbf{e}_c$ and assigns $\mathbf{e}$ the same class label as $\mathbf{e}_c$ (see Fig.4(a)).

It is easily comprehensible that RVPS is a threshold 1-NN classifier for a two-class classification problem. A threshold 1-NN classifier differs from a 1-NN classifier in terms of a (threshold) radius. Instead of searching only for the closest training example, the threshold 1-NN classifier searches for the closest *normal class* training example $\mathbf{e}_c$ and includes a threshold radius r. If the distance between sample $\mathbf{e}$ and example $\mathbf{e}_c$ is smaller than r, $\mathbf{e}$ gets the same class label as $\mathbf{e}_c$, otherwise it gets the opposite class label (see Figs. 4(b) and 4(c)). It is clear that r regulates the overfitting/underfitting behavior of the threshold 1-NN algorithm, comparable

to parameter k in the k-NN algorithm[5]. A "large" r (k, respectively) encloses more surrounding elements and reduces the effect of noise, whereas a "small" r results in the opposite effect.

It is interesting to note that RVPS can be straightforwardly modified[6] in a non-parametric density estimation technique, namely in a Parzen window estimator. This is demonstrated in the following section.

A. From RVPS to Parzen Window

Non-parametric density estimation is a method to estimate an unknown probability density $p(\cdot)$. Given N examples $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ drawn independently from $p(\cdot)$, the aim is to find an estimator $\widehat{p}(\cdot)$ which approximates $p(\cdot)$. Parzen window (also called kernel estimator) is defined as

$$\widehat{p}(\cdot) \quad = \quad \frac{1}{Nh} \sum_{i=1}^{N} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right) \tag{7}$$

and is one feasible approach to estimate $\widehat{p}(\cdot)$. The window width h controls the *smoothness*, i.e. the influence of the surrounding points $\mathbf{x}_i$, whereas the kernel function $K(\cdot)$ determines the *shape*. A kernel function must satisfy the condition

$$\int_{-\infty}^{\infty} K(\mathbf{x})\, d\mathbf{x} = 1. \tag{8}$$

The Epanechnikov kernel function (9) is optimal in terms of minimizing the mean integrated square error[7].

$$K(\mathbf{x}) = \begin{cases} \frac{1}{2}\ B_d^{-1}\ (d+2)\ (1 - \|\mathbf{x}\|^2) & \text{if } \|\mathbf{x}\| < 1 \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

where $B_d = \pi^{d/2}/\Gamma(d/2+1)$ is the volume of an unit hypersphere in dimension d. Loosely speaking, one fixed h and checks of how many samples fall in the hypersphere.

It should be stressed that the crucial parameter in Parzen window is not the choice of the kernel function but rather the window width h. The choice of h implies a trade-off between the bias and the variance. This fact also holds for SVM. Ji and Dasgupta [6] misinterpreted mathematical properties of SVM and naively conclude that the biggest disadvantage of SVM is the proper choice of the kernel function. The choice of the kernel is indeed an important parameter, overfitting/underfitting effects are however controlled by the kernel parameters and the slack variables. Moreover, manifold kernels e.g. for structured data like strings, trees, etc. enable the capturing of the proper learning domain.

RVPS and any non-parametric density estimation techniques indeed are suffering under high runtime complexities to classify test samples. Both methods have no training phase, however classification of test samples is computationally expensive, because one has to iterate through all[8] training examples to classify a test sample.

B. (Self) Element Reduction in RVPS with k-Means

In this section we demonstrate how the k-means clustering technique can be applied as a data reduction technique. More specifically, we demonstrate that Ji's and Dasgupta's argument that *RVPS is not a realistic solution when the number of self samples is larger than the number of detectors*, can not be supported.

Clustering is an unsupervised classification method, i.e. unlabelled data is partitioned into subsets (clusters), according to a similarity measure. That means, "similar" data is grouped into the same cluster. In k-means the grouping is performed as follows: Given input set $\{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_N\} \in \mathbb{R}^n$.

- Choose number of cluster k and initialize[9] randomly the cluster centers $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_k$.
- Assign each unlabeled point $\mathbf{p}_i$ to the nearest cluster center.
- Recompute the new cluster centers.
- Repeat these steps until $\mathbf{m}_i$ converge.

Usually k is chosen very small compared to N. In our case, k determines the data reduction ratio, i.e. $0 < k/N \leq 1$.

VI. Experiments

Handwritten digit recognition is a challenging problem in the field of machine learning due to the high-dimensional nature of the problem. In our experiments we benchmarked RVNS, RVPS and one-class SVM on the MNIST database [25]. The MNIST database contains of handwritten digits (see Fig. 5) which are size-normalized and centered in a 28×28 fixed-size image, the problem domain is hence of dimensionality 784.

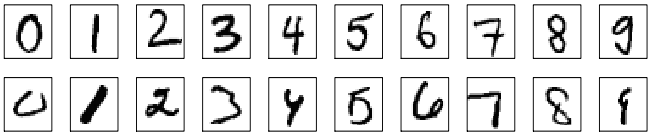


Fig. 5. Handwritten digits $(0, 1, \ldots, 9)$ from the MNIST database [25]. Each digit is size-normalized and centered in a 28×28 fixed-size image. The MNIST database contains noiseless digits (upper row) and also noisy digits (lower row).

The database consists of 60000 training examples and 10000 testing samples which are partitioned in digit sets 0 to 9 (see table I).

Due to the high runtime complexity of RVPS, it was not feasible to perform the experiments on the complete training and testing set in reasonable amount of time[10].

[5]The k-NN algorithm search for the k closest neighbors. The value of k is (usually) an odd number.

[6]This is not a surprise, because the k-NN approach can be applied as a density estimator.

[7]The mean integrated square error $\mathrm{E} \int \{\widehat{p}(\mathbf{x}) - p(\mathbf{x})\}^2 dx$ is measurement for quantifying the discrepancy of the density estimator $\widehat{p}(\cdot)$ from the true density $p(\cdot)$.

[8]For RVPS in the worst-case.

[9]Or sample randomly k centers from the input set.

[10]The experiments are performed on an Intel Xeon 3.2 GHz Dual-CPU machine with 4 GB RAM and 750 GB hard-disk.

Number of examples in each digit set 0 to 9 of training set

0	1	2	3	4	5	6	7	8	9
5923	6742	5958	6131	5842	5421	5918	6265	5851	5949

Number of samples in each digit set 0 to 9 of testing set

0	1	2	3	4	5	6	7	8	9
980	1135	1032	1010	982	892	958	1028	974	1009

TABLE I
NUMBER OF DIGITS IN TRAINING AND TESTING SET.

We reduced the training set by means of the k-means clustering method to 10% of the original set size. Moreover to evaluate the reduction quality of k-means, the original training set is reduced into a second training set by randomly sampling (without replacement) 10% of the original size. As a result, both reduced training sets contain of 6000 examples. The testing set is also reduced to the size of 10% of the original testing set by randomly sampling (without replacement) from the testing set. To obtain comparable results, each set is normalized with min-max normalization to $[0,1]^{784}$.

Each classifier is trained with examples from only one digit set and then validated on the whole testing set, i.e. testing digit sets 0 to 9. This training and validation is performed separately for each digit training set. The classification results of RVPS and one-class SVM are shown in tables II, III and IV. For the RVNS, it was not possible to obtain any results. This is discussed in the subsequent section in detail.

A. Failed RVNS Experiments

For the sake of comparability we used the RVNS implementation (termed V-detector) provided by Ji [26]. The RVNS experiments are performed with different parameter combinations of $c_0 = \{0.8, 0.9, 0.99\}$ and $r_s = \{0.1, 0.5, 1, 2.5, 5, 7.5\}$. Moreover Ji proposed two (similar) versions of RVNS: the variable-sized detectors version [16] and the boundary-aware version [20] — the experiments are performed with both RVNS versions. For the sake of correctness, the detector coverage is estimated with the hypothesis testing approach proposed in [19], rather than with the "naive" and imprecise integration technique [18], [16].

To summarize the experimental results, it was not feasible to obtain any classification results because the V-detector algorithm does not terminate. More precisely, it is infeasible to cover the high-dimensional hypercube $[0,1]^{784}$ with a proper[11] set of hyperspheres when applying the V-detector algorithm. In the worst case no proper set of hyperspheres exist at all, or, one has to sample the space an exponential number of times to find the proper set of hyperspheres. As explained in [4], the volume in high-dimensional hypercubes is concentrated in large corners, which themselves become very long "spikes". In hyperspheres however, the volume is concentrated at the surface, rather than inside. As a consequence, one cannot efficiently "fill" high-dimensional hypercubes with a proper set of hyperspheres without covering the distributed self elements — for more details see [4].

To be sure that the RVNS experiments are not biased by the memory consumptive and (slow) Java implementation [26], the training and testing set for RVNS is reduced to a toy-problem size. The toy-training set contains only 60 examples of digit 9 and the toy-testing set, 51 samples of all digits — both toy sets are randomly sampled (without replacement) from the original sets. For these toy sets, it is also not feasible to obtain any classification results.

These results clearly revealed that *RVNS is not a realistic approach* for real-world anomaly detection problems.

B. RVPS and one-class SVM Classification Results

The RVPS classification results show higher detection rates and lower false alarm rates than the one-class SVM classification rates (see tables II,III and IV). Moreover, one can see that the k-means reduced training set induces a lower false alarm rate, but also a slightly lower detection rate. This is not a surprise because the training set, which is reduced by k-means contains less noise than the original training set.

Although the training and testing set is reduced to 10% of the original size, RVPS has still a runtime complexity which is high for such real-world classification problems. One RVPS classification run (one digit only and one radius value) consumed ca. 2 hours of computation on the 3.2 GHz SMP machine. Whereas a complete one-class SVM classifcation run, i.e. the classification of all digits and for all parameter values of ν, consumed less than 5 minutes of CPU computation.

One possible approach to reduce further the computational complexity of RVPS is the usage of efficient algorithms to determine the nearest neighbor elements. Such algorithms approximate efficiently the nearest neighbors, rather than to determine the exact nearest neighbors (see [27] for more details).

VII. Conclusion

We discussed, compared and benchmarked RVNS, RVPS and one-class SVM. RVPS was identified as a threshold 1-NN classifier which can be modified straightforwardly in Parzen Window estimator. However, RVPS suffers under a high runtime complexity and therefore has limited applicability, for real-world classification problems, i.e. classification problems with a large number of training examples.

One feasible approach to overcome this limitation is the reduction of the training examples e.g. with randomly sampling a subset of the original training set or with

[11] A reasonable number of hyperspheres with large space coverage.

reduction techniques as k-means clustering. The one-class SVM is not affected by such a high runtime complexity problem as the final decision function depends only on the support vectors, rather than on all training examples. Moreover, the one-class SVM is capable of dealing with high-dimensional classification problems without suffering under the the curse of dimensionality. In contrast, the RNVS is limited on low-dimensional classification problems. It was therefore not a great surprise that RVNS failed on the high-dimensional digit recognition experiment because previous theoretical investigations showed that RVNS suffer under the curse of dimensionality, whereas RVPS and one-class SVM do not present such a problem. Hence, arguments stated by Ji and Dasgupta, with regard to RVPS and one-class SVM and the superiority of RNVS are neither justifiable nor supportable.

We conclude this paper with the insight formulated by Vladimir Vapnik [10] "*Nothing is more practical than a good theory*".

Acknowledgment

Thomas Stibor thanks Erin Gardner and Dawn Yackzan for their valuable suggestions and comments.

References

[1] Ebner, M., Breunig, H.G., Albert, J.: On the use of negative selection in an artificial immune system. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2002, Morgan Kaufmann Publishers (2002) 957–964

[2] Stibor, T., Mohr, P.H., Timmis, J., Eckert, C.: Is negative selection appropriate for anomaly detection ? In: Proceedings of Genetic and Evolutionary Computation Conference – GECCO-2005, ACM Press (2005) 321–328

[3] Stibor, T., Timmis, J., Eckert, C.: 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., Springer-Verlag (2005) 262–275

[4] Stibor, T., Timmis, J., Eckert, C.: On the use of hyperspheres in artificial immune systems as antibody recognition regions. In: Proceedings of 5th International Conference on Artificial Immune Systems. Lecture Notes in Computer Science, Springer-Verlag (2006) 215–228

[5] Ji, Z., Dasgupta, D.: Applicability issues of the real-valued negative selection algorithms. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO), ACM Press (2006) 111–118

[6] Ji, Z., Dasgupta, D., Yang, Z., Teng, H.: Analysis of dental images using artificial immune systems. In: Proceedings of Congress On Evolutionary Computation (CEC), IEEE Press (2006) 528–535

[7] Dasgupta, D.: Advances in artificial immune systems. IEEE Computational Intelligence Magazine **1** (2006) 40–49

[8] Boser, B.E., Guyon, I.M., Vapnik, V.: A training algorithm for optimal margin classifiers. In: Proceedings of the fifth annual workshop on Computational learning theory (COLT), ACM Press (1992) 144–152

[9] Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning **20** (1995) 273–297

[10] Vapnik, V.: The Nature of Statistical Learning Theory. Second edn. Springer-Verlag (1999)

[11] Schölkopf, B., Platt, J.C., Shawe-Taylor, Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. Neural Computation **13** (2001) 1443–1471

[12] Campbell, C., Bennett, K.P.: A linear programming approach to novelty detection. In: Advances in Neural Information Processing Systems 13, MIT Press (2001) 395–401

[13] Schölkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press (2001)

[14] Steinwart, I., Hush, D., Scovel, C.: A classification framework for anomaly detection. Journal of Machine Learning Research **6** (2005) 211–232

[15] Vapnik, V.: Statistical Learning Theory. Wiley-Interscience (1998)

[16] 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 Lecture Notes in Computer Science., Seattle, WA, USA, Springer-Verlag (2004) 287–298

[17] González, F., Dasgupta, D., Niño, L.F.: 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., Edinburgh, UK, Springer-Verlag (2003) 261–272

[18] Ji, Z., Dasgupta, D.: Augmented negative selection algorithm with variable-coverage detectors. In: Congress on Evolutionary Computation (CEC), IEEE (2004) 1081–1088

[19] Ji, Z., Dasgupta, D.: Estimating the detector coverage in a negative selection algorithm. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO), ACM Press (2005) 281–288

[20] Ji, Z.: A boundary-aware negative selection algorithm. In: Proceedings of the 9th International Conference on Artificial Intelligence and Soft Computing, ACTA Press (2005)

[21] Cooper, P.W.: The hypersphere in pattern recognition. Information and Control **5** (1962) 324–346

[22] Cooper, P.W.: A note on an adaptive hypersphere decision boundary. IEEE Transactions on Electronic Computers **EC-15** (1966) 948–949

[23] Batchelor, B.G.: Practical approach to pattern classification. Springer (1974)

[24] Hudak, M.J.: RCE classifiers: theory and practice. Cybernetics and Systems **23** (1993) 483–515

[25] LeCun, Y.: The MNIST database of handwritten digits (1998) http://yann.lecun.com/exdb/mnist/index.html.

[26] Ji, Z.: V-detector java source code (2006) http://www.zhouji.net/prof/vdetector.html.

[27] Gregory Shakhnarovich, T.D., Indyk, P., eds.: Nearest-Neighbor Methods in Learning and Vision: Theory and Practice. MIT Press (2006)

ν	0	1	2	3	4	5	6	7	8	9	mean	stdev
0.01	0.7739	0.8202	0.2556	0.3769	0.7940	0.2502	0.8865	0.5879	0.5596	0.7034	0.6008	0.2360
	0.0215	0.0272	0.0431	0.0204	0.0108	0.0322	0.0495	0.0280	0.0574	0.0582	0.0348	0.0163
0.05	0.9514	0.9719	0.3597	0.4833	0.8281	0.3186	0.9143	0.7323	0.5618	0.7948	0.6916	0.2440
	0.0752	0.0818	0.0862	0.0612	0.0217	0.0645	0.0792	0.0654	0.0919	0.0679	0.0695	0.0195
0.1	0.9735	0.9887	0.4819	0.6330	0.8546	0.4013	0.9466	0.8241	0.6484	0.8405	0.7592	0.2073
	0.1290	0.1363	0.1465	0.0816	0.0760	0.0967	0.0990	0.1214	0.0919	0.1262	0.1104	0.0244
0.15	0.9878	0.9943	0.5599	0.7206	0.8766	0.4641	0.9677	0.8812	0.6856	0.8639	0.8001	0.1848
	0.1612	0.1818	0.2155	0.1122	0.1413	0.1075	0.1188	0.1869	0.1149	0.1650	0.1505	0.0373

TABLE II

One-class SVM classification results for training and testing set which is randomly sampled from the original sets. The reduced sets have a size of 10% of the orginal sets. The RBF Kernel is used with parameter $\nu = \{0.01, 0.05, 0.1, 0.15\}$. The upper row shows for each ν-value the detection rate, the lower row the false alarm rate. The last column shows the mean and standard deviation value of the detection rate and false alarm rate over all digits.

r	0	1	2	3	4	5	6	7	8	9	mean	stdev
6.2	0.9944	0.9426	0.9264	0.9013	0.7962	0.9272	0.9132	0.7939	0.8378	0.7881	0.8821	0.0726
	0.1720	0.0090	0.4310	0.1836	0.0978	0.2043	0.1089	0.0373	0.2643	0.0485	0.1556	0.1262
6.3	0.9922	0.9359	0.9015	0.8858	0.7797	0.8985	0.9010	0.7749	0.8258	0.7759	0.8671	0.0746
	0.1612	0.0090	0.4224	0.1530	0.0760	0.1935	0.0891	0.0373	0.2068	0.0388	0.1387	0.1212
6.4	0.9900	0.9292	0.8846	0.8558	0.7610	0.8809	0.8954	0.7648	0.8148	0.7458	0.8522	0.0798
	0.1505	0.0090	0.3793	0.1326	0.0434	0.1505	0.0693	0.0373	0.1839	0.0388	0.1194	0.1093
6.5	0.9867	0.9101	0.8585	0.8337	0.7345	0.8621	0.8820	0.7480	0.8017	0.7179	0.8335	0.0847
	0.1075	0.0090	0.2931	0.1326	0.0434	0.1290	0.0594	0.0373	0.1724	0.0291	0.1012	0.086
6.6	0.9801	0.9078	0.8416	0.8037	0.7191	0.8379	0.8687	0.7245	0.7754	0.6989	0.8157	0.0897
	0.0860	0.0090	0.2241	0.1326	0.0434	0.1182	0.0495	0.0373	0.1494	0.0291	0.0878	0.0674
6.7	0.9713	0.8921	0.8144	0.7760	0.7004	0.7949	0.8487	0.7077	0.7458	0.6767	0.7928	0.0925
	0.0645	0.0090	0.1982	0.0918	0.0434	0.0860	0.0495	0.0373	0.1494	0.0291	0.0758	0.0584
6.8	0.9547	0.8764	0.7907	0.7339	0.6850	0.7618	0.8387	0.6920	0.7185	0.6610	0.7712	0.0941
	0.0537	0.0090	0.1551	0.0918	0.0	0.0537	0.0297	0.0280	0.1149	0.0291	0.0565	0.0495

TABLE III

RVPS classification results for different hypersphere radii $r = \{6.2, 6.3, \ldots, 6.8\}$. The training and testing set is randomly sampled from the original sets and have a size of 10% of the orginal sets.

r	0	1	2	3	4	5	6	7	8	9	mean	stdev
6.2	0.9481	0.9067	0.7714	0.7006	0.6883	0.7475	0.8576	0.7133	0.6834	0.6499	0.7666	0.1027
	0.0107	0.0090	0.0775	0.0510	0.0108	0.0107	0.0198	0.0186	0.0574	0.0097	0.0275	0.0249
6.3	0.9283	0.8943	0.7500	0.6651	0.6696	0.7188	0.8320	0.7054	0.6374	0.6231	0.7424	0.1075
	0.0107	0.0090	0.0603	0.0510	0.0108	0.0107	0.0198	0.0093	0.0459	0.0097	0.0237	0.0203
6.4	0.9040	0.8786	0.7115	0.6363	0.6552	0.6802	0.8175	0.6875	0.6144	0.6020	0.7187	0.1092
	0.0	0.0090	0.0517	0.0408	0.0108	0.0107	0.0198	0.0093	0.0459	0.0	0.0198	0.0191
6.5	0.8787	0.8651	0.6911	0.6008	0.6321	0.6515	0.7997	0.6696	0.5717	0.5808	0.6941	0.1141
	0.0	0.0090	0.0344	0.0306	0.0108	0.0107	0.0198	0.0093	0.0344	0.0	0.0159	0.0131
6.6	0.8434	0.8494	0.6640	0.5620	0.6134	0.6130	0.7697	0.6550	0.5421	0.5562	0.6668	0.1153
	0.0	0.0090	0.0258	0.0204	0.0108	0.0107	0.0099	0.0	0.0344	0.0	0.0121	0.0116
6.7	0.8092	0.8348	0.6300	0.5388	0.5903	0.5678	0.7497	0.6382	0.5049	0.5317	0.6395	0.1186
	0.0	0.0090	0.0172	0.0204	0.0	0.0107	0.0	0.0	0.0344	0.0	0.0091	0.0117
6.7	0.7706	0.8089	0.5938	0.5033	0.5704	0.5292	0.7208	0.6103	0.4654	0.5117	0.6084	0.1192
	0.0	0.0090	0.0172	0.0204	0.0	0.0107	0.0	0.0	0.0344	0.0	0.0091	0.0117

TABLE IV

RVPS classification results for different hypersphere radii $r = \{6.2, 6.3, \ldots, 6.8\}$. The training is reduced by means of k-means clustering to a size of 10% of the orginal training set. The testing set is randomly sampled from the original testing set and has a size of 10% of the original testing set.