Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2016) 24: 298 – 312

doi:10.3906/elk-1307-137

Heuristic sample reduction method for support vector data description

Wenzhu SUN*, Jianling QU, Yang CHEN, Yazhou DI, Feng GAO
Naval Aeronautical Engineering Institute, Qingdao Branch, Qingdao, P.R. China

Received: 17.07.2013 • **Accepted/Published Online:** 05.11.2013 • **Final Version:** 01.01.2016

Abstract: Support vector data description (SVDD) has become one of the most promising methods for one-class classification for finding the boundary of the training set. However, SVDD has a time complexity of $O(N^3)$ and a space complexity of $O(N^2)$. When dealing with very large sizes of training sets, e.g., a training set of the aeroengine gas path parameters with the size of $N > 10^6$ sampled from several months of flight data, SVDD fails. To solve this problem, a method called heuristic sample reduction (HSR) is proposed for obtaining a reduced training set that is manageable for SVDD. HSR maintains the classification accuracy of SVDD by building the reduced training set heuristically with the samples selected from the original. For demonstration, several artificial datasets and real-world datasets are used in the experiments. In addition, a practical example of the training set of the aeroengine gas path parameters is also used to compare the performance of SVDD based on the proposed HSR with conventional SVDD and other improved methods. The experimental results are very encouraging.

Key words: Support vector data description, heuristic sample reduction, novelty detection, one-class classification

1. Introduction

Support vector data description (SVDD), inspired by the support vector machine (SVM) by Vapnik, was proposed by Tax and Duin in 1999 [1,2]. It is a promising one-class classifier that can distinguish outliers from targets effectively by shaping a smooth decision boundary around the training set. SVDD shows a promising performance in the field of outlier detection, also called novelty detection [3–7] and pattern denoising [8]. Additionally, SVDD can also be used as a clustering method [9,10]. Furthermore, SVDD was recently extended to the two-class classification problem [11], and a competitive generalization performance was reported.

However, SVDD has the same shortcoming as SVM for founding a kernel matrix with N^2 elements and solving a quadratic programming (QP) problem with $O(N^3)$ time complexity, where N is the size of training set. Obviously, when N is large, the SVDD training becomes prohibitive. Thus, it has become an important research topic to speed up SVDD training, and many researchers have focused on solving the problem.

The least squares one-class SVM (LS-OCSVM) [12] proposed by Choi reduced the time complexity from $O(N^3)$ to $O(N^2)$ by replacing the inequality constraints with equality constraints in the QP. Despite the fact that LS-OCSVM has a fast training speed, its testing speed is slow because it gets a number of support vectors when solving QP.

In order to allow the QP to apply better to larger datasets, Platt and Schölkopf et al. suggested the use of the sequential minimal optimization (SMO) algorithm [13,14]. SMO breaks the large QP problem into

*Correspondence: sunwenzhulm@gmail.com

a series of the smallest possible QP problems. These small QP problems are solved analytically and need few operations. Both time complexity and space complexity are linear to the training set size, but it is hard for the SMO method to obtain the global optimal solution because its solutions are determined by approximately optimal Karush–Kuhn–Tucker (KKT) conditions.

Another possible way is to use chunks of samples. A chunk is a predefined small number of samples, which is much less than the total number of samples in the whole training set [15]. Each chunk of samples is condensed to support vectors by using SVDDs, and another SVDD is applied to these support vectors again to obtain the final decision boundary. Although this is a feasible way of avoiding memory capacity and cost problems, it is still very time-consuming for huge training sets.

A more radical possibility to improve the training speed of SVDD is by reducing the training samples. The first sample reduction algorithm was proposed by Hart in the condensed nearest neighbor (CNN) rule [16], which finds a subset X_{tr}^{sub} of the training set X_{tr} such that every member of X_{tr} is closer to a member of X_{tr}^{sub} of the same class than to a member of X_{tr}^{sub} of a different class. Inspired by the CNN rule, a series of methods was proposed [17–22]. Nearly all of these methods decide to remove or retain samples in a training set based on the 2 classes' (or more than 2 classes') classification results that could not be used in one-class classification. Recently, Angiulli proposed the CPDD method [23]. CPDD selects a minimum consistent subset from the original training set for nearest neighbor one-class classification, which can also be used for condensing a large-sized training set for other classifiers. However, in the context of SVDD, there is no guarantee that CPDD methods will retain all possible support vectors (as demonstrated experimentally in Section 5), which would directly affect the construction of a decision boundary.

In this study, a sample reduction method referred to as heuristic sample reduction (HSR), which heuristically reduces the training set in terms of the characteristics of SVDD, is proposed. In SVDD, we notice that the decision boundary is only decided by a small proportion of training samples called support vectors, which are always distributed in the boundary area. Therefore, HSR devotes itself to finding a subset of a training set that includes all of the potential support vectors, and it ensures that the potential support vectors become support vectors in SVDD training. Afterwards, an accurate SVDD decision boundary can be rapidly obtained by the subset.

The remainder of the paper is organized as follows: Section 2 introduces the basics of SVDD; Section 3 introduces the proposed HSR method in detail; Section 4 presents the efficiency analysis of HSR; Section 5 evaluates the proposed method through experiments with an artificial dataset, real-world datasets, and flight data recorded by a flight data recorder (FDR), respectively; and Section 6 provides the conclusions.

2. SVDD

Given a training set $X_{tr} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$, SVDD identifies a hypersphere with minimum volume that captures the given training set. The hypersphere volume is characterized with its center $\mathbf{c}$ and radius r. Minimization of the hypersphere volume is achieved by minimizing r^2, which represents structural errors.

$$\min \ r^2, \tag{1}$$

$$\text{s.t.} \ \ \|\mathbf{x}_i - \mathbf{c}\|^2 \leq r^2 \ \ \forall i. \tag{2}$$

The above constraints do not allow any data to fall outside of the hypersphere. In order to tolerate the potential singular samples within the training set, a penalty cost function is introduced as follows:

$$\min \; r^2 \; + \; C \sum_i \xi_i, \tag{3}$$

$$\text{s.t.} \;\; \|\mathbf{x}_i - \mathbf{c}\|^2 \; \leq \; r^2 \; + \; \xi_i \;\; \xi_i \; \geq \; 0 \;\; \forall i, \tag{4}$$

where C is the penalty coefficient for each outlier (also referred to as the regularization parameter) and ξ_i is the distance between the ith sample and the hypersphere. This minimization problem can be solved efficiently by introducing Lagrange multipliers for the constraints. The Lagrange function is defined as follows:

$$L(r, \mathbf{c}, \xi, \gamma) \; = \; r^2 \; + \; C \sum_i \xi_i \; - \; \sum_i \alpha_i (r^2 + \xi_i \; - \; \|\mathbf{x}_i \; - \; \mathbf{c}\|^2) \; - \; \sum_i \gamma_i \xi_i, \tag{5}$$

where γ_i and α_i are the Lagrange multipliers, $\gamma_i \; \geq \; 0$, $\alpha_i \; \geq \; 0$. Note that for each training sample $\mathbf{x}_i$, the corresponding α_i and γ_i are defined. L has to be minimized with respect to r, $\mathbf{c}$, and ξ and maximized with respect to α and γ.

Taking the derivatives of Eq. (5) with respect to r, $\mathbf{c}$, and ξ, and equating them to zero, the constraints can be obtained as follows:

$$\mathbf{c} \; = \; \sum_i \alpha_i \mathbf{x}_i, \tag{6}$$

$$C \; - \; \alpha_i \; - \; \gamma_i \; = \; 0 \;\; \forall i, \tag{7}$$

$$\sum_i \alpha_i \; = \; 1. \tag{8}$$

Given that $\gamma_i \; \geq \; 0$ and $\alpha_i \; \geq \; 0$, the constraint from Eq. (7) can be rewritten as:

$$0 \; \leq \; \alpha_i \; \leq \; C \;\; \forall i. \tag{9}$$

Substituting Eqs. (6), (8), and (9) into Eq. (5), we get:

$$L(\alpha) \; = \; \sum_i \alpha_i (\mathbf{x}_i \; \cdot \; \mathbf{x}_i) \; - \; \sum_{i,j} \alpha_i \alpha_j (\mathbf{x}_i \; \cdot \; \mathbf{x}_j) \tag{10}$$

The minimization of L with the constraints of Eqs. (8) and (9) is a well-known QP problem, and standard algorithms exist to solve this. Samples corresponding to $\alpha_i \; > \; 0$ will be defined as support vectors. The center of the hypersphere can then be calculated using Eq. (6). The radius r is defined as the distance between the center $\mathbf{c}$ and one of the support vectors.

The hypersphere obtained by the above method is a rigid spherical hypersphere. However, in most cases, the distribution of samples in a feature space is not spherical. A rigid decision boundary cannot fit most sample distributions well. Therefore, the kernel trick is introduced to arrive at a more flexible decision boundary by replacing the inner product in Eq. (10) with a kernel function. There are a lot of kernel types, including the polynomial kernel [24], the sigmoid kernel [25], and the Gaussian kernel. In this paper, only the Gaussian kernel is involved due to its excellent performance [2,26].

A Gaussian kernel function is defined as:

$$K(\mathbf{x}_i, \; \mathbf{x}_j) \; = \; \varphi(\mathbf{x}_i) \; \cdot \; \varphi(\mathbf{x}_j) \; = \; \exp\left(\frac{- \|\mathbf{x}_i \; - \; \mathbf{x}_j\|^2}{\sigma^2} \right) \;\; \forall i, \; j, \tag{11}$$

where σ is a width parameter. Note that:

$$K(\mathbf{x}_i,\ \mathbf{x}_i)\ =\ 1,\ \ \forall i. \tag{12}$$

Thus, the Lagrange function of Eq. (10) can be rewritten as:

$$L(\alpha)\ =\ -\sum_{i,j}\alpha_i\alpha_j K(\mathbf{x}_i\ \cdot\ \mathbf{x}_j). \tag{13}$$

A discrimination function that is used to evaluate whether or not a testing sample $\mathbf{x}$ is accepted by the decision boundary is given by:

$$f_{SVDD}\ =\ sgn(\breve{B}\ -\ B), \tag{14}$$

$$\breve{B}\ =\ \sum_{i}^{N_{SV}}\alpha_i\ \exp\left(\frac{-\left\|\mathbf{x}\ -\ \mathbf{x}_i\right\|^2}{\sigma^2}\right), \tag{15}$$

$$B\ =\ \sum_{i,j=1}^{N_{SV}}\alpha_i\alpha_j K(\mathbf{x}_i,\ \mathbf{x}_j), \tag{16}$$

where N_{SV} denotes the number of support vectors. A testing sample $\mathbf{x}$ will be accepted if $f_{SVDD}\ =\ 1$ and rejected if $f_{SVDD}\ =\ -1$.

3. HSR method

This section describes the HSR method in detail. HSR produces a reduced training set $\tilde{X}_{tr}$ by reducing a portion of the samples in training set X_{tr}. The steps of the HSR method are shown as follows:

Step 1. Normalize X_{tr} and set $\tilde{X}_{tr}$ to empty.

Step 2. The k-means method is applied to X_{tr} for obtaining the cluster centers $\mathbf{c}_j$, $(j\ =\ 1,\ \cdots,\ k)$ of X_{tr}.

Step 3. A distance variance d_i, which denotes the distance from the ith sample to its nearest $\mathbf{c}_j$, is defined by:

$$d_i\ =\ \min\ \left(\left\|\mathbf{x}_i\ -\ \mathbf{c}_j\right\|\right),\ \ \forall j;\ i\ =\ 1,\ \cdots,\ N. \tag{17}$$

Step 4. A sorted training set X_{tr}^{sorted} is built by sorting all of the X_{tr} samples in descending order in terms of d_i.

Step 5. Set the first sample in X_{tr}^{sorted} to highlight sample $\mathbf{x}_H$ and add the $\mathbf{x}_H$ into $\tilde{X}_{tr}$.

Step 6. Remove the samples satisfying $\left\|\mathbf{x}_H,\ \mathbf{x}_i\right\|\ <\ \delta\ \ (\forall i)$ from X_{tr}^{sorted} (including $\mathbf{x}_H$), where δ is the distance threshold set by user.

Step 7. Repeat steps 5 and step 6 until X_{tr}^{sorted} is empty. In the meantime, $\tilde{X}_{tr}$ is built.

An example of a sample reduction algorithm is shown in Figure 1. Figure 1a shows the k-means clustering result with $k\ =\ 5$, where k denotes the number of clusters. The gradient gray colors denote the contours of d_i. Figure 1b shows the sample reduction result with $\delta\ =\ 0.1$.

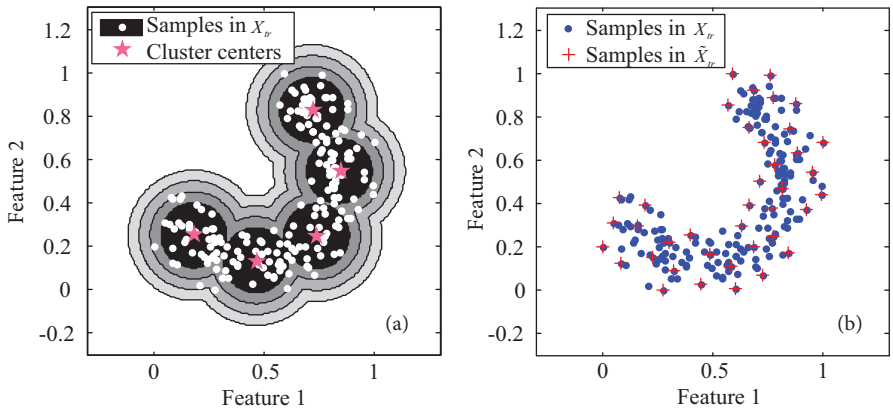

Figure 1. An example of the HSR method using the 2D Banana dataset with 200 samples: (a) k-means clustering result with $k = 5$; (b) result of HSR with $\delta = 0.1$.

In the HSR method, a most critical parameter is the number of clusters k, which needs to be set in Step 2. A lot of studies provide methods to identify the optimal number of clusters in k-means [27–29]. Another more direct method is to identify the k value by expertise, which can obtain similar results as the methods mentioned above.

In the SVDD based on HSR, the time complexity can be expressed as:

$$T_t = O(N) + \sum_{i=1}^{\tilde{N}} O(N_i) + O(\tilde{N}^3), \tag{18}$$

where N_i is the number of samples left in X_{tr}^{sorted} in the ith iteration. In practice, the loop times in the ith iteration can be decreased to a level far less than N_i with the improved algorithm. Because the samples in X_{tr}^{sorted} have been sorted, the ith cycle can be broken ahead of schedule if $d_H - d_j > \delta$ $(j = 1, \cdots, N_i)$, where d_H is the distance variance of $\mathbf{x}_H$. Therefore, the term $\sum_{i=1}^{\tilde{N}} O(N_i)$ becomes much less in practice. $O(N) + \sum_{i=1}^{\tilde{N}} O(N_i)$ is less than $O(\tilde{N}^3)$ in orders of magnitude and can be neglected compared to $O(\tilde{N}^3)$. Thus, the time complexity can be approximately defined as $T_t = O(\tilde{N}^3)$. The space complexity of the proposed method is $T_s = O(\tilde{N}^2)$.

4. Efficiency analysis of HSR

This section explains why HSR performs successfully. In order to speed up the SVDD training, the HSR method cuts off a portion of samples in X_{tr} to form a reduced training set $\tilde{X}_{tr}$. The samples in $\tilde{X}_{tr}$ have the following characteristics:

- The distance between any 2 samples is greater than or equal to δ;
- The distribution of samples in $\tilde{X}_{tr}$ is nearly the same as that of X_{tr}.

With these 2 characteristics, the decision boundary trained by $\tilde{X}_{tr}$ can be similar to the decision boundary obtained by X_{tr}, which can be explained by discussing the characteristics of SVDD, as follows.

For the convenience of elaboration, SVDD falls into 3 types in terms of different σ, namely small σ, intermediate σ, and large σ. A SVDD training experiment of these 3 types with σ ranging from 0.08 to 8

using a 2D Banana dataset is shown in Figure 2, where + denotes the training samples, O denotes the support vectors, and a dotted line denotes the decision boundary.

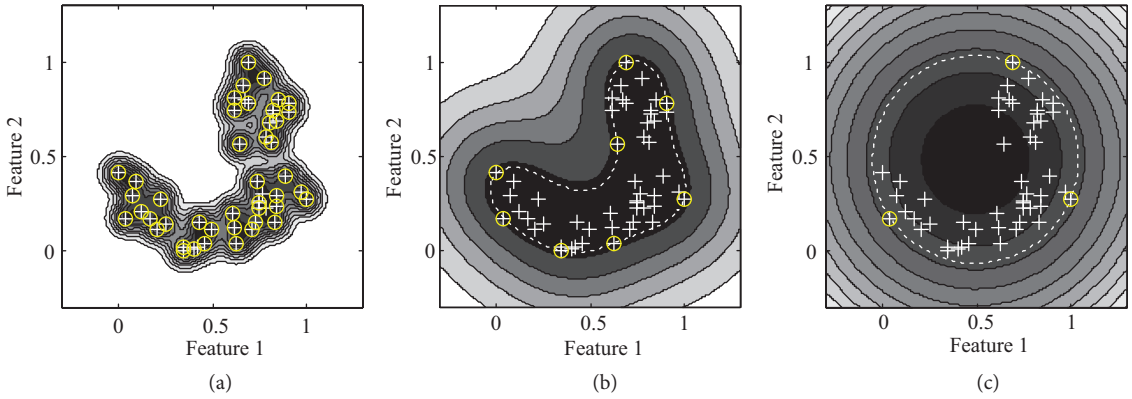

Figure 2. An example of Gaussian kernel SVDD training results with different width parameters σ, where white plus signs denote the training samples, yellow circles denote support vectors, and dotted line denotes the decision boundary. (a) $\sigma = 0.08$; (b) $\sigma = 0.4$; (c) $\sigma = 8$.

For a sufficiently small σ, we have:

$$\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2} \ll 0. \tag{19}$$

Thus, the differences between $K(\mathbf{x}_i, \mathbf{x}_j)$ for different i and j are very small; namely:

$$\sum_{j=1}^{N} \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}\right) \approx \sum_{i=1}^{N} \exp\left(\frac{-\|\mathbf{x}_j - \mathbf{x}_i\|^2}{\sigma^2}\right) (\forall i, j). \tag{20}$$

Taking the constraint of $0 \leq \alpha_i \leq C$ $(\forall i)$ and $\sum_i \alpha_i = 1$ into account, Eq. (13) is minimized when all objects become support vectors with equal $\alpha_i = 1 / N$ $(\forall i)$, as illustrated in Figure 2a. This type of SVDD is identical to a one-class Parzen classifier with a Gaussian window. Each individual sample now supports a small, equally Gaussian window, and the total model is a sum of all of these Gaussian windows. In this case, the decision boundary trained by $\tilde{X}_{tr}$ is nearly identical to that of X_{tr} because the distribution of $\tilde{X}_{tr}$ and X_{tr} is nearly the same.

For intermediate σ values, $\sum_{j=1}^{N} K(\mathbf{x}_i, \mathbf{x}_j)$ is large if $\mathbf{x}_i$ is located in the central area of distribution and $\sum_{j=1}^{N} K(\mathbf{x}_i, \mathbf{x}_j)$ is small if $\mathbf{x}_i$ is located in the margin area of distribution. In the process of Lagrange function minimization, the Lagrange multipliers α_i tend to become 0 if the values of $\sum_{j=1}^{N} K(\mathbf{x}_i, \mathbf{x}_j)$ are large. Only for the smallest $\sum_{j=1}^{N} K(\mathbf{x}_i, \mathbf{x}_j)$does the corresponding α_i become larger than 0, and the samples corresponding to these α_i become support vectors. Thus, these support vectors are always located in the margin area of distribution (see Figure 2b). Eqs. (14) to (16) clearly show that the decision boundary is identical to a weighted one-class Parzen classifier constructed by support vectors. These support vectors are very likely to be selected into $\tilde{X}_{tr}$ because they are distributed in the margin of the training set and they obtain a higher priority than the samples distributed in the center of the training set in HSR. In the training process using $\tilde{X}_{tr}$, these support vectors still become support vectors because $\tilde{X}_{tr}$ has nearly the same distribution as X_{tr}.

For large σ values, $K(\mathbf{x}_i,\ \mathbf{x}_j)$ can be transformed into Taylor series expansions, which can be expressed as:

$$\begin{aligned} K(\mathbf{x}_i,\ \mathbf{x}_j) &= \exp\left(-\left\|\mathbf{x}_i\ -\ \mathbf{x}_j\right\|^2\ /\ \sigma^2\right) \\ &= 1\ -\ \mathbf{x}_i\ /\ \sigma^2\ -\ \mathbf{x}_j\ /\ \sigma^2\ +\ 2(\mathbf{x}_i\ \cdot\ \mathbf{x}_j)\ /\ \sigma^2\ +\ o(1\ /\ \sigma^2), \end{aligned} \tag{21}$$

where $o(1\ /\ \sigma^2)$ is a higher-order infinitesimal than the infinitesimal $1\ /\ \sigma^2$.

Substituting Eq. (21) into Eq. (13), we obtain:

$$L\ =\ -1\ +\ \frac{2}{\sigma^2}\left(\sum_i \alpha_i(\mathbf{x}_i\ \cdot\ \mathbf{x}_i) - \sum_i \alpha_i\alpha_j(\mathbf{x}_i\ \cdot\ \mathbf{x}_j)\right)\ +\ o(1\ /\ \sigma^2). \tag{22}$$

Note that Eqs. (22) and (10) have similar forms; the only differences between them are scaling factor $2\ /\ \sigma^2$, offset -1, and infinitesimal $o(1\ /\ \sigma^2)$. However, these 3 factors do not affect QP solutions. Therefore, the decision boundary of SVDD in a very large σ is equal to the rigid hypersphere SVDD, and only a few training samples distributed in the outermost areas of the training set become support vectors. As in the above case, these support vectors are most likely the same as the support vectors trained by $\tilde{X}_{tr}$.

For a demonstration of the analysis above, an experiment is conducted. In this experiment, both conventional SVDD (C-SVDD) and SVDD based on HSR (H-SVDD) are applied to the same 2D Banana dataset with 200 samples. The training sets (including a reduced training set for H-SVDD), support vectors, and decision boundaries are shown in Figure 3. Figure 3a shows the training result of C-SVDD with $C\ =\ 1$ and $\sigma\ =\ 0.4$. The white dots denote training samples and the dotted line denotes the decision boundary obtained by C-SVDD. The decision boundary encloses all of the training samples with a small volume and smooth surface, which means that the classifier has strong learning and generalization abilities. Figures 3b and 3c show the training results using H-SVDD with $\delta\ =\ 0.05$ and $\delta\ =\ 0.1$, respectively. Parameters C and σ are set to the same values as those of C-SVDD. Samples in $\tilde{X}_{tr}$ are signified with + and the support vectors are signified with small circles. $\tilde{X}_{tr}$ in Figures 3b and 3c include 82 and 39 samples, respectively, but only the support vectors resolved by H-SVDD are exactly the same as the support vectors resolved by C-SVDD, which indicates that the decision boundary obtained by $\tilde{X}_{tr}$ is the same as the decision boundary obtained by X_{tr}. It is worth mentioning that when the training samples are reduced sharply with HSR, the performance of H-SVDD is still adequate. Figure 3d shows the training result of H-SVDD with $\delta\ =\ 0.3$. The size of $\tilde{X}_{tr}$ is just 7, which is far less than that of X_{tr}, and all of these samples in $\tilde{X}_{tr}$ become support vectors in the H-SVDD resolving process. At the same time, the decision boundary generated by these 7 samples is still similar to the decision boundary in Figure 3a.

5. Experiments

In this section, a series of experiments are conducted aiming to assess the performance of the proposed HSR method. First, the measures that are employed during the experiments are described as follows. The false positive rate (FPR) identifies the probability that outliers are accepted wrongly. The true positive rate (TPR) identifies the probability that targets are recognized correctly. The receiver operating characteristic (ROC) curve is the plot of the FPR versus the TPR and can serve to compare the classification accuracy of the one-class classification methods. The more the ROC curves are located in the upper left corner, the better the targets and outliers are separated. To determine the ROC curve of SVDD, the B value in Eq. (14) varies from

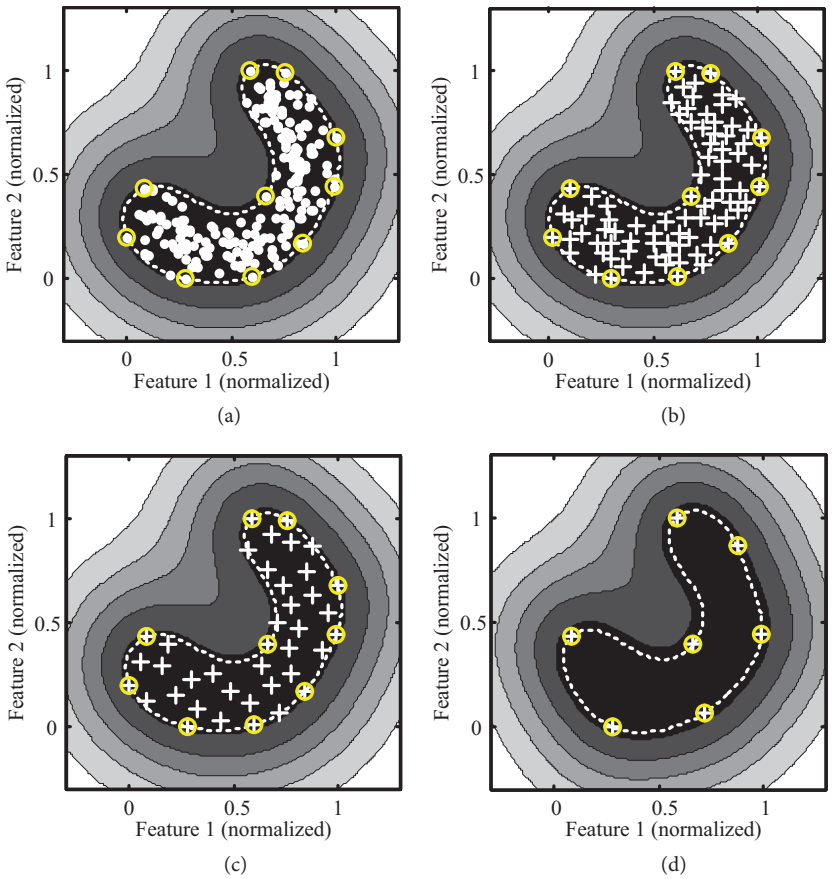

Figure 3. The training results comparison between C-SVDD and H-SVDD using the Banana dataset. (a) The training result of C-SVDD with $C = 1$ and $\sigma = 0.4$, where the white dots denote the training samples, support vectors are indicated by yellow circles, and the dashed line denotes the decision boundary; (b) the training result of H-SVDD with $C = 1$ $\sigma = 0.4$ and $\delta = 0.05$, where the white plus signs denote the reduced training set; (c) the training result of H-SVDD with $C = 1$ $\sigma = 0.4$ and $\delta = 0.1$; (d) the training result of H-SVDD with $C = 1\sigma = 0.4$ and $\delta = 0.3$.

1 to 0, so the hypersphere volume varies from minimum to maximum. Meanwhile, the FPR also varies from 0 to 1. Thus, the TPR can be obtained correspondingly. The area under the ROC curve (AUC) value [30] is also introduced to compare the classification accuracy of classification methods. Generally, the one that has a larger AUC value in a given problem is considered better than the others. All of these experiments were performed on a computer with a Pentium Dual E2140 1.6 GHz CPU, 2 GB RAM, and MATLAB V 7.6.0.

5.1. Sensitivity analysis

A sensitivity analysis of the HSR is performed in order to understand the behavior of the parameters and the analysis results are shown in Figure 4. Figure 4a shows how the ROC curves of H-SVDD vary with δ on the 2D Banana dataset. Predictably, when δ increases, the ROC curve moves toward the lower right corner of the graphic. We also see that the ROC curve with $\delta = 0.1$ nearly coincides with the C-SVDD ROC curve, which implies that the classification ability of H-SVDD with $\delta = 0.1$ and C-SVDD is nearly the same. In Figure 4b, the parameter δ is varied from 0 to 0.5, and then a series of combinations of values for different σ and δ is considered. For each σ, the AUC variation curve is determined. Note that when $\delta = 0$, the H-SVDD coincides with the C-SVDD. All of these AUC value curves keep flatness within the range of [0, 0.125]. The

AUC curve of $\sigma = 0.1$ begins to fall at $\delta = 0.125$ and the AUC curve of $\sigma = 0.2$ begins to fall at $\delta = 0.2$. For $\sigma = 0.3$ and 0.4, the AUC curves begin to fall at $\delta = 0.275$. For $\sigma = 0.45$, the AUC curve does not fall sharply. A series of extended experiments are conducted on datasets of different dimensionalities, and a rule for determining the appropriate value of δ is deduced as follows:

$$0 \leq \delta \leq 0.1 \cdot \sqrt{D}, \tag{23}$$

where D is the dimensionality of the training set.

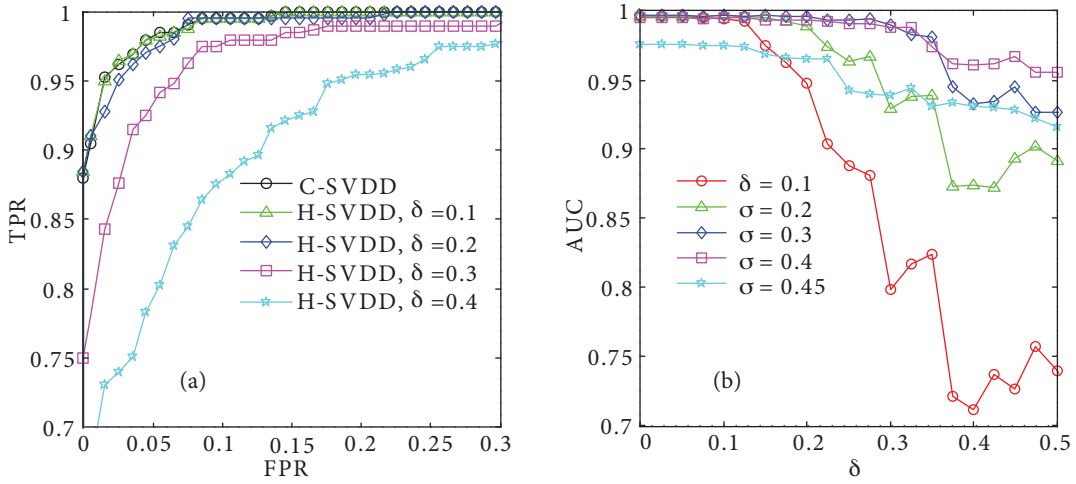

Figure 4. Experimental results of H-SVDD on different parameters. (a) ROC curves of H-SVDD with different δ values and C-SVDD; (b) AUC value curves of H-SVDD with different σ values.

5.2. Experiments using artificial and real-world datasets

In this subsection, a series of experiments on artificial datasets and real-world datasets are conducted to compare the performances among H-SVDD, C-SVDD, and other improved one-class support vector classification methods, including LS-OCSVM [12], SMO-SVDD [13,31] and Kim-SVDD [15]. In addition, a method of combining the sample condensing method with SVDD, which is referred to as CPDD-SVDD [23], is also introduced into the experiments. In each experiment, the parameters have been varied and the best result has been recorded. The artificial datasets including Banana (2D), Lithuanian (2D) and Difficult (4D) are generated by the Prtools toolbox (http://prtools.org/), and the real-world datasets including Letter (16D), Pendigits (16D) and Optdigits (64D) are downloaded from the UCI machine-learning database (http://archive.ics.uci.edu/ml/).

The artificial datasets all contain two classes, one of which serves as targets, while the other serves as outliers. N targets form the training set, and extra N targets and N outliers form the testing set. Table 1 reports the experimental results using artificial datasets when $N = 200$. The column labeled N_{SV} contains the number of support vectors of the training results. From Table 1, we can see that in terms of the AUC, the classification accuracy of C-SVDD is superior to the others; it is slightly higher than that of Kim-SVDD and H-SVDD, and much higher than that of LS-OCSVM, SMO-SVDD, and CPDD-SVDD. Based on the training time results, we can see that LS-OCSVM has the fastest training speed, followed by H-SVDD, Kim-SVDD, CPDD-SVDD, and SMO-SVDD, while C-SVDD is the most time-consuming method. The testing time of support vector classification methods depends on the number of support vectors, which can also be confirmed by the experimental results in Table 1. The greater the support vector number is, the longer the testing time

of classification method becomes. C-SVDD, Kim-SVDD, CPDD-SVDD, and H-SVDD all have fewer support vectors, so they have shorter testing times. The fact that LS-OCSVM and SMO-SVDD have plenty of support vectors also demonstrates that they do not have the global optimal solution for the QP. Table 2 reports the experimental results in the condition of $N = 500$. We can see that the classification accuracy of C-SVDD becomes very low. This is because, in MATLAB, in order to avoid too many computations or impossible tasks, the QP solving function quadprog.m loses the stopping condition when the size of the training set is excessively large. On this occasion, the C-SVDD cannot obtain the global optimal solution, while the Kim-SVDD and H-SVDD show the best classification accuracy. Table 3 shows the experimental results when $N = 1000$. This time, the training speed of H-SVDD surpasses that of LS-OCSVM and ranks first, which means that with the increase of the training set size, the training time growth rate of H-SVDD is much less than that of LS-OCSVM. The classification accuracy of H-SVDD is also the best.

Table 1. Experimental results using artificial datasets with $N = 200$.

	Dataset	Training time (s)	AUC	N_{SV}	Testing time (s)
C-SVDD	Banana	7.673	0.9912	10	0.048
	Lithuanian	7.562	0.9934	14	0.034
	Difficult	3.147	0.8139	29	0.105
LS-OCSVM	Banana	0.140	0.9851	200	0.581
	Lithuanian	0.159	0.9902	200	0.584
	Difficult	0.143	0.7031	198	0.606
SMO-SVDD	Banana	4.882	0.9845	198	0.565
	Lithuanian	4.532	0.9887	196	0.547
	Difficult	3.949	0.7981	184	0.656
Kim-SVDD	Banana	1.405	0.9912	10	0.049
	Lithuanian	1.978	0.9895	16	0.043
	Difficult	1.827	0.8114	30	0.106
CPDD-SVDD	Banana	1.726	0.9884	11	0.049
	Lithuanian	2.659	0.9885	13	0.046
	Difficult	2.792	0.7978	24	0.091
H-SVDD	Banana	1.581	0.9912	10	0.046
	Lithuanian	1.549	0.9909	15	0.037
	Difficult	1.518	0.8122	28	0.098

As for the real-world datasets, we compare the H-SVDD with the other 5 methods mentioned above using the intermediate and high dimensionality datasets. For the 16D Letter dataset, 780 samples of the letter A serve as targets, and the other 19,211 samples serve as outliers; for the 16D Pendigits dataset, 780 samples of the number 2 serve as targets, and the remaining 6714 samples serve as outliers; for the 64D Optdigits dataset, 570 samples of the number 1 serve as targets, and the other 5049 samples serve as outliers. The 10-fold cross validation training method [32] is applied to these experiments, so 10 iterations are needed during the implementation. Within each of the iterations, 9 folds of targets are used to train and the remaining fold targets, and all of the outliers are used to test.

Tables 4, 5, and 6 show the experimental results using the Letter, Pendigits, and Optdigits datasets, respectively. As we can see from these tables, the classification accuracy of H-SVDD is the highest in the experiment with the Letter dataset and is a little lower than that of Kim-SVDD in the experiment with the Pendigits and Optdigits datasets. In addition, compared to the other methods, H-SVDD has both a fast training speed and a fast testing speed.

Table 2. Experimental results using artificial datasets with $N = 500$.

	Dataset	Training time (s)	AUC	N_{SV}	Testing time (s)
C-SVDD	Banana	204.846	0.9147	292	1.185
	Lithuanian	199.886	0.9880	295	1.130
	Difficult	30.870	0.7357	300	1.301
LS-OCSVM	Banana	0.843	0.9881	352	1.746
	Lithuanian	0.847	0.9863	500	1.749
	Difficult	0.872	0.7122	445	1.841
SMO-SVDD	Banana	46.705	0.9868	354	1.872
	Lithuanian	43.893	0.9828	325	1.728
	Difficult	29.798	0.7864	298	1.934
Kim-SVDD	Banana	4.335	0.9909	10	0.056
	Lithuanian	5.205	0.9873	16	0.077
	Difficult	3.246	0.8301	44	0.183
CPDD-SVDD	Banana	8.029	0.9836	10	0.058
	Lithuanian	10.890	0.9861	14	0.091
	Difficult	9.419	0.8141	35	0.185
H-SVDD	Banana	1.641	0.9897	11	0.058
	Lithuanian	2.045	0.9864	14	0.093
	Difficult	1.661	0.8358	34	0.144

Table 3. Experimental results using artificial datasets with $N = 1000$.

	Dataset	Training time (s)	AUC	N_{SV}	Testing time (s)
C-SVDD	Banana	1217.163	0.9729	755	3.283
	Lithuanian	1199.092	0.9895	763	3.201
	Difficult	283.688	0.7749	794	3.527
LS-OCSVM	Banana	4.360	0.9886	545	4.607
	Lithuanian	4.369	0.9800	753	4.662
	Difficult	4.479	0.7525	709	4.823
SMO-SVDD	Banana	340.621	0.9612	693	3.098
	Lithuanian	337.728	0.9816	673	3.915
	Difficult	256.554	0.7823	772	3.898
Kim-SVDD	Banana	58.303	0.9835	10	0.075
	Lithuanian	55.267	0.9848	15	0.098
	Difficult	29.134	0.8324	48	0.263
CPDD-SVDD	Banana	26.943	0.9879	9	0.072
	Lithuanian	28.993	0.9819	14	0.098
	Difficult	30.521	0.8227	39	0.213
H-SVDD	Banana	2.773	0.9878	11	0.076
	Lithuanian	3.331	0.9839	13	0.104
	Difficult	2.907	0.8252	41	0.226

5.3. Experiments of novelty detection on flight data

Flight data recorded by a FDR are well-known tools for investigating the cause of air crashes. However, flight data are also widely used for monitoring the condition of airborne equipment. A qualified flight data interpreter can detect hidden faults by reading the flight data. In China, flight data interpretation is an essential step in the military airplane maintenance system. Only the aircrafts without abnormalities in their former sortie flight data interpretation can participate in subsequent flight missions. Up until now, all of the flight data interpretation

work has been done manually. It is a very difficult and boring job, and it has a hidden danger of misdetection. With the rapid development of computer hardware and algorithms, an automatic novelty detection method is an effective way to replace manual interpretation work. Given the outstanding performance of H-SVDD described in the above sections, the novelty detection work will be done using H-SVDD.

Table 4. Experimental results using Letter datasets (16D).

	Training time (s)	AUC	N_{SV}	Testing time (s)
C-SVDD	96.936	0.9976	486.7	8.389
LS-OCSVM	1.968	0.9618	701.3	32.505
SMO-SVDD	87.241	0.9767	698.2	19.931
Kim-SVDD	22.805	0.9987	151.8	7.734
CPDD-SVDD	73.067	0.9943	141.5	6.628
H-SVDD	2.874	0.9989	128.4	6.420

Table 5. Experimental results using the Pendigits dataset (16D).

	Training time (s)	AUC	N_{SV}	Testing time (s)
C-SVDD	92.113	0.9858	491.4	7.296
LS-OCSVM	1.975	0.9610	702.0	12.009
SMO-SVDD	84.519	0.9635	692.7	10.129
Kim-SVDD	29.787	0.9939	190.6	3.973
CPDD-SVDD	62.264	0.9814	232.7	4.707
H-SVDD	2.873	0.9922	179.1	3.921

Table 6. Experimental results using the Optdigits dataset (64D).

	Training time (s)	AUC	N_{SV}	Testing time (s)
C-SVDD	36.404	0.9673	308.6	4.612
LS-OCSVM	1.027	0.9828	505.0	7.857
SMO-SVDD	32.876	0.9762	484.3	7.541
Kim-SVDD	23.506	0.9875	54.4	0.798
CPDD-SVDD	135.085	0.9687	53.5	0.792
H-SVDD	2.274	0.9870	52.9	0.781

In the flight data interpretation of a certain type of airplane, the aeroengine gas path parameters are one of the primary focuses. Interpreters analyze 2 important parameters: high-pressure rotor speed (HPRS) and exhaust gas temperature (EGT). In the normal state, HPRS and EGT have a correspondence in that the EGT increases with an increase in HPRS. If this correspondence breaks up, this indicates a possible abnormality, and the interpreter reports the error. Figure 5 shows a sortie of flight data containing a section of abnormalities. The left subplot in Figure 5 shows the correspondence of HPRS and EGT in the normal state, and the right subplot shows the parameters in an abnormal state. From 2490 s, HPRS decreases sharply, but EGT does not decrease correspondingly. On the contrary, EGT increases sharply in the latter case. This phenomenon lasts for about 15 s and then the parameters gradually retreat from the abnormal state. By post hoc analysis, it is a typical surge. It happens because the aircraft inhales the gas from the exhaust from the plane in front of it.

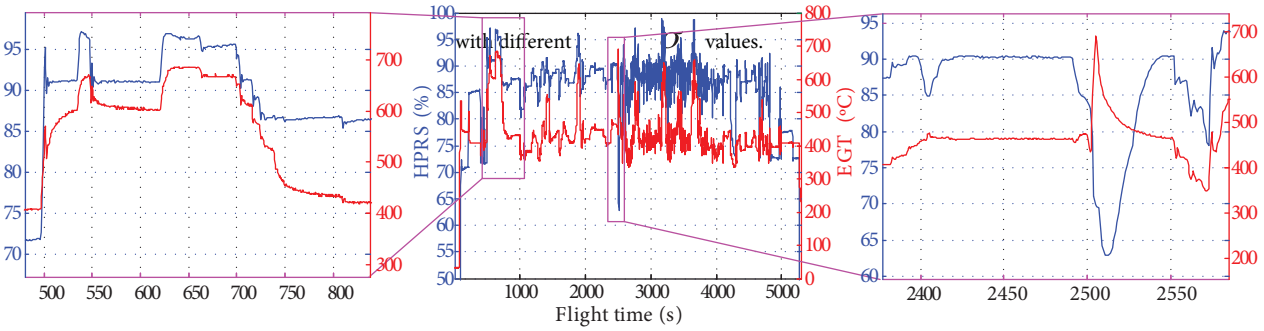

Figure 5. Flight data curves of HPRS and EGT in one sortie, where the red line denotes EGT and the blue line denotes HPRS.

When detecting novelty in gas parameters using H-SVDD, HPRS and EGT can be used as the features. The sample rate of HPRS and EGT in this type of FDR are both 2 Hz, so it generates 2 samples per second. In this experiment, samples generated by 86 flight sorties including 531,793 s of normal state flight data form the training set. The size of the training set is 1,063,586. The testing set is formed by the sortie of flight data shown in Figure 5. The parameters δ, C, and σ are set to 0.05, 1, and 0.3, respectively. Figure 6 shows the testing results. The classification label of 0 denotes that the state of the aeroengine gas path is normal and a classification label of 1 denotes that it is abnormal. From Figure 6, abnormality appears from 2503 to 2521 s, and no false alarm happens, which means that H-SVDD has detected the abnormality of the aeroengine condition accurately.

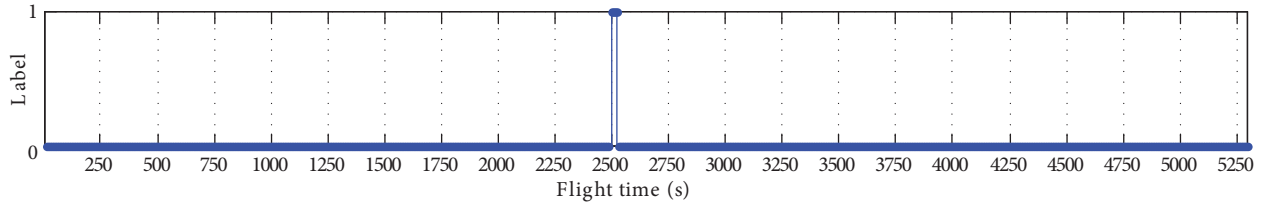

Figure 6. Novelty detection results of aeroengine gas path parameters using H-SVDD.

6. Conclusions

In this paper, we addressed the issue of speeding up the training of SVDD. A sample reduction method, which is referred to as HSR, is proposed. The method first finds the cluster centers of the training set by k-means. All of the training samples are then sorted in descending order according to the distance from the corresponding cluster center. After that, the farthest sample is moved from the sorted training set into the reduced training set. The samples that have a shorter distance to the farthest sample from δ are then deleted from the training set. The above process is repeated until the sorted training set is empty, and the reduced training set is built. In this way, HSR reduces the time complexity of SVDD from $O\left(N^3\right)$ to $O\left(\tilde{N}^3\right)$, where $\tilde{N}$ is the size of the reduced training set, by cutting out a large portion of samples in the training set. The efficiency of HSR was theoretically proven by an efficiency analysis of SVDD with a different σ, including small σ, intermediate σ, and large σ. The experiment was carried out on artificial datasets generated by the Prtools toolbox, real-world datasets downloaded from the UCI database, and flight data recorded by a FDR. Computational comparisons with conventional SVDD, LS-OCSVM, SMO-SVDD, Kim-SVDD, and CPDD-SVDD have shown that SVDD based on the proposed HSR method not only has a fast training and testing speed, but also provides high classification accuracy. For flight data novelty detection, H-SVDD detects the abnormalities in the aeroengine

gas path parameters effectively, which shows that H-SVDD is a promising method for realizing flight data automatic interpolation.

Acknowledgment

This research was supported by the Aeronautical Science Fund of China under Grant No. 20101785005.

References

[1] Tax D, Duin RP. Support vector domain description. Pattern Recognit Lett 1999; 20: 1191–1199.

[2] Tax D. One class classification: concept-learning in the absence of counter-examples. PhD, University of Delft, Delft, the Netherlands, 2001.

[3] Guo SM, Chen LC, Tsai JSH. A boundary method for outlier detection based on support vector domain description. Pattern Recognit 2009; 42: 77–83.

[4] Lee K, Kim DW, Lee D, Lee KH. Improving support vector data description using local density degree. Pattern Recognit 2005; 38: 1768–1771.

[5] Wang S, Yu J, Lapira E, Lee J. A modified support vector data description based novelty detection approach for machinery components. Appl Soft Comput 2012; 13: 1193–1205.

[6] Peng X, Xu D. Efficient support vector data descriptions for novelty detection. Neural Comput Appl 2012; 21: 2023–2032.

[7] Liu YH, Liu YC, Chen YJ. Fast support vector data descriptions for novelty detection. IEEE T Neural Networ 2010; 21: 1296–1313.

[8] Park J, Kang D, Kim J, Kwok JT, Tsang IW. SVDD-based pattern denoising. Neural Comput 2007; 19: 1919–1938.

[9] Camastra F, Verri A. A novel kernel method for clustering. IEEE T Pattern Anal 2005; 27: 801–805.

[10] Ben-Hur A, Horn D, Siegelmann HT, Vapnik V. Support vector clustering. J Mach Learn Res 2002; 2: 125–137.

[11] Huang G, Chen H, Zhou Z, Yin F, Guo K. Two-class support vector data description. Pattern Recognit 2011; 44: 320–329.

[12] Choi YS. Least squares one-class support vector machine. Pattern Recognit Lett 2009; 30: 1236–1240.

[13] Platt JC. Fast training of support vector machines using sequential minimal optimization. In: Schölkopf B, Burges CJC, Smola AJ, editors. Advances in Kernel Methods–Support Vector Learning. Cambridge, MA, USA: MIT Press, 1999. pp. 185–208.

[14] Schölkopf B, Platt JC, Shawe-Taylor J, Smola AJ, Williamson RC. Estimating the support of a high-dimensional distribution. Neural Comput 2001; 13: 1443–1471.

[15] Kim PJ, Chang HJ, Song DS, Choi JY. Fast support vector data description using k-means clustering. In: ISNN 2007 4th International Symposium on Neural Networks; 3–7 June 2007; Nanjing, China. pp. 506–514.

[16] Hart PE. The condensed nearest neighbor rule. IEEE T Inform Theory 1968; 14: 515–516.

[17] Gates GW. The reduced nearest neighbor rule. IEEE T Inform Theory 1972; 18: 431–433.

[18] Ritter G, Woodruff H, Lowry S, Isenhour T. An algorithm for a selective nearest neighbor decision rule. IEEE T Inform Theory 1975; 21: 665–669.

[19] Hastie T, Tibshirani R. Discriminant adaptive nearest neighbor classification. IEEE T Pattern Anal 1996; 18: 607–616.

[20] Wilson DR, Martinez TR. Reduction techniques for instance-based learning algorithms. Mach Learn 2000; 38: 257–286.

[21] Aha DW, Kibler D, Albert MK. Instance-based learning algorithms. Mach Learn 1991; 6: 37–66.

[22] Rico-Juan JR, Iñesta JM. New rank methods for reducing the size of the training set using the nearest neighbor rule. Pattern Recognit Lett 2012; 33: 654–660.

[23] Angiulli F. Prototype-based domain description for one-class classification. IEEE T Pattern Anal 2012; 34: 1131–1144.

[24] Yuan Y. Forecasting the movement direction of exchange rate with polynomial smooth support vector machine. Math Comput Model 2012; 57: 932–944.

[25] Camps-Valls G, Martin-Guerrero J, Rojo-Alvarez J, Soria-Olivas. Fuzzy sigmoid kernel for support vector classifiers. Neurocomputing 2004; 62: 501–506.

[26] Yuan C, Casanent D. Support vector machines for class representation and discrimination. In: International Joint Conference on Neural Networks; 20–24 July 2003; Portland, OR, USA. pp. 1610–1615.

[27] Su T, Dy J. A deterministic method for initializing K-means clustering. In: 16th IEEE International Conference on Tools with Artificial Intelligence; 15–17 November 2004; Boca Raton, FL, USA. pp. 784–789.

[28] Hamerly G, Elkan C. Learning the k in k-means. Adv Neural Inf Process Syst 2004; 16: 281.

[29] Bischof H, Leonardis A, Selb A. MDL principle for robust vector quantisation. Pattern Anal Appl 1999; 2: 59–72.

[30] Bradley AP. The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognit 1997; 30: 1145–1159.

[31] Hu L, Hu NQ, Qin GJ. Support vector machines detection method for turbopump test data analysis. J Propul Technol 2008; 29: 244–248.

[32] Refaeilzadeh P, Tang L, Liu H. Cross-validation. In: Liu L, Özsu MT, editors. Encyclopedia of Database Systems. Berlin, Germany: Springer, 2009. pp. 532–538.