Online Support Vector Machine Based on Convex Hull Vertices Selection

Di Wang, Hong Qiao, *Senior Member, IEEE*, Bo Zhang, *Member, IEEE*, and Min Wang

***Abstract*— The support vector machine (SVM) method, as a promising classification technique, has been widely used in various fields due to its high efficiency. However, SVM cannot effectively solve online classification problems since, when a new sample is misclassified, the classifier has to be retrained with all training samples plus the new sample, which is time consuming. According to the geometric characteristics of SVM, in this paper we propose an online SVM classifier called VS-OSVM, which is based on convex hull vertices selection within each class. The VS-OSVM algorithm has two steps: 1) the samples selection process, in which a small number of skeleton samples constituting an approximate convex hull in each class of the current training samples are selected and 2) the online updating process, in which the classifier is updated with newly arriving samples and the selected skeleton samples. From the theoretical point of view, the first $d+1$ (d is the dimension of the input samples) selected samples are proved to be vertices of the convex hull. This guarantees that the selected samples in our approach keep the greatest amount of information of the convex hull. From the application point of view, the new algorithm can update the classifier without reducing its classification performance. Experimental results on benchmark data sets have shown the validity and effectiveness of the VS-OSVM algorithm.**

***Index Terms*— Kernel, machine learning, online classifier, samples selection, support vector machine.**

I. Introduction

SUPPORT vector machine (SVM) [1] is a well-known classification and regression method first proposed by Vapnik in 1995 and based on the structural risk minimization (SRM) principle [2], [3]. Geometrically, it is to find a hyperplane with a maximal margin between two classes. The main advantages of the SVM method are summarized by Shilton *et al*. [4] as follows. First, because of the combination of the empirical risk minimization and the prevention of overfitting, SVM can achieve a good generalization performance. Second, the problem of computing the hyperplane with a maximal margin can be converted into a convex quadratic optimization problem, which can be solved by quadratic programming techniques and has a global optimal solution. Finally, the obtained classifier is determined only by support vectors, and it can also be applied to nonlinear cases by using the kernel trick. Due to the above advantages, SVM has been successfully used in many areas such as face recognition [5]–[8], pedestrian detection [9], hand-writing digital character recognition [10], and classification tasks of text [11], [12].

However, a main drawback of SVM is that it requires a long time for training and a lot of memory for dealing with large-scale data sets. Therefore, many techniques, such as sequential minimal optimization (SMO) [13]–[17], decomposed support vector machine (DSVM) [18]–[23] and core vector machine (CVM) [24], [25], have been proposed to speed up SVM and have been successfully applied to solve many large-scale classification problems (see [29], [30] for good literature surveys on this topic). However, the online learning issue of SVM is still not addressed satisfactorily. We need to retrain the SVM to adjust the classifier when a new sample arrives. This may not be feasible for real-time applications due to the expensive computation cost for re-training the SVM.

In the field of machine leaning, online learning is referred to as continuous updating of a classifier with a potentially endless flow of newly arrived samples such that the generalization ability of the classifier can be gradually improved [30], [45], [55]. Online learning has important applications in real-time pattern recognition systems, such as pedestrian detection and aircraft visual navigation systems. In order to address the online learning issue of SVM, several successful algorithms [31], [32] have been proposed, which will be reviewed in the next section. There are few works that consider updating SVM online by preserving the skeleton samples based on the geometric characteristics of SVM. Geometrically, for linearly separable data sets, the computation of SVM is equivalent to the problem of computing the nearest points between the convex hulls formed by the positive and negative samples [26]–[28]. Therefore, we can permanently and safely delete samples within the convex hulls and only preserve the vertices of the convex hulls for online learning. The classification result of the online classifier trained with only the vertices of the convex hulls and newly arrived samples is the same as that obtained by retraining with all samples.

Based on the above motivation, in this paper, we propose an online SVM classification algorithm. A main advantage of

Manuscript received May 30, 2012; revised November 9, 2012; accepted December 30, 2012. Date of publication January 25, 2013; date of current version February 13, 2013. This work was supported in part by NNSFC, under Grant 61033011, Grant 60725310, Grant 61100098, and Grant 11131006.

D. Wang is with the College of Mathematics and Information Sciences, Wenzhou University, Wenzhou 325035, China (e-mail: wangdi@amss.ac.cn).

H. Qiao and M. Wang are with the Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China (e-mail: hong.qiao@ia.ac.cn; min.wang@ia.ac.cn).

B. Zhang is with LSEC and the Institute of Applied Mathematics, AMSS, Chinese Academy of Sciences, Beijing 100190, China (e-mail: b.zhang@amt.ac.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2013.2238556

our approach over the existing ones is that it can deal with large-scale data sets efficiently since only a small number of samples are selected for online learning. There are two main steps in our algorithm, which are described as follows.

1) *Off-Line Samples Selection:* The skeleton vertices of convex hulls, which are formed by the current positive and negative training samples, are selected and stored for the following online adjustment step.
2) *Online Classifier Adjustment:* The SVM classifier is updated based on samples selected in the off-line samples selection step and the newly arriving samples whose distances to the current classification hyperplane are within a given threshold. Online updating of the classifier can be achieved since only very limited training samples are maintained in the off-line step.

We apply the off-line step only in the case when the number of the current training samples, which consist of the selected samples and all newly arrived samples exceeds a large number. We execute periodically the off-line step in order to reduce the number of the current training samples. This makes the online learning with newly arrived samples possible. Experimental results on benchmark data sets have shown the validity and effectiveness of the proposed method.

The rest of this paper is organized as follows. In Section II, the existing online learning algorithms for SVM are reviewed. The convex hull vertices selection algorithm is described in Section III. In Section IV, a detailed theoretical analysis of the proposed algorithm is given. The online SVM based on the convex hull vertices selection is presented in Section V. In Section VI, experiments are conducted on benchmark data sets to illustrate the validity and effectiveness of the proposed method. Some concluding remarks are given in Section VII. Proofs of the main theorems in Section IV are presented in the appendix.

II. Existing Online SVM Algorithms

In this section, the existing online learning methods for SVM are briefly reviewed. These approaches can be divided into two groups: online methods by solving the primal problem of SVM and online methods by solving the dual problem of SVM.

In the first group, many works use the stochastic gradient descent (GSD) method to address the online learning issue of SVM [31], [33]–[35]. GSD-based methods find the gradient with respect to only one randomly selected sample instead of all training samples. Therefore, if the number of the training samples is very large, these GSD-based methods can run significantly faster than the gradient descent methods. Due to this feature, they can be applied to online adjustment of the SVM classifier when new training samples arrive.

On the other hand, several online learning algorithms based on solving the dual problem of SVM have also been proposed. Syed *et al.* [36] and Mitra *et al.* [37] retrained the classifier with newly arrived samples and support vectors of the current trained SVM. To speed up the above algorithms, several works [38]–[40] have introduced a questing process. When new samples arrive, they are quested by their distances to the current SVM classification hyperplane. Only the support vectors of the current model and the samples whose distances to the hyperplane are within a given threshold are used to update the classifier. Obviously, using only the support vectors of the current classifier in the updating process will possibly result in a large deviation from the final true classifier for many data sets. Singh *et al.* [41] proposed a method called 2ν-OGSSVM, in which the decision hyperplane is trained with the initial training samples. When a new sample arrives, the classifier is retrained by incorporating the new sample into the decision hyperplane, and the current existing support vectors that are unnecessary and irrelevant are then removed. This procedure can guarantee that the number of support vectors does not increase dramatically with the increasing of the training samples. Bordes *et al.* [42] proposed an online method called LASVM, which is based on SMO. The update of the classifier is achieved by alternating two kinds of direction searches, called PROCESS and REPROCESS, when new samples arrive. Inspired by LASVM, Wang *et al.* [43] proposed an online support vector classifier algorithm, which is used in cell phase identification. To accommodate the ever-changing experimental conditions, the penalty weights of the training samples are reassigned according to their importance when new training samples arrive, and the support vectors in the old model will be dropped out once they become non-support vectors during optimization. Cheng *et al.* [44] proposed an incremental SVM by using active query. They first select a subset of the training samples using the K-means clustering to compute an initial separating hyperplane. Then in each iteration, the classifier is updated based on the training samples selected by active query, and the iteration stops when there are no unused informative training samples. Lau *et al.* [45] presented an online support vector classifier, in which the classifier is adjusted by retraining the SVM with the current support vectors, the samples violating the Karush-Kuhn-Tucker (KKT) conditions and the newly arriving samples. The updating process stops only if no sample violates the KKT conditions and no new sample arrives. There are also algorithms [32], [46] which consider the change of the KKT conditions to update the classifier. When a new sample comes, they utilize the change of the KKT conditions to compute the perturbation of the classifier coefficients, and online updating of the classifier can then be achieved. Wang *et al.* [47] proposed an online CVM classifier that updates the classifier coefficients by using an adaptive minimum-enclosing-ball adjustment.

Bordes *et al.* [48] proposed an efficient online SVM algorithm, named Huller. When a new misclassified sample arrives, Huller attempts to make this new sample become a support vector and removes another support vector from the current classifier via the update of the nearest points of the convex hulls. The classifier is thus adjusted. The Huller algorithm is closely related to our method in this paper. Both our algorithm and the Huller algorithm update the classifier based on the geometrical interpretation of SVM. However, the numerical objective of the two algorithms is different. Our method tries to select the vertices of the convex hulls of the current training samples in the off-line step and then

retrains the classifier using these selected samples and newly arrived samples. Since the training set is greatly reduced in the off-line step, the online learning is tractable. In each online iteration, our algorithm gives the optimal solution of the SVM classifier trained with the selected samples and all newly arrived samples, while, when a new sample arrives, the Huller algorithm tries to decrease the distance of the current nearest points to update the classifier. But the classifier obtained by Huller is not the optimal solution of the SVM classifier in the current iteration.

III. Convex Hull Vertices Selection Algorithm

In this section, we give a detailed description of the proposed method. In Section III-A, we introduce the kernel convex hull distance and the linear subspace distance. In Section III-B, we use a synthetic example to illustrate the proposed method. In Section III-C, we propose a method of selecting $d+1$ vertices of a d-simplex in $\mathbb{R}^d$. In Section III-D, we introduce an approach for data partition such that the number of each part is no more than a given bound. In Section III-E, we give a description of the proposed convex hull vertices selection algorithm. In Section III-F, We propose a method that can convert samples in a (possibly infinite-dimensional) feature space into samples in a finite dimensional space. As proved in Theorem 3 below, the vertices of the convex hull of the samples in the finite dimensional space also correspond to the vertices of the convex hull of the samples in the feature space. Thus the vertices of the convex hull of the samples in the feature space can be obtained by selecting the vertices of the convex hull of the samples in the finite dimensional space with Algorithm 2.

A. Distances

In the proposed method, we will use two distances: the distance of a point to a linear subspace and the distance of a point to a convex hull, which are named the linear subspace distance and the convex hull distance, respectively. In this section, we will give a brief description of how to compute these two distances.

1) Linear Subspace Distance in $\mathbb{R}^d$:

Definition 1: For $\tilde{U} = \{u_i\}_{i=1}^n \subset \mathbb{R}^d$, the linear subspace spanned by $\tilde{U}$ is defined as

$$\mathcal{L}(\tilde{U}) = \left\{ \sum_{i=1}^{n} \alpha_i u_i | u_i \in \tilde{U}, \alpha_i \in \mathbb{R} \right\}.$$

Given a set $\tilde{U} = \{u_i\}_{i=1}^n \subset \mathbb{R}^d$ and a vector $u \in \mathbb{R}^d$, the distance between u and $\mathcal{L}(\tilde{U})$ can be computed by solving the following quadratic optimization problem:

$$\min_{\alpha} \frac{1}{2}\alpha^\top Q\alpha - c^\top \alpha \tag{1}$$

where $U = [u_1, \ldots, u_n]$, $Q = U^\top U$ and $c = U^\top u$.

Equation (1) is an unconstrained quadratic optimization problem so, if Q is invertible, then the global optimal solution of (1) is given by $\alpha^* = Q^{-1}c$. Then the distance between u and $\mathcal{L}(\tilde{U})$ is given by

$$d_s(u, \mathcal{L}(\tilde{U})) = \sqrt{u^\top u - 2c^\top \alpha^* + \alpha^{*\top} Q\alpha^*}. \tag{2}$$

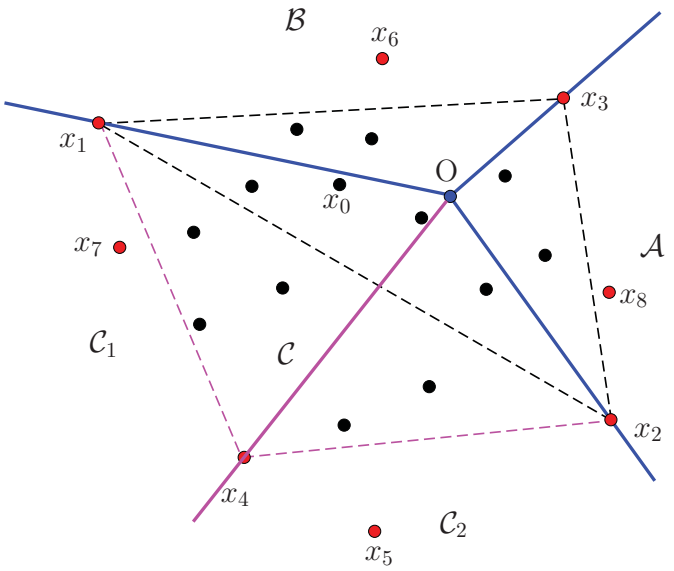


Fig. 1. Schematic diagram of the convex hull vertices selection algorithm.

2) Convex Hull Distance:

Definition 2: The convex hull of the set $P = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$ is defined as: $\text{conv}(P) = \{\sum_{i=1}^n \alpha_i x_i | x_i \in P, \sum_{i=1}^n \alpha_i = 1, \alpha_i \geq 0, i = 1, \ldots, n\}$.

Given a set $P = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$ and a point $x \in \mathbb{R}^d$, the Euclidean distance between x and $\text{conv}(P)$ can be computed by solving the following quadratic optimization:

$$\begin{aligned} &\min_{\alpha} \ \frac{1}{2}\alpha^\top Q\alpha - c^\top \alpha \\ &\text{s.t.} \ \ e^\top \alpha = 1, \ \alpha \geq 0 \end{aligned} \tag{3}$$

where $e = [1, 1, \ldots, 1]^\top$, $Q = X^\top X$ and $c = X^\top x$ with $X = [x_1, \ldots, x_n]$.

Suppose the optimal solution of (3) is α^*. Then the convex hull distance between x and $\text{conv}(P)$ is given by

$$d_c(x, \text{conv}(P)) = \sqrt{x^\top x - 2c^\top \alpha^* + \alpha^{*\top} Q\alpha^*}. \tag{4}$$

B. Synthetic Example

Before formally introducing the samples selection method, we present a synthetic example in $\mathbb{R}^2$ (see Fig. 1) to explain the basic idea of the proposed method.

As shown in Fig. 1, we first select three samples with the following steps. First, randomly select a sample, denoted by x_0. Second, the furthest sample to x_0 is selected and denoted by x_1. Thirdly, the furthest sample to x_1 is selected and denoted by x_2. Finally, the sample that is the furthest to the straight line passing through x_1 and x_2 is selected and denoted by x_3. Then $\{x_1, x_2, x_3\}$ are the samples we need. The above steps are the main steps of Algorithm 1 presented in Section III-C. The aim of Algorithm 1 is to select the $d+1$ samples that can construct an as large d-simplex as possible.

Suppose O is the mean of the samples $\{x_1, x_2, x_3\}$. Then radials Ox_1, Ox_2 and Ox_3 divide the samples outside of $\text{conv}(\{x_1, x_2, x_3\})$ into three parts, denoted by $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$, respectively. From Fig. 1, it can be seen that the number of samples in part $\mathcal{C}$ is still large, so we need to further divide

Algorithm 1 Selecting $d+1$ Vertices of a d-Simplex in $\mathbb{R}^d$

Input: $P = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$.

1: Randomly select $x_0 \in P$. Let $x_{j_0} = \arg\max_{x\in P} \|x_0 - x\|$, and $x_{j_1} = \arg\max_{x\in P} \|x_{j_0} - x\|$. Initialize $S_1 = \{x_{j_0}, x_{j_1}\}$ and $t = 1$.
2: Let $\tilde{U} = \{u_i | u_i = x_i - x_{j_0}\}_{i=1}^n$ and $\tilde{U}_{S_1} = \{u_{j_1}\}$.
3: **while** $t < d$ **do**
4: Use (2) to compute $i_0 = \arg \max_{i\in\{i|u_i\in\tilde{U}\setminus\tilde{U}_{S_t}\}} d_s(u_i, \mathcal{L}(\tilde{U}_{S_t}))$.
5: Let $S_{t+1} = S_t \cup \{x_{i_0}\}$, $\tilde{U}_{S_{t+1}} = \tilde{U}_{S_t} \cup \{u_{i_0}\}$ and $t = t+1$.
6: **end while**
7: Let $S = S_t$.

Output: S.

this part into several smaller parts. In part $\mathcal{C}$, the sample which is furthest to the line x_1x_2 is selected and denoted by x_4. Then the radial Ox_4 further divides the samples in $\mathcal{C}$ but outside of conv($\{x_1, x_2, x_3, x_4\}$) into two parts, denoted by $\mathcal{C}_1$ and $\mathcal{C}_2$. So far, we have divided the samples outside of conv($\{x_1, x_2, x_3, x_4\}$) into four parts, and the numbers of samples in the four parts are all small. The edges x_2x_3, x_1x_3, x_1x_4, and x_2x_4 are called the corresponding edges of parts $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}_1$, and $\mathcal{C}_2$, respectively. The above procedure is the main idea of Algorithm 2 presented in Section III-D, the aim of which is to divide samples into several small parts.

From Fig. 1, it can be seen that the distance to conv($\{x_1, x_2, x_3, x_4\}$) of each sample in each part is equal to its distance to the corresponding edge of that part (this will be illustrated in Remark 1). For example, the distance to conv($\{x_1, x_2, x_3, x_4\}$) of each sample in part $\mathcal{A}$ is equal to the distance of the sample to conv($\{x_2, x_3\}$) (the corresponding edge of part $\mathcal{A}$). We call the convex hull of the samples in one part as a sub-convex hull which is initialized to be the corresponding edge of that part, that is, the sub-convex hulls of parts $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}_1$, and $\mathcal{C}_2$ are initialized to be the edges x_2x_3, x_1x_3, x_1x_4, and x_2x_4, respectively. We individually compute the distances between the samples and the sub-convex hull in each part. Then the furthest sample is selected, which is denoted as x_5. The sub-convex hull of part $\mathcal{C}_2$ which x_5 belongs to is then spanned by adding x_5 to the vertices of the sub-convex hull. The distances between the samples in part $\mathcal{C}_2$ and the spanned sub-convex hull are recomputed, while the distances between samples in other parts and their corresponding sub-convex hulls will not be changed. Then the next furthest sample will be selected. In such a procedure, samples x_6, x_7, and x_8 are selected in turn. These are the main procedures of Algorithm 3 given in Section III-E. Note that if we denote by m_d the distance between the next chosen sample and the sub-convex hull of the part which it belongs to, then the convex hull of all selected samples can approach any sample with an error not exceeding m_d.

Remark 1: As shown in Fig. 2, assume that $x_1, \ldots, x_v$ are the selected samples in Algorithms 1 and 2. Then the radials $Ox_1, \ldots, Ox_v$ can divide the samples out of conv($\{x_1, \ldots, x_v\}$) into parts $\mathcal{P}_1, \ldots, \mathcal{P}_v$ such that the number

Algorithm 2 Constrained Data Partition

Input: $P \subset \mathbb{R}^d$, $S = \{x_i\}_{i=1}^{d+1}$ obtained by Algorithm 1 and l_0.

1: Initialize $P_{oc} = \{P_i\}_{i=1}^{d+1}$ and $S_{oc} = \{S_i\}_{i=1}^{d+1}$, where P_i is defined by (6), and $S_i = \{x_i\}_{\substack{k=1\\k\neq i}}^{d+1}$ for $i = 1, \ldots, d+1$.
2: Let $I = \{i | P_i \in \tilde{P}, |P_i| > l_0\}$ and $N = |I|$.
3: **while** $N > 0$ **do**
4: **for** $k = 1; k \le N; k++$ **do**
5: Let $x_{I(k)}^{(d+1)} = \arg \max_{x\in P_{I(k)}} \min_{\sum_{i=1}^d \alpha_i = 1} \|x - \sum_{i=1}^d \alpha_i x_{I(k)}^{(i)}\|^2$.
6: Let $B_i = \{x | x \in P_{I(k)}, \beta = W_i^{-1}(x - \bar{x}), \sum_{t=1}^d \beta_t \ge 1, \beta \ge \mathbf{0}\}$ and $E_i = \{x_{I(k)}^{(t)}\}_{\substack{t=1\\t\neq i}}^{d+1}$ for $i = 1, \ldots, d$, where $W_i = [x_{I(k)}^{(1)} - \bar{x}, \ldots, x_{I(k)}^{(i-1)} - \bar{x}, x_{I(k)}^{(i+1)} - \bar{x}, \ldots, x_{I(k)}^{(d+1)} - \bar{x}]$.
7: Let $P_{oc} = P_{oc} \cup (\bigcup_{i=1}^d B_i)$, $S_{oc} = S_{oc} \cup (\bigcup_{i=1}^d E_i)$ and $S = S \cup \{x_{I(k)}^{(d+1)}\}$.
8: **end for**
9: Let $P_{oc} = P_{oc} \setminus (\bigcup_{k=1}^N P_{I(k)})$ and $S_{oc} = S_{oc} \setminus (\bigcup_{k=1}^N S_{I(k)})$.
10: Suppose $M = |P_{oc}|$, we denote $P_{oc} = \{P_i\}_{i=1}^M$ and $S_{oc} = \{S_i\}_{i=1}^M$.
11: Let $I = \{i | P_i \in P_{oc}, |P_i| > l_0\}$, and $N = |I|$.
12: **end while**

Output: S, P_{oc} and S_{oc}.

of samples in each part is no more than a given bound. Let x be a sample in part $\mathcal{P}_{i_0-1}$ and let $\bar{x}$ be the projection of x onto conv($\{x_1, \ldots, x_v\}$). Then $\bar{x}$ must be on one edge of conv($\{x_1, \ldots, x_v\}$). Suppose $\bar{x}$ is not on the corresponding edge of part $\mathcal{P}_{i_0-1}$, and for convenience, assume that $\bar{x}$ is on the edge $x_{i_0}x_{i_0+1}$, as shown in Fig. 2. Then there exists an intersection of the line segment $x\bar{x}$ with the radial Ox_{i_0}, denoted by $\tilde{x}$ ($x \neq x_{i_0}$). Since x_{i_0} is strictly inside of conv($\{x_{i_0-1}, x_{i_0+1}, \tilde{x}\}$) and $\tilde{x}$ is on the line segment $\bar{x}x$, then x_{i_0} is strictly inside conv($\{x_{i_0-1}, x_{i_0+1}, x\}$). This contradicts to the selection of x_{i_0}, which is a vertex sample if x_{i_0} is selected in Algorithm 1 or is furthest to the line $x_{i_0+1}x_{i_0-1}$ if x_{i_0} is selected in Algorithm 2. Hence, the distance to conv($\{x_1, \ldots, x_v\}$) of every sample in each part is equal to its distance to the corresponding edge of that part.

C. Vertices Selection of a d-Simplex in $\mathbb{R}^d$

In this section, we present an overall description of how to select $d+1$ vertices of a d-simplex in $\mathbb{R}^d$, which is summarized in Algorithm 1.

In Algorithm 1, S is the set of $d+1$ vertices of the d-simplex and S_t is the set of the selected vertices in the tth iteration. Two samples denoted by x_{j_0} and x_{j_1} are first selected

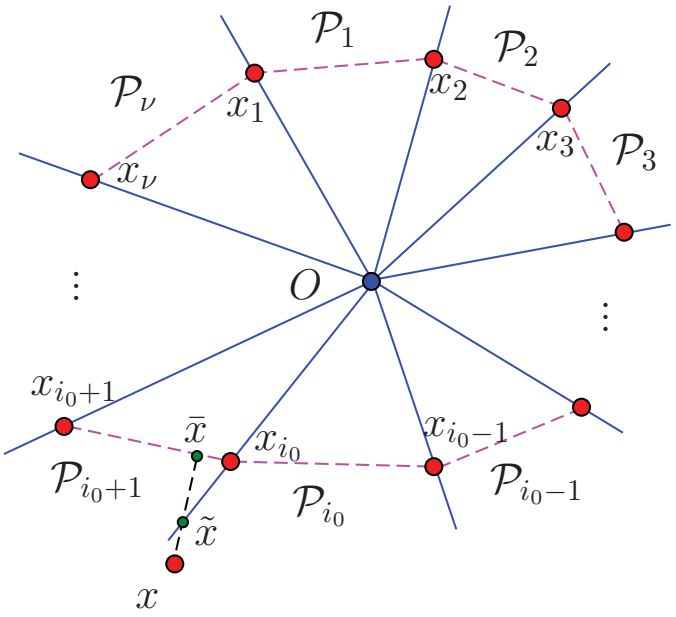


Fig. 2. Illustration of the convex hull distance in $\mathbb{R}^d$.

in Step 1. As shown in Lemma 1, the two samples are the vertices of the d-simplex. Let $\tilde{U} = \{u_i | u_i = x_i - x_{j_0}\}_{i=1}^n$ and $\tilde{U}_{S_1} = \{u_{j_1}\}$, where $u_{j_1} = x_{j_1} - x_{j_0}$. Then the third sample, denoted as x_{j_2}, is selected such that u_{j_2} is the furthest to the current linear subspace $\mathcal{L}(\tilde{U}_{S_1})$, where $u_{j_2} = x_{j_2} - x_{j_0}$. Now span the current linear subspace by adding vector u_{j_2}, that is, $\tilde{U}_{S_2} = \tilde{U}_{S_1} \cup \{u_{j_2}\}$. The next furthest sample is selected with the same procedure. The algorithm stops when the number of selected samples is $d+1$. It will be proved theoretically in Theorem 1 that the samples selected in Algorithm 1 are all vertices of conv(P).

In Algorithm 1, an efficient way to evaluate $d_s(u_i, \mathcal{L}(\tilde{U}_{S_t}))$ is needed. In fact, from (2) it can be seen that we only need to compute Q_t^{-1} efficiently. Exploiting the incremental nature of Algorithm 1, we can update Q_t^{-1} recursively. Using the matrix inversion lemma, after adding a new sample, Q_{t+1}^{-1} can be updated as

$$Q_{t+1}^{-1} = \begin{bmatrix} & & 0 \\ & Q_t^{-1} & \vdots \\ & & 0 \\ 0 & 0 \cdots & 0 \end{bmatrix} + \frac{1}{\Delta} \begin{bmatrix} \alpha^* \\ -1 \end{bmatrix} \begin{bmatrix} \alpha^* & -1 \end{bmatrix} \tag{5}$$

where $\Delta = d_s(u_{i_0}, \mathcal{L}(\tilde{U}_{S_t}))$ and $\alpha^* = \arg\min_\alpha d_s(u_{i_0}, \mathcal{L}(\tilde{U}_{S_t}))$.

It can be seen that Algorithm 1 works only when $rank(\mathcal{L}(\tilde{U})) = d$. In this case, Q_t is invertible in the tth iteration for $t = 1, \ldots, d$, where $Q_t = U_{S_t}^\top U_{S_t}$ with $U_{S_t} = [u_{j_1}, \ldots, u_{j_t}]$. For the case when $rank(\mathcal{L}(\tilde{U})) < d$, an alternative approach is presented in Section III-F.

D. Data Partition Within One Class

In this section, we present a method to divide samples within one class into several parts such that the number of the samples in each part is no more than a given bound.

By applying Algorithm 1, we obtain $d+1$ samples, denoted by $\{x_i\}_{i=1}^{d+1}$, which constitute a d-simplex in $\mathbb{R}^d$. The d-simplex has $d+1$ facets. By using the vertices of the $d+1$ facets we can divide the samples into $d+1$ parts

$$P_i = \Big\{ x | x \in P, (x - \bar{x}) = \sum_{k=1, k\neq i}^{d+1} \alpha_k (x_k - \bar{x}), \sum_{k=1, k\neq i}^{d+1} \alpha_k \geq 1, \alpha_k \geq 0 \Big\}, \quad i = 1, \ldots, d+1 \tag{6}$$

where $\bar{x}$ is the mean of the $d+1$ samples selected in Algorithm 1, that is

$$\bar{x} = \frac{1}{d+1} \sum_{i=1}^{d+1} x_i \tag{7}$$

and $P_i, i = 1, \ldots, d+1$, are the $d+1$ divided parts. It can be seen that each facet of the d-simplex together with $\bar{x}$ determines a part, and the facet is called the corresponding facet of the part. From Theorem 2, $\bigcup_{i=1}^{d+1} P_i$ is the set whose samples are outside of the conv($\{x_i\}_{i=1}^{d+1}$).

Using the above approach, if the number of samples in one part is still very large, we need to further divide this part into more smaller parts in the following way. Suppose the number of the samples in P_{i_0} is larger than a threshold and the samples which constitute the corresponding facet of the part P_{i_0} is $\{x_{i_0}^{(k)}\}_{k=1}^d$. Denote by $x_{i_0}^{d+1}$ the sample in P_{i_0} which is the furthest to the hyperplane formed by $\{x_{i_0}^{(k)}\}_{k=1}^d$. Then the samples in $\{x_{i_0}^{(k)}\}_{k=1}^{d+1}$ constitute a d-simplex. The newly generated d facets of the d-simplex together with $\bar{x}$ defined by (7) can further divide P_{i_0} into d new smaller parts via (6). As an example, we consider Fig. 1. The number of samples in part $\mathcal{C}$ is larger than a threshold, and $\{x_1, x_2\}$ is the set of samples which constitute the corresponding edge (facet) of $\mathcal{C}$. It is seen that the sample x_4 is the furthest one to the straight line passing through x_1 and x_2. The set $\{x_1, x_2, x_4\}$ constitutes

Algorithm 3 Convex Hull Vertices Selection (CHVS) Algorithm

Input: $P_{oc} = \{P_i\}_{i=1}^M$, $S_{oc} = \{S_i\}_{i=1}^M$ and S obtained by Algorithm 2. Input a positive integer m.

1: **for** $i = 1; i \leq M; i++$ **do**
2: Use (4) to compute $m_{\mathrm{d}}^i = \max_{x \in P_i} d_{\mathrm{c}}(x, \mathrm{conv}(S_i))$ and $x_i^* = \arg\max_{x \in P_i} d_{\mathrm{c}}(x, \mathrm{conv}(S_i))$.
3: **end for**
4: $m_{\mathrm{d}} = \max_{i \in \{1,\ldots,M\}} m_{\mathrm{d}}^i$.
5: **while** $|S| < m$ and $m_{\mathrm{d}} > 0$ **do**
6: Let $j_0 = \arg\max_{i \in \{1,\ldots,M\}} m_{\mathrm{d}}^i$, $S_{j_0} = S_{j_0} \cup \{x_{j_0}^*\}$ and $P_{j_0} = P_{j_0} \setminus \{x_{j_0}^*\}$.
7: Let $m_{\mathrm{d}}^{j_0} = \max_{x \in P_{j_0}} d_{\mathrm{c}}(x, \mathrm{conv}(S_{j_0}))$ and $x_{j_0}^* = \arg\max_{x \in P_{j_0}} d_{\mathrm{c}}(x, \mathrm{conv}(S_{j_0}))$.
8: Let $m_{\mathrm{d}} = \arg\min_{i \in \{1,\ldots,M\}} m_{\mathrm{d}}^i$.
9: **end while**

Output: S.

a triangle (2−simplex), and x_1x_4 and x_2x_4 are its two newly generated edges (facets), so O and the two new edges can divide part $\mathcal{C}$ into two new smaller parts: $\mathcal{C}_1$ and $\mathcal{C}_2$. Using the above method, we can divide samples within one class into several parts, so that the number of samples in each part is no more than a given bound.

Algorithm 2 gives a detailed description on how to divide samples in one class. In Algorithm 2, P_{oc} denotes the set of the divided parts, P_i is the ith part of P_{oc}, $x_i^{(j)}$ represents the jth vertex of the corresponding facet of P_i, and S_i represents the set of the vertices of the corresponding facet of P_i, $B_i(i = 1, \ldots, d)$ are the newly divided parts of an old part which contains more than l_0 samples. Here, l_0 is a given integer (served as a threshold), and $E_i(i = 1, \ldots, d)$ are the sets of the vertices of the newly generated facets.

E. Convex Hull Vertices Selection (CHVS) Algorithm

In this section, the convex hull vertices selection algorithm is presented in Algorithm 3, in which m is the number of samples we aim to select. In each iteration of Algorithm 3, the sample which is the furthest to the sub-convex hull of its part is selected (Step 6). This guarantees that the samples selected by Algorithm 3 keep the greatest amount of information of the convex hull.

Remark 2: m is a key parameter that makes a compromise between the computational precision and the complexity. We will give an adaptive selection method for m in Section V-A.

F. Dimension Conversion

In real-world applications, we are often faced with samples in different classes that cannot be linearly separated in the input space $\mathbb{R}^d$. A common strategy is to map the original samples into a high dimensional feature space using the kernel trick where their images in the feature space are likely to be linearly separable. Since our algorithm is based on a finite dimensional space, we need to convert samples in the feature space into a finite dimensional space. Such a dimension conversion method is stated in Algorithm 4. Moreover, from Theorem 3 we can conclude that vertices of a convex hull in the input space correspond to the vertices of the convex hull in the converted finite dimensional space. Therefore, such a dimension conversion process will not affect the final vertices selection results.

In Algorithm 4, Δ is the distance between the two samples selected in Step 1 and ϵ is a small threshold. If the maximal distance between the samples and the current subspace is no more than $\Delta\epsilon$, we say that all samples are in the current subspace. From Theorem 3 we know that the columns of U^z can be regarded as an orthonormal basis of this current subspace. Then we establish a new coordinate system in the subspace by using the basis, and the dimension of the new coordinate system is finite. In Algorithm 4, d_z is the dimension of the output data set $\{z_i\}_{i=1}^n$. The case for $rank(\mathcal{L}(\tilde{U})) < d$ mentioned in Subsection III-C can also be solved by setting the kernel function to be the linear kernel in Algorithm 4.

Algorithm 4: Dimension Conversion

Input: $P = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$, kernel function and parameter ϵ.

1: Randomly select $x_0 \in P$. Let $x_{j_0} = \arg\max_{x \in P} \|\varphi(x_0) - \varphi(x)\|$, and $x_{j_1} = \arg\max_{x \in P} \|\varphi(x_{j_0}) - \varphi(x)\|$. Initialize $S = \{\varphi(x_{j_0}), \varphi(x_{j_1})\}$.

2: Let $\tilde{U} = \{u_i | u_i = \varphi(x_i) - \varphi(x_{j_0})\}_{i=1}^n$, $\tilde{U}_S = \{u_{j_1}\}$, and $\Delta = \|u_{j_1}\|$.

3: Use (8) to compute $m_{\mathrm{d}} = \max_{u_i \in \tilde{U} \setminus \tilde{U}_S} d_{\mathrm{s}}(u_i, \mathcal{L}(\tilde{U}_S))$ and $j_2 = \arg\max_{u_i \in \tilde{U} \setminus \tilde{U}_S} d_{\mathrm{s}}(u_i, \mathcal{L}(\tilde{U}_S))$. Let $t = 2$.

4: **while** $m_{\mathrm{d}} > \Delta\epsilon$ **do**

5: Let $\tilde{U}_S = \tilde{U}_S \cup \{u_{j_t}\}$, $S = S \cup \{\varphi(x_{j_t})\}$ and $t = t + 1$.

6: Let $m_{\mathrm{d}} = \max_{u_i \in \tilde{U} \setminus \tilde{U}_S} d_{\mathrm{s}}(u_i, \mathcal{L}(\tilde{U}_S))$ and $j_t = \arg\max_{u_i \in \tilde{U} \setminus \tilde{U}_S} d_{\mathrm{s}}(u_i, \mathcal{L}(\tilde{U}_S))$.

7: **end while**

8: Let $d_z = t - 1$, $I_S = \{j_t\}_{t=1}^{d_z}$, $\tilde{z}_{I_S} = eye(|I_S|)$ and $\tilde{z}_{j_0} = \mathbf{0}$.

9: Let $I_N = \{1, \ldots, n\} \setminus (I_S \cup \{j_0\})$ and $\tilde{z}_i = \arg\min_{\alpha} \|u_i - \mathcal{L}(\tilde{U}_S)\|$ for $i \in I_N$.

10: Let $[K^u]_{pq} = \langle u_{j_p}, u_{j_g} \rangle$, where $p, q \in \{1, \ldots, d_z\}$. Suppose $K^u = V \Lambda V^\top$, where V is a orthogonal matrix and Λ is a diagonal matrix.

11: Let $T = V\Lambda^{-\frac{1}{2}}$ and $z_i = T^{-1}\tilde{z}_i$ for $i = 1, \ldots, n$.

Output: $Z = \{z_i\}_{i=1}^n$.

Remark 3: In Algorithm 4, $\psi(\cdot)$ is only used to denote the feature mapping associated with the kernel function $k(\cdot, \cdot)$ (that is, $k(x, y) = \psi(x)^\top \psi(y)$) and is not needed in all computations. In fact, all computations involving $\psi(\cdot)$ can be converted into the computation involving the kernel function $k(\cdot, \cdot)$. For example, $d_{\mathrm{s}}(u_i, \mathcal{L}(\tilde{U}_S))$ can be computed as

$$\begin{aligned} d_{\mathrm{s}}(u_i, \mathcal{L}(\tilde{U}_S)) &= \min_{\alpha} \|u_i - \sum_{\tau=1}^{t} \alpha_\tau u_{j_\tau}\| \\ &= \min_{\alpha} \sqrt{u_i^\top u_i - 2(c^u)^\top \alpha + \alpha^\top K^u \alpha} \end{aligned} \quad (8)$$

where $\|\cdot\|$ denotes the ℓ_2 norm and $\alpha = (\alpha_1, \ldots, \alpha_t)^\top$. In (8), since $u_i = \psi(x_i) - \psi(x_{j_0})$, we have $u_i^\top u_i = k(x_i, x_i) - k(x_{j_0}, x_i) - k(x_i, x_{j_0}) + k(x_{j_0}, x_{j_0})$, $K^u = [\tilde{k}_{pq}]_{t \times t}$ with $\tilde{k}_{pq} = u_{j_p}^\top u_{j_q} = k(x_{j_p}, x_{j_q}) - k(x_{j_0}, x_{j_p}) - k(x_{j_0}, x_{j_q}) + k(x_{j_0}, x_{j_0})$ for $p, q = 1, \ldots, t$, and $c^u = (c_1^u, \ldots, c_t^u)^\top$ with $c_\tau^u = k(x_i, x_{j_\tau}) - k(x_{j_0}, x_{j_\tau}) - k(x_{j_0}, x_i) + k(x_{j_0}, x_{j_0})$ for $\tau = 1, \ldots, t$. Thus, we only need to know the kernel functions in order to implement Algorithm 4.

G. Computational Complexity

Before giving the complexity of Algorithms 1–4, we first introduce the symbols used below. n is the number of samples, d is the dimension of samples in the original space, d_z is the dimension of z_i in Algorithm 4, and m is the number of selected vertex samples. We first analyze the complexity of

computing $d_s(u, \mathcal{L}(\tilde{U}))$ and $d_c(x, \text{conv}(P))$, which are used frequently in Algorithms 1–4.

1) Complexity of computing $d_s(u, \mathcal{L}(\tilde{U}))$: Define $t = |\tilde{U}|$. It follows from (1) that computing c takes $O(td)$ operations. Since Q^{-1} is updated recursively, it is seen from (5) that the complexity of computing Q^{-1} is $O(t^2)$. The complexity of computing α^* is $O(t^2)$ when Q^{-1} is known. Note that $d_s = \sqrt{u^\top u - 2c^\top \alpha^* + \alpha^{*\top} Q \alpha^*} = \sqrt{u^\top u - c^\top \alpha^*}$ since $c = Q\alpha^*$ and Q is symmetric. Thus, computing d_s takes $O(d + t)$ operations in the case when α^* and c are calculated. Hence, the total complexity of computing d_s is $O(td + t^2 + d + t) = O(td)$ since $t \leq d$.
2) Complexity of computing $d_c(x, \text{conv}(S_i))$: Computing c and Q in (3) takes $O(|S_i|d)$ and $O(|S_i|^2 d)$ operations respectively. The complexity of solving the quadratic problem (3) is $O(|S_i|^3)$. Computing $d_c = \sqrt{x^\top x - 2c^\top \alpha^* + \alpha^{*\top} Q \alpha^*}$ takes $O(d + |S_i| + |S_i|^2)$ operations in the case that α^* and c have been calculated. Hence, the total complexity of computing d_s is $O(|S_i|d + |S_i|^2 d + |S_i|^3 + d + |S_i| + |S_i|^2) = O(|S_i|^2 d + |S_i|^3)$.

We can now give the complexity of the four algorithms.

1) Complexity of Algorithm 1: In Step 1, selecting the two samples takes $O(nd)$ operations. In Step 2, the complexity of computing $\tilde{U}$ is $O(nd)$. In Step 4, computing $\max_{i \in \{i | u_i \in \tilde{U} \setminus \tilde{U}_{S_t}\}} d_s(u_i, \mathcal{L}(\tilde{U}_{S_t}))$ takes at most $O(n(td))$ operations in the tth iteration. Hence, the total complexity of Algorithm 1 is $O(nd + nd + \sum_{t=1}^{d} n(td)) = O(nd^3)$.

2) Complexity of Algorithm 2: Algorithm 2 is a method of dividing samples within one class into several parts such that the number of samples in each part is no more than a given number l_0. Step 6 is the key step in Algorithm 2. It divides the set $P_{I(k)}$, whose number of samples is larger than l_0, into d new smaller parts $(B_i, i = 1, \ldots, d)$.

In Step 5, solving the quadratic problem $\min_{\sum_{i=1}^{d} \alpha_i} \|x - \sum_{i=1}^{d} \alpha_i x_{I(k)}^{(i)}\|^2$ takes $O(d^3)$ operations. Then the complexity of Step 5 is $O(|P_{I(k)}|d^3)$. In Step 6, the complexity of computing W_i^{-1} and β is $O(d^3)$ and $O(d^2)$ respectively, so computing B_i takes $O(d^3 + |P_{I(k)}|d^2)$. Hence, the complexity of Step 6 is $O(d^4 + |P_{I(k)}|d^3)$. From Steps 4–8, the complexity is $O(\sum_{k=1}^{N}(d^4 + |P_{I(k)}|d^3))) \leq O(Nd^4 + nd^3)$ since $\bigcup_{k=1}^{N} P_{I(k)} \subseteq P$, where N is the number of sets in which the number of samples exceeds l_0. If the distribution of samples is uniform, then there are at most $\lceil \log_d(n/l_0) \rceil$ repetitions from Steps 3–12. The complexity of Algorithm 2 is thus at most $O(d^4(n/l_0) + \log_d(n/l_0) * nd^3)$.

3) Complexity of Algorithm 3: If the distribution of samples is uniform, then $M \approx O(n/l_0)$. In Step 2, the complexity is $O(|P_i|(|S_i|^2 d + |S_i|^3)) = O(|P_i|(d^3))$ since $|S_i| = d$. Thus, it takes $O(\sum_{i=1}^{M} |P_i|(d^3)) \leq O(nd^3)$ operations from Steps 1–3 since $P \supseteq \bigcup_{i=1}^{M} P_i$. In Step 7, computing $d_c(x, \text{conv}(S_{j_0}))$ takes $O(|S_{j_0}|^2 d + |S_{j_0}|^3)$ operations. Suppose the number of vertices selected in Algorithm 2 is n_2. Then it takes at most $O((|S| - (d+1) - n_2) l_0 (l_0^3 + l_0^2 d)) \leq O(m(l_0^4 + l_0^3 d))$ operations from Steps 5–8 since $|S_{j_0}| \leq l_0$. Hence, the complexity of Algorithm 3 is at most $O(m(l_0^3 d + l_0^4) + nd^3)$.

4) Complexity of Algorithm 4: Before analyzing the complexity of Algorithm 4, we assume the complexity of computing the value of a kernel function $k(\cdot, \cdot)$ is κ. Thus, computing the distance $\|\varphi(x') - \varphi(x'')\|^2 = k(x', x') + k(x'', x'') - 2k(x', x'')$ takes $O(\kappa)$ operations, so the complexity of Step 1 is $O(n\kappa)$. From (8), we know that the complexity of computing c^u is $O(t\kappa)$. Computing $(K^u)^{-1}$ takes $O(t^2)$ operations since $(K^u)^{-1}$ can be updated recursively from (5). The complexity of computing α^* is $O(t^2)$ when $(K^u)^{-1}$ is computed. Thus, the complexity of Step 6 in the tth iteration is $O(t\kappa + t^2 + \kappa + t) = O(t\kappa + t^2)$. We can see that it takes $O(\sum_{t=1}^{d_z}(n(t\kappa + t^2))) = O(n(\kappa d_z^2 + d_z^3))$ operations from Steps 3–7. The optimization problem in Step 9 has been solved in Step 6 in the last loop from Steps 4–7. In Step 10, computing K^u and its singular value decomposition (SVD) takes $O(d_z^2 \kappa)$ and $O(d_z^3)$ operations respectively. The complexity of Step 11 is $O(nd_z^2)$. Hence, the complexity of Algorithm 4 is $O(n\kappa + n\kappa d_z^2 + d_z^3 + d_z^2 \kappa + d_z^3 + nd_z^2) = O(n\kappa d_z^2 + d_z^3)$.

In training a classifier with n samples, the computational complexity of our method includes two parts: the sample selection complexity and the training complexity. The sample selection complexity is at most $O(nd^4)$, which is analyzed in the above paragraphs, and the training complexity is $O(m\bar{m}_{sv})$ [49] since we train the classifier with the selected samples by using the SMO algorithm, where m is the number of the selected samples and $\bar{m}_{sv}$ is the average number of the candidate support vectors during iterations of the training process. Hence, our method takes at most $O(nd^4 + m\bar{m}_{sv})$ operations in total. In each iteration of the Huller algorithm, the complexity of updating the coefficients is $O(n_{sv})$, where n_{sv} is the number of the candidate support vectors at the current iteration. Let $\bar{n}_{sv}$ be the average number of the candidate support vectors during iterations. Then the Huller algorithm takes about $O(n\bar{n}_{sv})$ operations after a single epoch. Since $m < n$ and $\bar{m}_{sv} \leq \bar{n}_{sv}$, then our method is more efficient than the Huller algorithm in the case when the dimensionality of data sets is not very high. However, for high dimensional data sets, the Huller algorithm is faster than ours. Hence, our algorithm can directly deal with large-scale data sets with not too high dimensionality. For data sets with high dimensionality, dimensionality reduction methods, such as locality preserving projections (LPP) [50], [51], locally linear embedding (LLE) [52], [53] and neighborhood preserving polynomial embedding (NPPE) [54], can be used as a preprocessing step to remove the redundant information. Then the dimensionality-reduced data can be dealt with by our method.

IV. Theoretical Analysis of the Proposed Algorithm

In this section, we present a detailed theoretical analysis of the proposed algorithm. Proofs of the main theorems are given in the appendix. Our main results in this section validate the proposed algorithms in Section III.

Theorem 1: Let $P = \{x_i\}_{i=1}^{n}$ be the set of samples in $\mathbb{R}^d$. Then we have the following conclusions.

1) At each iteration of Algorithm 1, if the number of samples in P which are the furthest to the current subspace

is at most two, then samples selected in Algorithm 1 are all vertices of conv(P), that is, if S_1 is the set of selected samples in Algorithm 1, then for any $x \in S_1$, $x \notin \text{conv}(P \setminus x)$.

2) Suppose S is a set of some vertices of conv(P). At each iteration, if S is spanned by adding the sample which is the furthest to conv(S) and, meanwhile, if there are no more than two furthest samples to conv(S) with the same distances to conv(S), then samples in S are all vertices of conv(P), that is, for any $x \in S$, $x \notin \text{conv}(P \setminus x)$.

Theorem 2: Let $x_0, x_1, \ldots, x_d$ be $d+1$ samples in $\mathbb{R}^d$, which are not coplanar, that is, $x_1 - x_0, \ldots, x_d - x_0$ constitute a basis of $\mathbb{R}^d$. Let Π be the hyperplane determined by $x_1, \ldots, x_d$. For any $x \in \mathbb{R}^d$, let $(x - x_0) = \sum_{i=1}^{d} \alpha_i (x_i - x_0)$. Then we have the following conclusions.

1) $\sum_{i=1}^{d} \alpha_i = 1$ if and only if x is on the hyperplane Π, that is, $x - x_1 = \sum_{i=2}^{d} \alpha_i (x_i - x_1)$.
2) $\sum_{i=1}^{d} \alpha_i < 1$ if and only if x_0 and x are at the same side of the hyperplane Π, that is, $\langle x_0 - \tilde{x}_0, x - \tilde{x}_0 \rangle > 0$ for the projection $\tilde{x}_0$ of x_0 onto Π.
3) $\sum_{i=1}^{d} \alpha_i > 1$ if and only if x_0 and x are at different sides of the hyperplane Π, that is, $\langle x_0 - \tilde{x}_0, x - \tilde{x}_0 \rangle < 0$ for the projection $\tilde{x}_0$ of x_0 onto Π.

Theorem 3: Let $\{\varphi(x_i)\}_{i=1}^{n}$ be the set of samples in the feature space and let $\{z_i\}_{i=1}^{n}$ be the set of samples obtained in Algorithm 4. Then we have the following three conclusions.

1) Let $U_S = [u_{j_1} \cdots u_{j_{d_z}}]$ and $U^z = U_S T$, where u_{j_t}, $t = 1, \ldots, d_z$, and T are obtained from Algorithm 4. Then $(U^z)^\top U^z = I$.
2) $\|z_i - z_j\|^2 = \|\varphi(x_i) - \varphi(x_j)\|^2$, $\forall i, j = 1, \ldots, n$.
3) Assume that I is the index set of vertices of conv($\{\varphi(x_i)\}_{i=1}^{n}$) and I' is the index set of vertices of conv($\{z_i\}_{i=1}^{n}$). Then $I = I'$.

Theorem 4: If the Gaussian kernel is used in Algorithm 4, then the dimension of the converted samples in the set Z is finite for any compact input domain.

Theorem 4 was proved in [55]. It guarantees that for any compact set of training samples, the dimension of the converted samples in the set Z obtained by Algorithm 4 is finite when we use the Gaussian kernel.

V. Online SVM Based on the Vertices Selection Algorithm

A. Vertices Selection (VS) Algorithm

By applying the CHVS algorithm on the samples of each class, the vertices selection (VS) algorithm is proposed. The diagram of VS algorithm is shown in Fig. 3.

In Fig. 3, N_{s+}^i and N_{s-}^i represent the numbers of samples that we need to select from the positive samples set P_+ and the negative samples set P_-, respectively, by using the CHVS algorithm (see the fourth box) in the ith iteration. $Train(P_s^{(i)})$ represents the fact that the sample set $P_s^{(i)}$, which is selected in the ith iteration, is used to train the SVM classifier. The classifier is denoted by Γ_i, and a_i is the classification rate of Γ_i on the testing samples. $a_\delta = [a_\delta^j]_{1\times 10}$ is an increment vector of the classification rate, where $a_\delta^j = a_{i-10+j} - a_{i-10}$. $a_\delta \le \varepsilon$ represents that all elements of

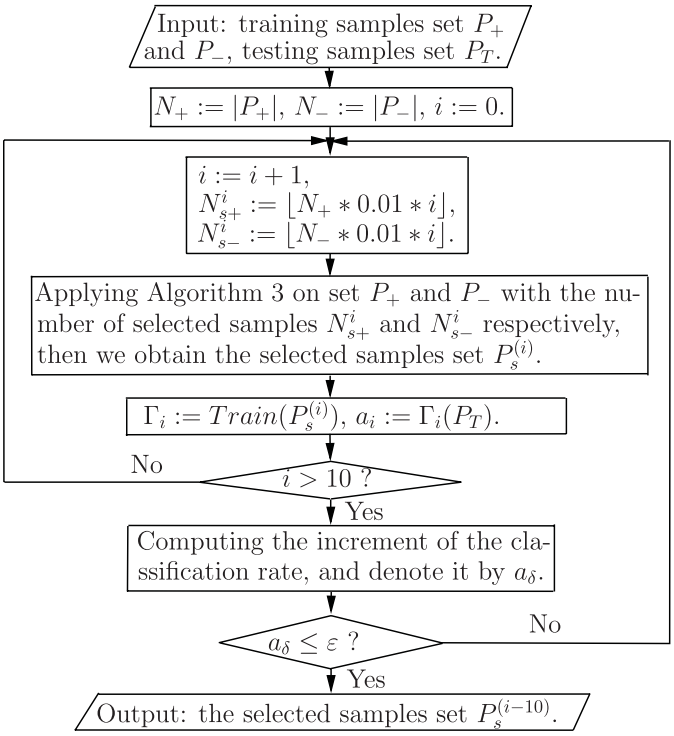


Fig. 3. Diagram of VS algorithm.

a_δ are not larger than ε, where ε is a small threshold. This means that the set $P_s^{(i-10)}$ of selected samples already has the greatest amount of information of the training samples. In our experiments, we always set $\varepsilon = 1/|P_T|$, where $|P_T|$ is the number of the testing samples.

As shown in Fig. 3, in each iteration, we add one percent of the training samples on the basis of the selected samples set in the last iteration. That is, in the i-iteration of VS algorithm, we select $\lfloor N_+ * i * 0.01 \rfloor$ positive samples and $\lfloor N_- * i * 0.01 \rfloor$ negative samples, respectively, by applying the CHVS algorithm to each class. We use these selected samples to train the SVM classifier. Generally speaking, the classification rate on testing samples increases with the number of selected samples. However, when the selected samples keep the greatest amount information of the convex hull, the classification rate will either not increase or only increase a little with the number of selected samples further increasing. This is the reason why we use the increment of the classification rate on testing samples as the stop criterion in Fig. 3. We do not further select samples if the increment of the classification rate within a small threshold ε. Hence, the number m of the selected samples in each class is adaptively determined via the increment of the classification rate on the testing samples.

B. Online SVM Based on VS (VS-OSVM) Algorithm

Based on the VS algorithm, an online updating of the SVM classifier, named VS-OSVM, is achieved. The online learning algorithm can be decomposed into four steps.

Step 1: Denote the current training samples set by P_{cur}, and it is initialized by all training samples.

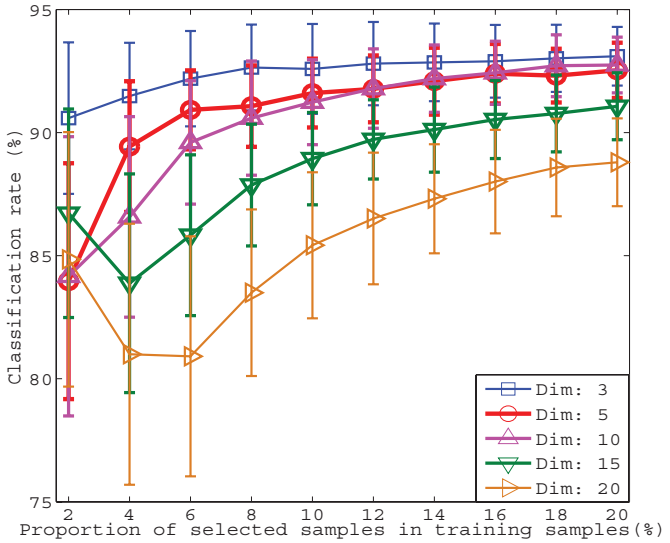


Fig. 4. Relationship between the selected convex hull vertices and the dimensionality of samples.

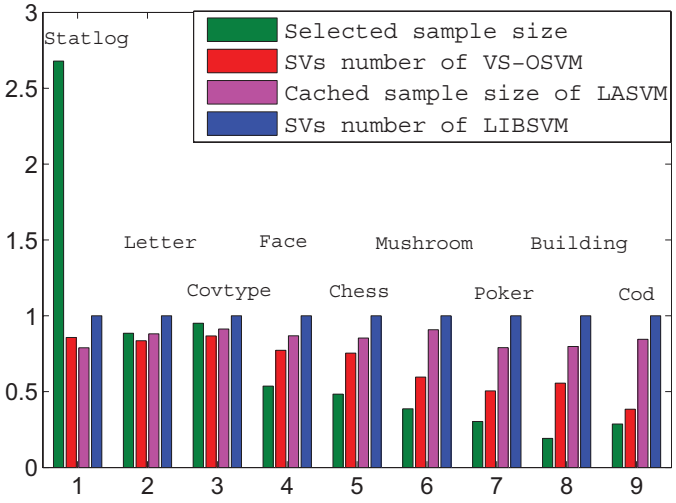


Fig. 5. Relationship among the number of the selected samples in the offline step, the number of the support vectors of our method at the end, the cached sample size in LASVM, and the number of the support vectors of LIBSVM.

Step 2: By applying VS algorithm on set P_{cur}, we obtain the selected samples set P_{s} which has the greatest amount of information of the convex hulls of P_{cur}. The samples in set P_{s} are stored for the following online adjustment step. Let $P_{\text{cur}} = P_{\text{s}}$.

Step 3: When a new sample x_{new} arrives, we compute the value $f(x_{new})$, where f is the function implemented by the current classifier. If $y_{new} f(x_{new}) \leq (1+\lambda)$, let $P_{\text{cur}} = P_{\text{cur}} \cup \{x_{new}\}$. Then the current SVM classifier will be updated by using P_{cur}. Otherwise, the current SVM classifier will not be updated.

Step 4: If $|P_{\text{cur}}| > M$, where M is a large integer, go to Step 2. Otherwise, go to Step 3.

We apply Step 1 only in the case when the number of the current training samples set P_{cur}, which consists of the selected samples in Step 1 and all newly arrived samples, exceeds a large number M. We execute periodically Step 1 off-line in order to reduce the number of the current training samples. This makes the online training with new arrived samples possible. In Step 2, $1+\lambda$ is a threshold on the decision value. We use the samples x that satisfy the condition $yf(x) \leq 1+\lambda$, where y is the label of x, since they are near the current hyperplane and may become support vectors with a high probability after updating the classifier. In the experiment, we set $\lambda = 0.1$.

VI. Experiments

In this section, we first conduct an artificial experiment that shows the relationship between the selected convex hull vertices and the dimensionality of samples. The overall description of the experiment design is then described. Finally, experiments on eight benchmark data sets are conducted to demonstrate the effectiveness of the proposed algorithm.

A. Relationship Between Convex Hull Vertices and Dimensionality of Samples

A d-dimensional data set of 1000 samples (500 samples of each class) are generated from d-dimensional Gaussian distributions, where $d = 3, 5, 10, 15, 20$. We decompose each data set into the training samples set and the testing samples set with a fixed proportion 7 : 3. These two sets are independent of each other. The CHVS algorithm is used to select samples from each class. In Fig. 4, we plot the proportion of selected samples in the training samples (Lateral axis) to the average testing classification rates of the SVM classifier trained with these selected samples (longitudinal axis) for different dimensions.

From Fig. 4, we have the following two conclusions.

1) If the classification rate is fixed, the number of selected convex hull vertices which are necessary for classification rate increases with the dimensionality increasing.
2) If the number of selected convex hull vertices is fixed, the classification rate decreases as the dimensionality increases.

These are reasonable since the higher the dimensionality of the data samples is, the more vertex samples the convex hull of the data set has. Thus, we should select more samples to keep the greatest amount of information of the convex hull.

B. Overall Description of the Experiment Design

All experiments are performed on a Desktop PC with Intel DuoCore 1.7 GHz CPU, 2G Ram and Windows XP operating system. Programs and codes are implemented in MATLAB. The procedure of each experiment can be decomposed into four steps, which are described below.

Step 1: Generating data. We decompose each data set into three parts: training samples, expanded samples and testing samples, which are independent of each other. Ten independent sets of training, expanded and testing samples are generated by running ten

TABLE I

OVERALL DESCRIPTION OF DATA SETS USED IN SECTION VI-B AND RESULTS OF SAMPLES SELECTION BY VS ALGORITHM. C IS THE PENALTY PARAMETER, AND γ IS THE PARAMETERS OF THE GAUSSIAN KERNEL $k(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2}$. DIM.: THE DIMENSIONALITY OF DATA USED IN THE EXPERIMENTS. TR#: NUMBER OF TRAINING SAMPLES. EX#: NUMBER OF EXPANDED SAMPLES WHICH ARE USED TO UPDATE THE CLASSIFIERS. TE#: NUMBER OF TESTING SAMPLES. SE#: NUMBER OF SELECTED SAMPLES BY VS. PR: THE PROPORTION OF THE SELECTED SAMPLES

Data Sets	Dim.	C	γ	TR#	EX#	TE#	SE#	PR (%)
UCI-Statlog	36	500	0.1142	1080	431	648	75±36	6.8±3.39
UCI-Letter	16	1	0.6576	778	310	467	162±59	20.7±7.60
UCI-Covertype	54	10	1480	10000	4000	6000	810±179	8.1±1.79
CBCL Face	361	10	204.1	3489	1394	2094	818±162	23.4±4.65
UCI-Chess	6	10	0.6127	2882	1152	1729	1049±69	36.4±2.41
UCI-Mushroom	112	1	26.349	4062	1624	2438	208±30	5.1±0.74
UCI-Pokerhand	10	20	0.808	1500	600	900	403±45	26.9±2.98
Building	50	10	0.016	900	360	540	146±47	16.3±5.22
Cod-rna	8	20	-	165576	66231	99345	7728±853	4.67±0.52

random divisions on data samples in each set with a fixed proportion 5:2:3. After the generating process, we have ten independent sets of training/expanded/testing samples. The next three steps are repeated on each of these sets.

Step 2: Sample Selection Process. We use two methods to select samples from the training samples: the random selection (RS) method which randomly selects samples from the set of training samples, and the vertices selection (VS) algorithm which selects samples by applying the CHVS algorithm on the samples of each class.

Step 3: Updating Process. We use the expanded samples to update the classifiers with four methods: VS-OSVM, RS-OSVM (the online version of SVM based on the RS method), LIBSVM (updating the classifier in batch mode) and LASVM (online setup with finishing step). We first equally divide the expanded samples into ten subsets and then update the classifiers with one subset at a time. After the classifiers are updated each time, we record the time cost of the update process.

Step 4: Testing Process. After the VS-OSVM, RS-OSVM, LIBSVM and LASVM classifiers are updated using all expanded samples, we apply them to classification tasks on the testing samples.

C. Experiments on Real-World Data

Fast and accurate online classifier is an important component for detection or classification tasks in real-time pattern classification systems such as pedestrian/building detection in driver assistant systems (DAS) and zip-code recognition in accommodation distribution systems. In this section, eight benchmark real-world data sets are used to evaluate the performance (both in speed and classification rate) of the proposed algorithm and demonstrate the validity of the theoretical analysis given in Section IV. All experiments use the Gaussian kernel to train the nonlinear SVM classifiers except for the experiment on the Cod-rna database which uses the linear kernel. The parameters are obtained by using five-fold cross validation, and the values are shown in Table I. In all tables, the form of mean value ± standard deviation is used to report the experimental results. Details of the experiments are presented below.

1) Description of the Data Sets: An overall description of the data sets is presented in Table I.

The UCI Statlog (Landset Satellite) database [56] contains images of eight different types of soil, and the attributes of samples are composed of multi-spectral values of pixels in 3×3 neighborhoods in a satellite image. We use the classes of "red soil" and "damp grey soil" in our experiment. The UCI Letter database [56], which contains 26 capital letters in English alphabet with 20 different fonts. Each sample in the database has 16 primitive numerical attributes in the range of integer values from 0 to 15, which record the statistical moments and edge counts. The experiment is conducted to classify letters "A" and "B". The UCI Covertype database [56], contains eight different types of forest covertype and the experiment is designed to classify the forest cover types "Douglas-fir" and "Krummholz". The `CBCL Face` database [57] has 6977 images of size 19×19 (2429 faces and 4548 nonfaces). The classification task aims to determine whether or not an image is a face image. The UCI Chess Endgame database [56] contains 18 classes which represent the depth-of-win of chess endgame. The depth-of-win "twelve" and "fifteen" are used in our experiment. The UCI Mushroom database [56] includes hypothetical samples of 23 species of gilled mushrooms in Agaricus and Lepiota Family. The database contains 8124 instances (with 4208 edible mushrooms and 3916 poisonous mushrooms). The building data base is built up by our lab for vision-based perceptual module in DAS. There are 900 building samples and 900 non-building samples in the data set. The resolution of the original images is 32×16, and we use the locality preserving projections (LPP) [50], [51] algorithm to reduce the dimensionality. There are 50 attributes preserved in our experiment. The UCI Poker hand database [56] contains ten poker hand classes. Each sample in the database is a hand consisting of five playing cards drawn from a standard deck of 52, and each card is described using two attributes (suit and rank). There are totally ten predictive attributes for each sample. The experiment is conducted to classify "Straight" and "Full house". Cod-rna is a bio-medical data set introduced

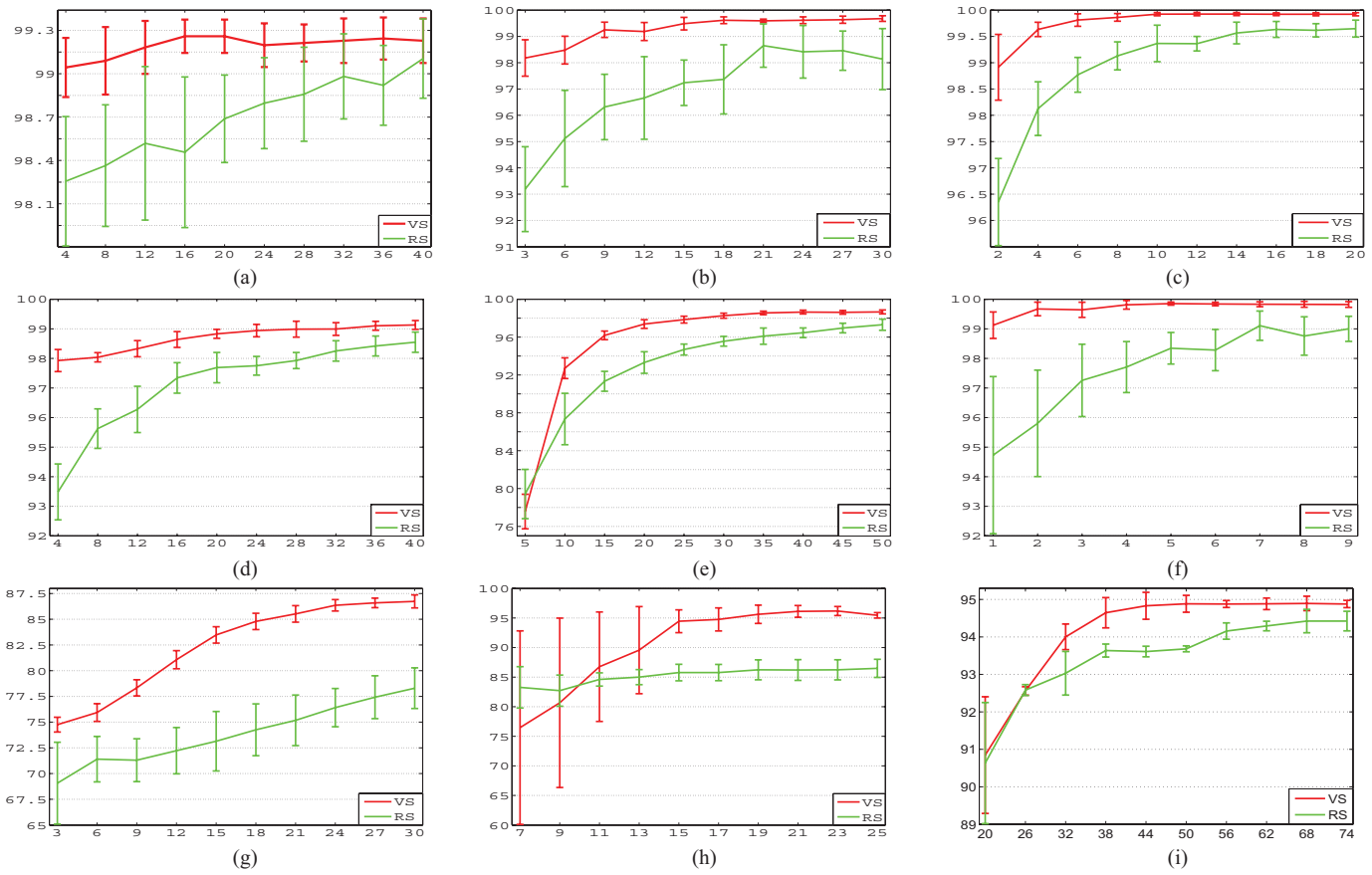


Fig. 6. Classification rate of SVM trained with samples selected from training samples by Algorithm 3 (red line) and the classification rate of SVM trained with randomly selected samples (green line). Lateral axis represents the proportion of selected samples in training samples (in % except for Cod-rna which is in ‰) and longitudinal axis represents the classification rate. (a) UCI-Statlog. (b) UCI-Letter. (c) UCI-Covertype. (d) CBCL Face. (e) UCI-Chess. (f) UCI-Mushroom. (g) UCI-Pokerhand. (h) Building. (i) Cod-rna.

TABLE II

CLASSIFICATION RATES OF LIBSVM, LASVM, RS-OSVM, AND VS-OSVM. THE MEAN RATE TOGETHER WITH STANDARD DEVIATION IS REPORTED. THE BEST CLASSIFICATION RESULTS ARE MARKED IN BOLDFACE

Data Sets	LIBSVM	LASVM	RS-OSVM	VS-OSVM
UCI-Statlog	99.259± 0.151	98.750± 0.257	98.673± 0.220	**99.275± 0.170**
UCI-Letter	**99.893± 0.107**	99.427± 0.150	98.544± 0.937	**99.893± 0.107**
UCI-Covertype	**99.972± 0.018**	99.935± 0.042	99.818± 0.032	99.970± 0.019
CBCL Face	**99.632± 0.132**	99.427± 0.151	98.997± 0.273	99.618± 0.128
UCI-Chess	**99.248± 0.215**	99.076± 0.402	98.294± 0.232	**99.248± 0.197**
UCI-Mushroom	**99.889± 0.026**	99.861± 0.085	94.430± 1.206	**99.889± 0.026**
UCI-Poker hand	**88.500± 0.4527**	87.733± 0.604	84.989± 0.688	88.321± 0.483
Building	**96.358± 0.698**	93.012± 0.887	89.835± 0.973	96.214± 1.048
Cod-rna	**95.222± 0.021**	95.219± 0.015	95.144± 0.023	95.215± 0.019

in [58], which includes 331152 samples with eight attributes. The classification task is to detect whether or not a sample is RNA of cod fish.

2) Experimental Results: In the samples selection process, the average classification rates measured on testing samples, as a function of the number of selected samples with the proposed samples selection method (red line) and the random selection method (green line), are shown in Fig. 6. It can be seen that the samples selected by our method is more effective than the samples selected by the RS method.

The relationship among the number of the selected samples in the offline step, the number of the support vectors of our method at the end, the cached sample size in LASVM and the number of the support vectors of LIBSVM is shown in Fig. 5. All these quantities for each data set are regularized by dividing by the number of the support vectors of LIBSVM. From Fig. 5, it can be seen that the number of the support vectors of LIBSVM is largest for all data sets (except for UCI Statlog) since it trains the classifier with all the existing samples. The number of the selected samples in the off-line step and the number of the support vectors of our method at the end are smaller than the cached sample size in LASVM and the number of the support vectors of LIBSVM for all data sets (except for UCI Statlog and UCI Covertype). This is because

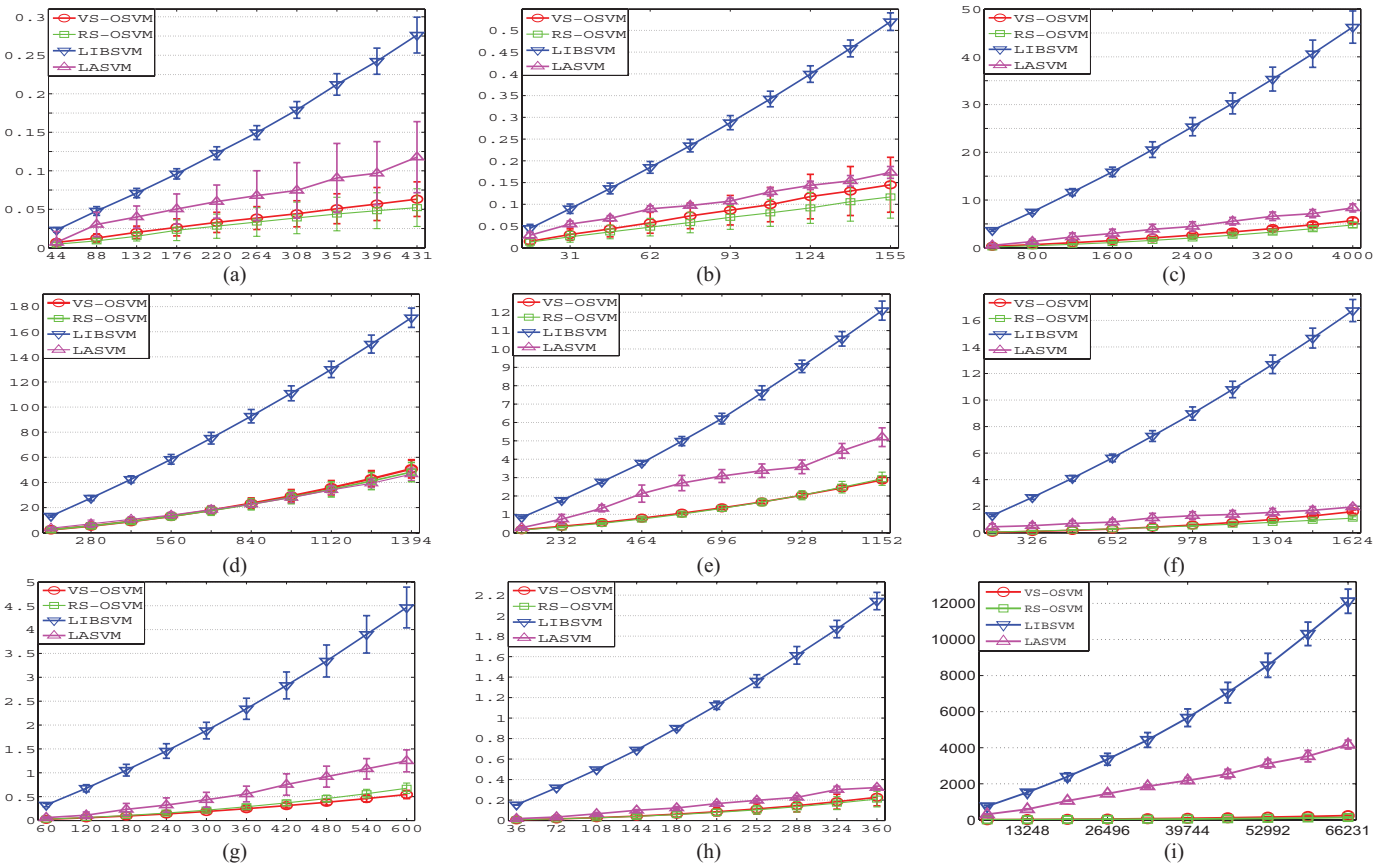


Fig. 7. Time cost for updating classifiers in the online step. Lateral axis represents the number of expanded samples and longitudinal axis represents the time cost (s). (a) UCI-Statlog. (b) UCI-Letter. (c) UCI-Covertype. (d) CBCL Face. (e) UCI-Chess. (f) UCI-Mushroom. (g) UCI-Pokerhand. (h) Building. (i) Cod-rna.

we do not select all the vertex samples but only the skeleton vertex samples which keep the greatest amount of information of the current training samples, and a large number of samples (at most 95.3%, as shown in Table I) are deleted in the off-line step. Then some negligible support vectors of the true final classifier are eliminated in the off-line step. In addition, we might also delete some support vectors of the final true classifier that are far away from the existing classifier margins in the online step since these support vector samples will possibly not help in improving the classification performance. Hence, the number of support vectors of our method at the end is smaller than that of LASVM and LIBSVM.

The smaller number of selected samples and support vectors in our method ensures that the online training of our method is faster than LASVM (except for `CBCL Face`), and much faster than LIBSVM, as shown in Fig 7. The time cost of VS-OSVM (red line) refers to the cost of updating the classifier by using the samples selected by the VS algorithm and the newly arrived samples in the online step, and does not contain the cost in the off-line step. The time cost of RS-OSVM (green line) refers to the cost of updating the classifier by using the randomly selected samples and the newly arrived samples in the online step. The blue line with legend "LIBSVM" represents the time cost without any sample selection process, and it is obtained by training the SVM classifier with all the training samples and the newly arrived samples via LIBSVM. The magenta line with "LASVM" represents the time cost of LASVM with online setup.

The classification rate of the four methods on the testing samples is shown in Table II. It can be seen that our method gives almost the same classification results as LIBSVM and outperforms RS-OSVM and LASVM (except for Cod-rna). This is because: 1) in the off-line step, we select the skeleton vertex samples which keep the greatest amount of information of the current training samples, and 2) in the process of the online step, we consider not only the new samples being closed to the current classifier margin, but also all the samples which have been involved in the previous updating process (these samples not only contain support vectors of the previous classifiers, support vectors and some non-support vectors of the current classifier and all the skeleton vertex samples selected in the off-line step, but will also become support vectors at the end with a high probability). Thus, our method rarely deletes important samples in the training process and retains the good generalization power of LIBSVM trained with all the samples. LASVM with online setup is only based on the current classifier to cache samples. Once the samples are kicked out from the cache in the REPROCESS step, they are discarded forever. LASVM with online setup would discard some non-negligible support vectors of the final true classifier in the REPROCESS step, and therefore its classification rate

is lower than that of our method. On the other hand, it should be pointed out that the nine data sets are linearly or nonlinearly separable and therefore, by using the kernel trick, the classification rate of our method and LIBSVM on these data sets is almost the same.

VII. Conclusion

This paper provide a new online SVM classifier based on the selection of vertices of the convex hull in each class. In the samples selection process, a small number of skeleton samples, which constitute the approximate convex hull of the current training samples, were selected for online learning. In the online classification process, the classifier was updated online based on the selected samples and newly arriving samples.

It was proved theoretically that the $d+1$ selected samples are all vertices of the convex hull. It guarantees that the samples selected by our approach retain the greatest amount of information of the current training samples. From the application point of view, the online adjustment of an SVM classifier based on the selected skeleton samples improves the classification rate very little but consumes much less time. Therefore, the proposed online SVM can be applied to various online classification tasks, such as pedestrian detection and visual tracking.

Experimental results on real-world data sets have validated the effectiveness of the proposed method.

We should remark that our method may not be directly applied to data sets with heavy noise since it is based on the vertices selection of the convex hull of the samples in each class. For heavily noised data sets, the involved training samples can be preprocessed by using de-noising methods (see [59], [60]) before samples selection in the off-line step, that is, we can add a preprocessing step at the beginning of Step 2 in the VS-OSVM algorithm. Recently, Geebelen *et. al.* [61] proposed a method for pruning the solutions of the SVM, which first removes the misclassified samples or flips their labels to make the training set separable and then formulates an approximate SVM solution with the upper-bounded weight vector for the modified separable training set. Inspired by this method, we can also remove the current misclassified samples or swap their labels before the samples selection step to reduce the heavy noise. However, we cannot eliminate the newly arrived noised data samples during the online step; these samples may cause the updated classifier to have some deviation from the final true classifier. How to eliminate the heavily noised data samples in the online step is still an intractable issue, and it will be one of our future works.

In Step 3 of the VS-OSVM algorithm, the classifier is updated only when there are newly arrived samples whose distances to the current classifier are within a given threshold, since we believed that these samples would become support vectors at the end with a very high probability. Hence, the initial data set is important for the training process of our classifier. If the initial data set cannot indicate the distribution of the whole data set, then the initial classifier trained with the selected samples may result in a large deviation from the final true classifier. Then some important support vectors of the final true classifier will be possibly deleted in the online updating process, and the generalization performance of our classifier will be reduced. Nevertheless, in each of our experiments, we randomly selected a half of the data samples as the initial data set, which can indicate the distribution of the whole data set to a great extent. Therefore, we obtained good experimental results. How to find a more effective method of selecting the initial data set may be an interesting issue, and it will be our future work.

Appendix

In this appendix, we prove Theorems 1–3 in Section IV. To this end, we need some lemmas.

Lemma 1: For $P=\{x_i\}_{i=1}^n \subset \mathbb{R}^d$ randomly select $x_{i_0} \in P$ and let $x_{j_0} = \arg\max_{x\in P} \|x_{i_0} - x\|$. Then x_{j_0} is a vertex of $\mathrm{conv}(P)$, that is, $x_{j_0} \notin \mathrm{conv}(P \setminus x_{j_0})$.

Proof: We prove the lemma by contradiction. Suppose the conclusion is not true, that is

$$x_{j_0} = \sum_{k=1,k\neq j_0}^{n} \alpha_k x_k, \qquad \sum_{k=1,k\neq j_0}^{n} \alpha_k = 1, \quad \alpha_k \geq 0. \tag{A1}$$

Let π be a hyperplane which is passing through the sample x_{j_0} and whose normal vector is $\overrightarrow{x_{j_0}x_{i_0}}$. Then

$$\langle x - x_{j_0}, x_{i_0} - x_{j_0}\rangle = 0 \quad \forall x \in \pi. \tag{A2}$$

Insert (A1) in (A2), we obtain

$$\sum_{k=1,k\neq j_0}^{n} \alpha_k \langle x - x_k, x_{i_0} - x_{j_0}\rangle = 0. \tag{A3}$$

Since $\alpha_k \geq 0$ for $k=1,\ldots,n$ with $k \neq j_0$, we need to consider two cases: 1) there exists $k_0 \neq j_0$ such that $\langle x - x_{k_0}, x_{i_0} - x_{j_0}\rangle = 0$, and 2) $\langle x - x_k, x_{i_0} - x_{j_0}\rangle \neq 0$ for $k=1,\ldots,n$ with $k \neq j_0$, so there exists $k_0 \neq j_0$ such that $\langle x - x_{k_0}, x_{i_0} - x_{j_0}\rangle > 0$.

From the definition of π, we have

$$\begin{aligned}\langle x - x_{k_0}, x_{i_0} - x_{j_0}\rangle &= \langle x - x_{j_0}, x_{i_0} - x_{j_0}\rangle \\ &\quad + \langle x_{j_0} - x_{k_0}, x_{i_0} - x_{j_0}\rangle \\ &= \langle x_{j_0} - x_{k_0}, x_{i_0} - x_{j_0}\rangle \quad \forall x \in \pi. \end{aligned} \tag{A4}$$

We first consider case (i): $\langle x - x_{k_0}, x_{i_0} - x_{j_0}\rangle = 0$. Then from (A4) it follows that:

$$\langle x_{j_0} - x_{k_0}, x_{i_0} - x_{j_0}\rangle = 0.$$

Thus, $\|x_{i_0} - x_{k_0}\| > \|x_{i_0} - x_{j_0}\|$. This contradicts to the definition of x_{j_0}.

We now consider case (ii): $\langle x - x_{k_0}, x_{i_0} - x_{j_0}\rangle > 0$. From (A4) it is known that $\langle x_{k_0} - x_{j_0}, x_{i_0} - x_{j_0}\rangle < 0$. Then

$$\begin{aligned}\|x_{i_0} - x_{j_0}\|^2 &= \langle x_{i_0} - x_{k_0} + x_{k_0} - x_{j_0}, x_{i_0} - x_{j_0}\rangle \\ &= \langle x_{i_0} - x_{k_0}, x_{i_0} - x_{j_0}\rangle + \langle x_{k_0} - x_{j_0}, x_{i_0} - x_{j_0}\rangle \\ &< \langle x_{i_0} - x_{k_0}, x_{i_0} - x_{j_0}\rangle \\ &\leq \|x_{i_0} - x_{k_0}\|\|x_{i_0} - x_{j_0}\|. \end{aligned}$$

Hence, $\|x_{i_0} - x_{j_0}\| < \|x_{i_0} - x_{k_0}\|$, which also contradicts to the definition of x_{j_0}. ■

From Lemma 1 it can be seen that x_{j_0} and x_{j_1} selected in Step 1 of Algorithm 1 are vertices of conv(P).

Lemma 2: For $x_1, x_2, \ldots, x_p, z \in \mathbb{R}^d$ suppose $z - x_1$ is not in the subspace $\mathcal{L}$ spanned by $\{x_i - x_1\}_{i=1}^p$. Let $\tilde{z} - x_1$ be the projection of $z - x_1$ on the subspace $\mathcal{L}$. Then we have

$$\langle\, x - z, \tilde{z} - z \,\rangle > 0 \quad \forall x - x_1 \in \mathcal{L}.$$

Proof: Since $\tilde{z} - x_1$ is the projection of $z - x_1$ on the subspace $\mathcal{L}$, we have

$$\langle\, x - x_1, \tilde{z} - z \,\rangle = 0 \quad \forall x - x_1 \in \mathcal{L}. \tag{A5}$$

Therefore

$$\begin{aligned}\langle\, x - z, \tilde{z} - z \,\rangle &= \langle\, \tilde{z} - z, \tilde{z} - z \,\rangle + \langle\, x - \tilde{z}, \tilde{z} - z \,\rangle \\ &= \|\tilde{z} - z\|^2 + \langle\, x - x_1, \tilde{z} - z \,\rangle \\ &\quad -\langle\, \tilde{z} - x_1, \tilde{z} - z \,\rangle \\ &= \|\tilde{z} - z\|^2 > 0 \end{aligned} \tag{A6}$$

where use has been made of the fact that, by (A5), the terms $\langle\, x - x_1, \tilde{z} - z \,\rangle$ and $\langle\, \tilde{z} - x_1, \tilde{z} - z \,\rangle$ are zero. ■

Lemma 3: For $x_1, x_2, \ldots, x_p, z_1, z_2 \in \mathbb{R}^d$ let $\mathcal{L}$ be the subspace spanned by $\{x_i - x_1\}_{i=1}^p$ and let $\tilde{z}_1 - x_1$, $\tilde{z}_2 - x_1$ be the projections of $z_1 - x_1$, $z_2 - x_1$ onto $\mathcal{L}$, respectively. Then:

1) $\langle\, \tilde{z}_1 - z_1, z_2 - z_1 \,\rangle > 0$ if $\|z_1 - \tilde{z}_1\| > \|z_2 - \tilde{z}_2\|$;
2) $\langle\, \tilde{z}_1 - z_1, z_2 - z_1 \,\rangle \geq 0$ if $\|z_1 - \tilde{z}_1\| \geq \|z_2 - \tilde{z}_2\|$.

Proof: The lemma is proved by contradiction. We only prove (i). (ii) can be proved similarly. Suppose the conclusion is not true, that is, $\langle\, \tilde{z}_1 - z_1, z_2 - z_1 \,\rangle \leq 0$. Since $\tilde{z}_2 - x_1 \in \mathcal{L}$, replace z and x with z_1 and $\tilde{z}_2$ in (A6), respectively, to get

$$\begin{aligned}\|\tilde{z}_1 - z_1\|^2 &= \langle\, \tilde{z}_1 - z_1, \tilde{z}_2 - z_1 \,\rangle \\ &= \langle\, \tilde{z}_1 - z_1, z_2 - z_1 \,\rangle + \langle\, \tilde{z}_1 - z_1, \tilde{z}_2 - z_2 \,\rangle \\ &\leq \langle\, \tilde{z}_1 - z_1, \tilde{z}_2 - z_2 \,\rangle \\ &\leq \|\tilde{z}_1 - z_1\|\|\tilde{z}_2 - z_2\| \end{aligned}$$

which means that $\|z_1 - \tilde{z}_1\| \leq \|z_2 - \tilde{z}_2\|$. This is a contradiction. ■

Lemma 4: Let $G \subset \mathbb{R}^d$ be a nonempty closed convex set. If $\tilde{z}$ is the projection of z onto G, then

$$\langle\, \tilde{z} - z, x - \tilde{z} \,\rangle \geq 0 \quad \forall x \in G.$$

Proof: The lemma is proved by contradiction. Suppose there exists a sample $\bar{x} \in G$ such that $\langle \tilde{z} - z, \bar{x} - \tilde{z} \rangle < 0$, that is, $\langle z - \tilde{z}, \bar{x} - \tilde{z} \rangle > 0$. We need to consider the following two cases:

Case 1: If $\|\bar{x} - \tilde{z}\|^2 \geq \langle z - \tilde{z}, \bar{x} - \tilde{z} \rangle$, then let

$$\begin{aligned}\lambda &= \frac{\langle z - \tilde{z}, \bar{x} - \tilde{z} \rangle}{\|\bar{x} - \tilde{z}\|^2} \\ e &= \lambda \bar{x} + (1 - \lambda)\tilde{z}. \end{aligned} \tag{A7}$$

It can be seen that $e \in G$ and $e - \tilde{z} = \lambda(\bar{x} - \tilde{z})$. On the other hand

$$\begin{aligned}\langle e - z, e - \tilde{z} \rangle &= \langle e - \tilde{z}, e - \tilde{z} \rangle + \langle \tilde{z} - z, e - \tilde{z} \rangle \\ &= \lambda \langle \tilde{z} - z, \bar{x} - \tilde{z} \rangle + \lambda^2 \|\bar{x} - \tilde{z}\|^2. \end{aligned} \tag{A8}$$

Substituting (A7) into (A8) gives $\langle e - z, e - \tilde{z} \rangle = 0$. Then we have $\|e - z\| < \|\tilde{z} - z\|$, which contradicts to the definition of $\tilde{z}$.

Case 2: If $\|\bar{x} - \tilde{z}\|^2 < \langle z - \tilde{z}, \bar{x} - \tilde{z} \rangle$, then

$$\begin{aligned} 0 &> \|\bar{x} - \tilde{z}\|^2 - \langle z - \tilde{z}, \bar{x} - \tilde{z} \rangle \\ &= \langle (\bar{x} - \tilde{z}) - (z - \tilde{z}), \bar{x} - \tilde{z} \rangle \\ &= \langle \bar{x} - z, \bar{x} - \tilde{z} \rangle. \end{aligned}$$

Thus we have

$$\begin{aligned}\|\bar{x} - z\|^2 &= \langle (\bar{x} - \tilde{z}) + (\tilde{z} - z), \bar{x} - z \rangle \\ &= \langle \bar{x} - \tilde{z}, \bar{x} - z \rangle + \langle \tilde{z} - z, \bar{x} - z \rangle \\ &< \langle \tilde{z} - z, \bar{x} - z \rangle \\ &\leq \|\tilde{z} - z\| \cdot \|\bar{x} - z\| \end{aligned} \tag{A9}$$

which implies that $\|\bar{x} - z\| < \|\tilde{z} - z\|$. This is also a contradiction to the definition of $\tilde{z}$. Hence, $\langle \tilde{z} - z, x - \tilde{z} \rangle \geq 0$ for all $x \in G$. ■

Lemma 5: Let $G \subset \mathbb{R}^d$ be a nonempty closed convex set and let $z \in \mathbb{R}^d \backslash G$. If $\tilde{z}$ is the projection of z onto G, then

$$\langle\, \tilde{z} - z, x - z \,\rangle > 0 \quad \forall x \in G.$$

Proof: Since $\tilde{z}$ is the projection of z onto G, we see by Lemma 4 that

$$\langle\, \tilde{z} - z, x - \tilde{z} \,\rangle \geq 0 \quad \forall x \in G.$$

Since z is not in G, it follows that for any $x \in G$

$$\begin{aligned}\langle \tilde{z} - z, x - z \rangle &= \langle \tilde{z} - z, x - \tilde{z} \rangle + \langle \tilde{z} - z, \tilde{z} - z \rangle \\ &\geq \|\tilde{z} - z\|^2 > 0. \end{aligned} \tag{A10}$$

■

Lemma 6: Let $G \subset \mathbb{R}^d$ be a nonempty closed convex set and let $z_1, z_2 \in \mathbb{R}^d$. Suppose $\tilde{z}_1$ and $\tilde{z}_2$ are the projections of z_1 and z_2 onto G, respectively. Then

1) $\langle\, \tilde{z}_1 - z_1, z_2 - z_1 \,\rangle > 0$ if $\|z_1 - \tilde{z}_1\| > \|z_2 - \tilde{z}_2\|$;
2) $\langle\, \tilde{z}_1 - z_1, z_2 - z_1 \,\rangle \geq 0$ if $\|z_1 - \tilde{z}_1\| \geq \|z_2 - \tilde{z}_2\|$.

Proof: We prove the lemma by contradiction. We only prove 1) since the proof of 2) is similar. Suppose the conclusion is not true, that is, $\langle\, \tilde{z}_1 - z_1, z_2 - z_1 \,\rangle \leq 0$. Since $\tilde{z}_2$ is the projection of z_2 onto G, then $\tilde{z}_2 \in G$. Replacing z and x with z_1 and $\tilde{z}_2$ in equation (A10) respectively, we have

$$\begin{aligned}\|\tilde{z}_1 - z_1\|^2 &\leq \langle\, \tilde{z}_1 - z_1, \tilde{z}_2 - z_1 \,\rangle \\ &= \langle\, \tilde{z}_1 - z_1, z_2 - z_1 \,\rangle + \langle\, \tilde{z}_1 - z_1, \tilde{z}_2 - z_2 \,\rangle \\ &\leq \langle\, \tilde{z}_1 - z_1, \tilde{z}_2 - z_2 \,\rangle \\ &\leq \|\tilde{z}_1 - z_1\|\|\tilde{z}_2 - z_2\| \end{aligned}$$

which means that $\|z_1 - \tilde{z}_1\| \leq \|z_2 - \tilde{z}_2\|$. This is a contradiction. The proof is complete. ■

Proof of Theorem 1. We only prove 1). The proof of 2) is similar to that of 1) by using Lemmas 5 and 6. For convenience, we assume that $S_1 = \{x_i\}_{i=1}^{d+1}$, where the index stands for the order of selection in Algorithm 2. We prove the theorem by contradiction. Suppose the conclusion is not true, that is, there exists a sample, namely $x_{j_0} \in S_1$, such that $x_{j_0} \in \text{conv}(P \setminus x_{j_0})$. This would imply that

$$x_{j_0} = \sum_{i=1, i \neq j_0}^{n} \alpha_i x_i, \quad \sum_{i=1, i \neq j_0}^{n} \alpha_i = 1, \quad \alpha_i \geq 0. \tag{A11}$$

Let $\mathcal{L}_1$ be the subspace spanned by $\{x_i - x_1\}_{i=1}^{j_0-1}$ and let $\tilde{x}_{j_0} - x_1$ be the projection of $x_{j_0} - x_1$ onto $\mathcal{L}_1$. By Lemma 2 we have

$$\langle\ \tilde{x}_{j_0} - x_{j_0}, x_i - x_{j_0}\ \rangle > 0, \quad i = 1, \ldots, j_0 - 1. \tag{A12}$$

From (A11), it follows that:

$$\sum_{i=1, i\neq j_0}^{n} \alpha_i (x_i - x_{j_0}) = 0.$$

Taking the inner product with $\tilde{x}_{j_0} - x_{j_0}$ on both sides of the above equation gives

$$\sum_{i=1, i\neq j_0}^{n} \alpha_i \langle\ x_i - x_{j_0}, \tilde{x}_{j_0} - x_{j_0}\ \rangle = 0. \tag{A13}$$

Here we need to consider two cases. For the case when the sample which is the furthest to the current subspace $\mathcal{L}_1$ in P is unique, that is, the distance from $x_i - x_1$ to $\mathcal{L}_1$ is strictly less than $\|x_{j_0} - \tilde{x}_{j_0}\|$ for $i = j_0+1, \ldots, n$, we have by Lemma 3 that

$$\langle\ \tilde{x}_{j_0} - x_{j_0}, x_i - x_{j_0}\ \rangle > 0, \quad i = j_0 + 1, \ldots, n. \tag{A14}$$

From (A12) and (A14), we see that (A13) gives a contradiction.

For the other case where there exists another sample, namely, x_t with $j_0 + 1 \leq t \leq n$, which is also the furthest to $\mathcal{L}_1$ in P, we obtain by Lemma 3 that

$$\langle \tilde{x}_{j_0} - x_{j_0}, x_t - x_{j_0} \rangle \geq 0, \tag{A15}$$

$$\langle \tilde{x}_{j_0} - x_{j_0}, x_i - x_{j_0} \rangle > 0,\ j_0 < i \leq n, i \neq t. \tag{A16}$$

From (A12), (A13), (A15) and (A16), we have

$$\alpha_i = 0, \quad i = 1, \ldots, n, i \neq j_0 \text{ and } i \neq t.$$

Then $\alpha_t = 1$ since $\sum_{i=1, i\neq j_0}^{n} \alpha_i = 1$, which means that $x_{j_0} = x_t$. This is also a contradiction. ∎

Proof of Theorem 2. We first prove 1). Since $(x - x_0) = \sum_{i=1}^{d} \alpha_i (x_i - x_0)$, we have $x + (\sum_{i=1}^{d} \alpha_i - 1)x_0 = \sum_{i=1}^{d} \alpha_i x_i$, so

$$x - x_1 = \sum_{i=2}^{d} \alpha_i (x_i - x_1) + \left(\sum_{i=1}^{d} \alpha_i - 1 \right)(x_1 - x_0).$$

This implies that 1) holds since $x_1 \neq x_0$.

We now prove 2). Let $\sum_{i=1}^{d} \alpha_i = \alpha$, $\alpha_i' = \frac{\alpha_i}{\alpha}$ and $\bar{x} = \frac{1}{\alpha} \sum_{i=1}^{d} \alpha_i x_i$. Then $\bar{x} - x_0 = \sum_{i=1}^{d} \alpha_i' (x_i - x_0)$. From 1) we see that $\bar{x}$ is on the hyperplane Π. Therefore, we have

$$\begin{aligned} \langle\ x_0 - \tilde{x}_0, x - \tilde{x}_0\ \rangle &= \langle\ x_0 - \tilde{x}_0, x - x_0\ \rangle + \|\tilde{x}_0 - x_0\|^2 \\ &= \alpha \langle\ x_0 - \tilde{x}_0, \bar{x} - x_0\ \rangle + \|\tilde{x}_0 - x_0\|^2 \\ &= \alpha \langle\ x_0 - \tilde{x}_0, \bar{x} - \tilde{x}_0 + \tilde{x}_0 - x_0\ \rangle \\ &\quad + \|\tilde{x}_0 - x_0\|^2 = (1 - \alpha)\|\tilde{x}_0 - x_0\|^2. \end{aligned}$$

From this 2) and 3) follow. The proof is complete. ∎

Proof of Theorem 3. 1) It is easy to see that

$$\begin{aligned} (U^z)^\top U^z &= T^\top U_S^\top U_S T = T^\top K^u T = T^\top V \Lambda V^\top T \\ &= \Lambda^{-\frac{1}{2}} V^\top V \Lambda V^\top V \Lambda^{-\frac{1}{2}} = I. \end{aligned}$$

2) For any $i, j = 1, \ldots, n$ we have

$$\begin{aligned} \|\varphi(x_i) - \varphi(x_j)\|^2 &= \|u_i - u_j\|^2 = \|U_S \tilde{z}_i - U_S \tilde{z}_j\|^2 \\ &= \|U_S T T^{-1} \tilde{z}_i - U_S T T^{-1} \tilde{z}_j\|^2 \\ &= \|U_S T (z_i - z_j)\|^2 = \|z_i - z_j\|^2. \end{aligned}$$

3) Let $I = \{j_i\}_{i=1}^{q}$,$I' = \{j_i'\}_{i=1}^{q'}$. We first prove that $I' \subset I$. From Algorithm 4, it is seen that for $t = 1, \ldots, n$

$$\varphi(x_t) = U^z z_t. \tag{A17}$$

On the other hand, since $\{\varphi(x_{j_i})\}_{i=1}^{q}$ is the set of vertices of $\text{conv}(\{\varphi(x_i)\}_{i=1}^{n})$, then there exist $\alpha_{t1}, \ldots, \alpha_{tq}$ such that for $t = 1, \ldots, n$

$$\begin{aligned} \varphi(x_t) &= [\varphi(x_{j_1}), \ldots, \varphi(x_{j_q})][\alpha_{t1}, \ldots, \alpha_{tq}]^T \\ &= U^z [z_{j_1}, \ldots, z_{j_q}][\alpha_{t1}, \ldots, \alpha_{tq}]^T \\ &\quad \sum_{i=1}^{q} \alpha_{ti} = 1,\ \alpha_{ti} \geq 0. \end{aligned} \tag{A18}$$

Since the columns of U^z are linearly independent from each other, then by combining (A17) and (A18) we have

$$z_t = [z_{j_1}, \ldots, z_{j_q}][\alpha_{t1}, \ldots, \alpha_{tq}]^T.$$

This implies that $I' \subset I$. Similarly, we can prove that $I \subset I'$. Thus, $I = I'$, and the proof is complete. ∎

ACKNOWLEDGMENT

The authors would like to thank the reviewers and the Associate Editor for their thorough consideration of this paper and for their valuable comments and suggestions, which helped to improve this paper.

REFERENCES

[1] V. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.

[2] V. Vapnik, "An overview of statistical learning theory," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 988–999, Sep. 1999.

[3] D. Anguita, A. Ghio, L. Oneto, and S. Ridella, "In-sample and out-of-sample model selection and error estimation for support vector machines," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 9, pp. 1390–1406, Sep. 2012.

[4] A. Shilton, M. Palaniswami, D. Ralph, and A. Tsoi, "Incremental training of support vector machines," *IEEE Trans. Neural Netw.*, vol. 16, no. 1, pp. 114–131, Jan. 2005.

[5] S. Pang, D. Kim, and S. Y. Bang, "Face membership authentication using SVM classification tree generated by membership-based LLE data partition," *IEEE Trans. Neural Netw.*, vol. 16, no. 2, pp. 436–446, Mar. 2005.

[6] Y. Liu and Y. Chen, "Face recognition using total margin-based adaptive fuzzy support vector machines," *IEEE Trans. Neural Netw.*, vol. 18, no. 1, pp. 178–192, Jan. 2007.

[7] P. Shih and C. Liu, "Face detection using discriminating feature analysis and support vector machine," *Pattern Recognit.*, vol. 39, no. 2, pp. 260–276, Feb. 2006.

[8] S. Kim, Y. Park, K. Toh, and S. Lee, "SVM-based feature extraction for face recognition," *Pattern Recognit.*, vol. 43, no. 8, pp. 2871–2881, 2010.

[9] X. Cao and H. Qiao, "A low-cost pedestrian detection system with a single optical camera," *IEEE Trans. Intell. Transp. Syst.*, vol. 9, no. 1, pp. 58–67, Mar. 2008.

[10] K. Labusch, E. Barth, and T. Martinetz, "Simple method for high-performance digit recognition based on sparse coding," *IEEE Trans. Neural Netw.*, vol. 19, no. 11, pp. 1985–1989, Nov. 2008.

[11] K. Kim, K. Jung, S. Park, and H. Kim, "Support vector machine-based text detection in digital video," *Pattern Recognit.*, vol. 34, no. 2, pp. 527–529, 2001.

[12] V. Mitra, C. Wang, and S. Banerjee, "A neuro-SVM model for text classification using latent semantic indexing," in *Proc. Int. Joint Conf. Neural Netw.*, Montreal, PQ, Canada, Jul. 2005, pp. 564–569.

[13] S. K. Shevade, S. S. Keerthi, C. Bhattacharyya, and K. R. K. Murthy, "Improvements to the SMO algorithm for SVM regression," *IEEE Trans. Neural Netw.*, vol. 11, no. 5, pp. 1188–1193, Sep. 2000.

[14] C. C. Chang and C. J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, article 27, Apr. 2011.

[15] P. Chen, R. Fan, and C. Lin, "A study on SMO-type decomposition methods for support vector machines," *IEEE Trans. Neural Netw.*, vol. 17, no. 4, pp. 893–908, Jul. 2006.

[16] F. Cai and V. Cherkassky, "Generalized SMO algorithm for SVM-based multitask learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 6, pp. 997–1003, Jun. 2012.

[17] J. Lopez and J. R. Dorronsoro, "Simple proof of convergence of the SMO algorithm for different SVM variants," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 7, pp. 1142–1147, Jul. 2012.

[18] C. C. Chang, C. W. Hsu, and C. J. Lin, "The analysis of decomposition methods for support vector machines," *IEEE Trans. Neural Netw.*, vol. 11, no. 4, pp. 1003–1008, Jul. 2000.

[19] C. Lin, "On the convergence of the decomposition method for support vector machines," *IEEE Trans. Neural Netw.*, vol. 12, no. 6, pp. 1288–1298, Nov. 2001.

[20] C. J. Lin, "A formal analysis of stopping criteria of decomposition methods for support vector machines," *IEEE Trans. Neural Netw.*, vol. 13, no. 5, pp. 1045–1052, Sep. 2002.

[21] J. Dong, A. Krzyzak, and C. Y. Suen, "Fast SVM training algorithm with decomposition on very large data sets," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 4, pp. 603–618, Apr. 2005.

[22] H. Qiao, Y. Wang, and B. Zhang, "A simple decomposition algorithm for support vector machines with polynomial-time convergence," *Pattern Recognit.*, vol. 40, no. 9, pp. 2543–2549, 2007.

[23] S. Lucidi, L. Palagi, A. Risi, and M. Sciandrone, "A convergent hybrid decomposition algorithm model for SVM training," *IEEE Trans. Neural Netw.*, vol. 20, no. 6, pp. 1055–1060, Jun. 2009.

[24] I. W. Tsang, J. T. Kwok, and P. M. Cheung, "Core vector machines: Fast SVM training on very large data sets," *J. Mach. Learn. Res.*, vol. 6, pp. 363–392, Apr. 2005.

[25] I. W. Tsang, J. T. Kwok, and J. Zurada, "Generalized core vector machines," *IEEE Trans. Neural Netw.*, vol. 17, no. 5, pp. 1126–1140, Sep. 2006.

[26] K. P. Bennett and E. J. Bredensteiner, "Duality and geometry in SVM classifiers," in *Proc. 17th Int. Conf. Mach. Learn.*, San Francisco, CA, 2000, pp. 57–64.

[27] D. J. Crisp and C. J. C. Burges, "A geometric interpretation of nu-SVM classifiers," in *Proc. 13th Annu. Conf. Neural Inf. Process. Syst.*, 2000, pp. 244–250.

[28] X. J. Peng and Y. F. Wang, "Geometric algorithms to large margin classifier based on affine hulls," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 2, pp. 236–246, Feb. 2012.

[29] L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, *Large Scale Kernel Machines.* Cambridge, MA: MIT Press, 2007.

[30] A. Menon. (2009). *Large-Scale Support Vector Machines: Algorithms and Theory* [Online]. Available: http://cseweb.ucsd.edu/∼akmenon/ResearchExam.pdf

[31] L. Bottou, *Online Learning and Stochastic Approximations*, Cambridge, U.K.: Cambridge Univ. Press, 1998, pp. 9–42.

[32] S. Agarwal, V. Saradhib, and H. Karnick, "Kernel-based online machine learning and support vector reduction," *Neurocomputing*, vol. 71, nos. 7–9, pp. 1230–1237, 2008.

[33] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2165–2176, Aug. 2004.

[34] A. Bordes, L. Bottou, and P. Gallinari, "SGD-QN: Careful quasi-Newton stochastic gradient descent," *J. Mach. Learn. Res.*, vol. 10, wpp. 1737–1754, Jul. 2009.

[35] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, "Pegasos: Primal estimated sub-gradient solver for SVM," *Math. Programming*, vol. 127, no. 1, pp. 3–30, Mar. 2011.

[36] N. Syed, H. Liu, and K. Sung, "Handling concept drifts in incremental learning with support vector machines," in *Proc. 5th ACM SIGKDD Int. Conf. Knowl. Siscovery Data Mining*, 1999, pp. 317–321.

[37] P. Mitra, C. Murthy, and S. Pal, "Data condensation in large databases by incremental learning with support vector machines," in *Proc. Int. Conf. Pattern Recognit.*, vol. 2. Sep. 2000, pp. 708–711.

[38] C. Domeniconi and D. Gunopulos, "Incremental support vector machine construction," in *Proc. 1st Int. Conf. Data Mining*, Nov. 2001, pp. 589–592.

[39] B. Peng, Z. Sun, and X. Xu, "SVM-based incremental active learning for user adaptation for online graphics recognition system," in *Proc. 1st Int. Conf. Mach. Learn. Cybern.*, Nov. 2002, pp. 1379–1386.

[40] J. An, Z. Wang, and Z. Ma, "An incremental learning algorithm for support vector machine," in *Proc. Int. Conf. Mach. Learn. Cybern.*, Nov. 2003, pp. 1153–1156.

[41] R. Singh, M. Vatsa, A. Ross, and A. Noore, "Biometric classifier update using online learning: A case study in near infrared face verification," *Image Vis. Comput.*, vol. 28, no. 7, pp. 1098–1105, Jul. 2010.

[42] A. Bordes, S. Ertekin, J. Weston, and L. Bottou, "Fast kernel classifiers with online and active learning," *J. Mach. Learn. Res.*, vol. 6, pp. 1579–1619, Sep. 2005.

[43] M. Wang, X. Zhou, F. Li, J. Huckins, R. King, and S. T. C. Wong, "Novel cell segmentation and online SVM for cell cycle phase identification in automated microscopy," *Bioinformatics*, vol. 24, no. 1, pp. 94–101, 2008.

[44] S. Cheng and F. Shih, "An improved incremental training algorithm for support vector machines using active query," *Pattern Recognit.*, vol. 40, no. 3, pp. 964–971, 2007.

[45] K. Lau and Q. Wu, "Online training of support vector classifier," *Pattern Recognit.*, vol. 36, no. 8, pp. 1913–1920, 2003.

[46] G. Cauwenberghs and T. Poggio, "Incremental and decremental support vector machine learning," in *Advances in Neural Information Processing Systems*, Cambridge, MA: MIT Press, 2001, pp. 409–415.

[47] D. Wang, B. Zhang, P. Zhang, and H. Qiao, "An online core vector machine with adaptive MEB adjustment," *Pattern Recognit.*, vol. 43, no. 10, pp. 3468–3482, 2010.

[48] A. Bordes and L. Bottou, "The Huller: A simple and efficient online SVM," in *Proc. 16th Eur. Conf. Mach. Learn.*, Oct. 2005, pp. 505–512.

[49] J. X. Dong, A. Krzyzak, and C. Y. Suen, "A practical SMO algorithm," in *Proc. Int. Conf. Pattern Recognit.*, vol. 3. Aug. 2002, pp. 1–4.

[50] X. F. He and P. Niyogi, "Locality preserving projections," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2003.

[51] X. He, S. Yan, Y. Hu, P. Niyogi, and H. Zhang, "Face recognition using Laplacianfaces," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 3, pp. 328–340, Mar. 2005.

[52] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, Dec. 2000.

[53] L. K. Saul and S. T. Roweis, "Think globally, fit locally: Unsupervised learning of low dimensional manifold," *J. Mach. Learn. Res.*, vol. 4, pp. 119–155, Jan. 2003.

[54] H. Qiao, P. Zhang, D. Wang, and B. Zhang, "An explicit nonlinear mapping for manifold learning," *IEEE Trans. Cybern.*, vol. 43, no. 1, pp. 51–63, Feb. 2013.

[55] F. Orabona, C. Castellini, B. Caputo, L. Jie, and G. Sandini, "Online independent support vector machines," *Pattern Recognit.*, vol. 43, no. 4, pp. 1402–1412, Apr. 2010.

[56] A. Frank and A. Asuncion. (2010, Mar. 1). *UCI Machine Learning Repository* [Online]. Available: http://archive.ics.uci.edu/ml

[57] *CBCL Face Database 1*. (2000) [Online]. Available: http://www.ai.mit.edu/projects/cbcl

[58] A. Uzilov, J. Keegan, and D. Mathews, "Detection of non-coding RNAs on the basis of predicted secondary structure formation free energy change," *BMC Bioinformatics*, vol. 7, no. 1, p. 173, Mar. 2006.

[59] D. M. J. Tax and R. P. W. Duin, "Support vector data description," *Mach. Learn.*, vol. 54, no. 1, pp. 45–66, 2004.

[60] D. M. J. Tax and R. P. W. Duin, "Outlier deletion using classifier instability," in *Proc. Joint Int. Workshops Adv. Pattern Recognit.*, 1998, pp. 593–601.

[61] D. Geebelen, J. A. K. Suykens, and J. Vandewalle, "Reducing the number of support vectors of SVM classifiers using the smoothed separable case approximation," *IEEE Trans. Neural Netw. Learn. Syst*, vol. 23, no. 4, pp. 682–688, Apr. 2012.

Di Wang received the B.Sc. degree from Shandong University, Jinan, China, and the Ph.D. degree in applied mathematics from the Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China, in 2007 and 2012, respectively.

He is currently an Assistant Professor with the College of Mathematics and Information Sciences, Wenzhou University, Wenzhou, China. His current research interests include theory and application of support vector machines.

Hong Qiao (SM'06) received the B.Eng. degree in hydraulics and control and the M.Eng. degree in robotics from Xian Jiaotong University, Xian, China, the M.Phil. degree in robotics control from the Industrial Control Center, University of Strathclyde, Strathclyde, U.K., and the Ph.D. degree in robotics and artificial intelligence from De Montfort University, Leicester, U.K., in 1995.

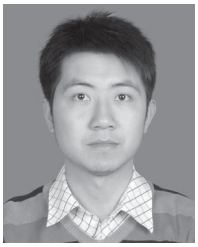
She was a University Research Fellow with De Montfort University from 1995 to 1997. She was a Research Assistant Professor from 1997 to 2000 and an Assistant Professor from 2000 to 2002 with the Department of Manufacturing Engineering and Engineering Management, City University of Hong Kong, Kowloon, Hong Kong. In 2002, she joined as a Lecturer with the School of Informatics, University of Manchester, Manchester, U.K. She is currently a Professor with the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China. She first proposed the concept of "the attractive region in strategy investigation" that she successfully applied for robot assembly, robot grasping, and part recognition, which is reported in *Advanced Manufacturing Alert* (Wiley, 1999). Her current research interests include information-based strategy investigation, robotics and intelligent agents, animation, machine learning, and pattern recognition.

Dr. Qiao is a member on the Program Committee of the IEEE International Conference on Robotics and Automation from 2001 to 2004. She is currently an Associate Editor of the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, PART B, and the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING.

Bo Zhang (M'10) received the B.Sc. degree in mathematics from Shandong University, Jinan, China, the M.Sc. degree in mathematics from Xi'an Jiaotong University, Xian, China, and the Ph.D. degree in applied mathematics from the University of Strathclyde, Strathclyde, U.K., in 1983, 1985, and 1992, respectively.

He is currently a Professor with the Institute of Applied Mathematics, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China. From 1985 to 1988, he was a Lecturer with the Department of Mathematics, Xi'an Jiaotong University. He was a Post-Doctoral Research Fellow with the Department of Mathematics, Keele University, Keele, U.K., from 1992 to 1994, and with the Department of Mathematical Sciences, Brunel University, Uxbridge, U.K., from 1995 to 1997. In 1997, he joined the School of Mathematical and Informational Sciences, Coventry University, Coventry, U.K., as a Senior Lecturer, where he was a Reader of applied mathematics in 2000 and a Professor of applied mathematics in 2003. His current research interests include direct and inverse scattering problems, radar and medical imaging, machine learning, and pattern recognition.

Dr. Zhang is currently an Associate Editor of the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, PART B, and *Applicable Analysis*.



Min Wang received the B.Sc. degree from the University of Science and Technology of China, Hefei, China, and the Ph.D. degree in pattern recognition and intelligent systems from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2007 and 2012, respectively.

He is currently an Assistant Professor with the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences. His current research interests include manifold learning methods for visual tracking, dynamic image processing, and vision of intelligent vehicles.