A Fast Algorithm of Convex Hull Vertices Selection for Online Classification

Shuguang Ding, Xiangli Nie, *Member, IEEE*, Hong Qiao, *Senior Member, IEEE*, and Bo Zhang, *Member, IEEE*

***Abstract*— Reducing samples through convex hull vertices selection (CHVS) within each class is an important and effective method for online classification problems, since the classifier can be trained rapidly with the selected samples. However, the process of CHVS is NP-hard. In this paper, we propose a fast algorithm to select the convex hull vertices, based on the convex hull decomposition and the property of projection. In the proposed algorithm, the quadratic minimization problem of computing the distance between a point and a convex hull is converted into a linear equation problem with a low computational complexity. When the data dimension is high, an approximate, instead of exact, convex hull is allowed to be selected by setting an appropriate termination condition in order to delete more nonimportant samples. In addition, the impact of outliers is also considered, and the proposed algorithm is improved by deleting the outliers in the initial procedure. Furthermore, a dimension convention technique via the kernel trick is used to deal with nonlinearly separable problems. An upper bound is theoretically proved for the difference between the support vector machines based on the approximate convex hull vertices selected and all the training samples. Experimental results on both synthetic and real data sets show the effectiveness and validity of the proposed algorithm.**

***Index Terms*— Convex hull decomposition, kernel, online classification, projection.**

I. Introduction

IN THE field of machine learning, online learning is referred to gradually updating a classifier based on a stream of samples arriving sequentially [1], [22], [24], [26], [30]. When a newly coming sample is misclassified, the current classifier should be updated immediately so that the prediction ability of the classifier can be improved step by step. Owing to the advantage of its scalability and high efficiency, online learning has wide applications in real-world large-scale data mining, and especially in real-time pattern recognition, such as aircraft visual navigation and pedestrian detection [29]–[31].

The main challenge of online learning is how to improve the computational efficiency in the online step. In recent years, various online learning algorithms have been proposed to address this issue. In general, almost all of the online learning algorithms are developed from the traditional algorithms, such as the Bayesian classifier [17], [19], [32], the kernel method [1], [4], [15], the perceptron algorithm [10], [20], and the support vector machine (SVM) classifier [9], [26], [29], [30]. These methods are mainly based on two approaches: 1) using the stochastic gradient descent algorithm, which can reduce the computational complexity, to optimize the object function in the online step and 2) reducing the training samples in the off-line step or reducing the new coming samples periodically to keep the number of the current training samples in a small range. The main idea of reducing samples is to use a criterion to reduce the large amount of redundant samples of the training set. Therefore, the classifier can be trained only with the small amount of remaining samples, so the computational efficiency can be improved.

In this paper, we focus on the idea of reducing samples. Based on this idea, several successful online learning algorithms have been proposed [1], [3], [4], [9], [16], [23], [29], [30], which will be briefly reviewed in Section II. Among these algorithms, the online SVM algorithm based on convex hull vertices selection (called VS-OSVM), which was proposed in [30] and based on the geometric feature of SVM, selects a small number of skeleton samples constituting an approximate convex hull and trains the SVM classifier with the selected samples. The VS-OSVM algorithm is able to guarantee that the samples inside the approximate convex hull for each class can be safely deleted. Thus, it can improve the efficiency of the online updating process. In addition, VS-OSVM can deal with nonlinearly separable problems by using the kernel trick together with a dimension conversion technique.

However, the VS-OSVM algorithm has some shortcomings. First, the process of selecting the convex vertices is time-consuming when the number of the training samples is large and the dimension of the training samples is high. Though the process of convex hull vertices selection (CHVS) on the training set is off-line, this process should be executed when the number of new coming samples is greater than a given threshold. Therefore, improving the efficiency of the algorithm is meaningful to online learning. Second, if the data dimension is high, almost all the samples are the convex hull vertices so that few samples can be deleted. To overcome this difficulty,

Manuscript received January 1, 2016; revised September 14, 2016; accepted December 31, 2016. This work was supported in part by NNSFC under Grant 61210009, Grant 61379093, and Grant 61602483 and in part by the Strategic Priority Research Program of CAS under Grant XDB02080003. *(Corresponding author: Bo Zhang.)*

S. Ding is with the Institute of Applied Mathematics, Chinese Academy of Sciences, Beijing 100190, China (e-mail: dingshuguang12b@mails.ucas.ac.cn).

X. Nie is with the State Key Laboratory of Management Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China (e-mail: xiangli.nie@ia.ac.cn).

H. Qiao is with the State Key Laboratory of Management Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, also with the CAS Centre for Excellence in Brain Science and Intelligence Technology, Shanghai 200031, China, and also with the University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: hong.qiao@ia.ac.cn).

B. Zhang is with the LSEC and Academy of Mathematics and Systems Sciences, Chinese Academy of Sciences, Beijing, 100190, China, and also with the University of Chinese Academy of Sciences, Beijing 100049, 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.2017.2648038

the number of selected vertices is limited to a given bound in VS-OSVM, but it cannot measure the degree of approximation of the current convex hull thus obtained. Finally, the algorithm does not consider the case where there may be some outliers in the data, and thus always selects the outliers as convex hull vertices in such a case.

In this paper, we aim to address the above issues and propose a fast algorithm to select the convex hull vertices. The main contributions of this paper, which are also the differences between the proposed algorithm and VS-OSVM, are given as follows: 1) the convex hull decomposition and the property of projection are applied to convert the quadratic optimization problem of computing the distance between a point and a convex hull into a linear equation problem, which reduces the computational complexity and improves the efficiency of the process of the convex hull vertices selection; 2) an approximate, instead of exact, convex hull is obtained by controlling the maximum distance between the training samples and the approximate convex hull in order to delete more samples which are less important for high-dimensional data; here, the importance of a sample is measured by its distance from the approximate convex hull, that is to say, if the distance between a sample and the approximate convex hull less than a given threshold, the sample can be deleted safely; 3) the proposed algorithm is improved by deleting the outliers; and 4) an upper bound is theoretically proved for the difference between the SVMs using only the approximate convex hull vertices selected and all the training samples. It should be remarked that the contributions 2) and 3) above will play an important role in avoiding overfitting, so they will have a very positive impact on the performance of the proposed algorithm, as demonstrated in Section IV.

The above contributions are summarized in five algorithms in Section III. Algorithm 1 gives the method of selecting convex hull vertices by using the convex hull decomposition. Algorithm 2 presents the method of computing the distance between a point and a hyperplane. Algorithm 3 selects the convex hull vertices by using the property of projection. Algorithms 4 and 5 present the method of deleting outliers and the method of computing distance between a point and a hyperplane via the kernel trick, respectively.

The rest of this paper is organized as follows. In Section II, the existing online learning algorithms based on reducing samples are briefly reviewed. Section III presents the proposed fast algorithm of convex hull vertices selection for online learning. Experimental results on synthetic and real data are shown in Section IV. Some concluding remarks are given in Section V.

II. Related Works

This section is divided into two parts: 1) Section II-A gives a brief review on the existing online learning algorithms based on reducing samples and 2) Section II-B discusses briefly the VS-OSVM algorithm introduced previously in [30].

A. Online Learning Algorithms Based on Reducing Samples

Online learning algorithms based on reducing samples can be divided into three types: 1) choosing the most important support vectors; 2) using the reduced set method; and 3) deleting most of the redundant samples.

For the first type, since the SVM classifier is only related to the support vectors [6], [11], [14], [28], many algorithms have been proposed to choose the most important support vectors. Bordes *et al.* [4] presented an online SVM algorithm called LASVM, which uses two operations, called PROCESS and REPOCESS, to update the classifier with the newly coming samples. The PROCESS is used to add a support vector to the current set, while the REPROCESS is used to remove a nonsupport vector from the current set. The main criterion in the two operations is based on the τ-violating pair, which is induced by the Karush-Kuhn-Tucker (KKT) condition of the primal problem of SVMs. With the aid of LASVM, Ertekin *et al.* [9] introduced a nonconvex online SVM, which uses the ramp loss to substitute the hinge loss in the SVM classifier. The merit of the ramp loss is that it is not sensitive to the outliers and can reduce the impact of the outliers. Geebelen *et al.* [12] proposed a separable case approximation (SCA) approach to reduce the support vectors. In SCA, a linear SVM classifier is first trained, and then, the training set is modified by using two methods: flipping the labels of the misclassified samples or removing the misclassified samples. In addition, the smoothed SCA is presented to restrict the norm of the weight vector of the classifier. Agarwal *et al.* [1] made use of the span of support vectors (SVs) to reduce the support vectors. The advantage of this method is that it can handle problems in the case when the memory is restricted. Bordes and Bottou [3] proposed an online kernel classifier algorithm called Huller. Based on the geometrical property of SVM, the Huller adds and removes support vectors from the current classifier by updating the nearest points of the convex hulls.

The second type is the reduced set method. Burges [5] proposed the concept of reduced set vectors that are not support vectors and not necessarily selected from the training set. The size of the reduced set is much smaller than that of the set of support vectors, so it can be used to decrease the complexity of SVM. Instead of solving the primal problem of SVM, Burges [5] suggested to compute an approximation to the origin SVM classifier in terms of the reduced set of vectors. Schölkopf *et al.* [23] gave a detailed introduction to the reduced subset selection methods and the subset extraction methods. Inspired by these methods, several improved algorithms based on the reduced set technique are proposed. Nguyen and Ho [18] gave an efficient method by reducing the necessary support vectors. In the reduction process of their algorithm, two support vectors, which are the nearest in the same class, are replaced by a convex combination of the two vectors. The combination coefficients are obtained by the Lagrangian multipliers of the two support vectors. In [33], to control the sparseness of the classifier, an explicit constraint is added to the primal formulation of SVM.

The third type is to delete most of the redundant samples. By using an adaptive minimum-enclosing-ball adjustment, Wang *et al.* [29] proposed an online core vector machine (CVM) algorithm. In the off-line process, the online CVM algorithm deletes all the samples inside a definite ball, which does not affect the final classifier. Motivated by the

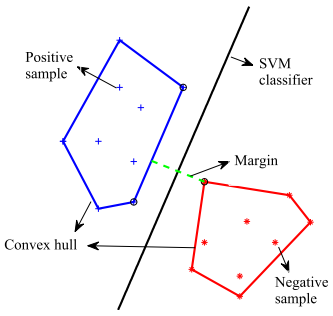


Fig. 1. Geometric feature of SVM.

geometrical property of SVMs, Wang *et al.* [30] proposed an online SVM algorithm called VS-OSVM, based on CHVS, which can update the classifier without reducing its prediction ability. Our algorithm is based on this algorithm, so we will introduce VS-OSVM in detail in Section II-B.

B. VS-OSVM Algorithm

For the linearly separable problem, the solution of SVM is equivalent to finding the minimum distance between two convex hulls, which are formed by the samples of the two classes (see Fig. 1). Therefore, the SVM classifier only relates to the samples, which are the convex hull vertices. The VS-OSVM algorithm uses this nature of SVM as its starting point.

The VS-OSVM classifier is based on selecting the vertices of the convex hull and consists of two steps: 1) the off-line sample selection process, in which only the samples constituting an approximate convex hull of the training samples in each class are selected and 2) the online updating process, in which the classifier is updated with the newly coming samples and the selected samples. The samples selection process (SSP) is executed periodically in order to constrain the number of the currently selected samples.

The VS-OSVM algorithm can be divided into four stages. First, the vertices of a d-simplex is selected (it is assumed that the data samples are in $\mathbb{R}^d$). In the iteration step of this algorithm, the sample that is the furthest to the current linear subspace is selected. Second, the samples in the feature space are converted into those in a finite dimensional space by using the dimension conversion technique. Third, all the training samples are divided into several parts so that the number of the samples is less than a given bound in each part. Finally, the sample, which is the furthest to the current convex hull, is selected at each iteration.

To some extent, VS-OSVM can successfully delete the redundant samples in the feature space. However, the computation efficiency is low when the number of the data samples is large and the dimension of the data samples is high. In addition, for high-dimensional data most of the data samples are very close to the edge of the convex hull [13], so few samples can be deleted. This difficulty is overcome in [30] by controlling the number of the selected samples. In the SSP, a sequence of vertices is selected by restricting the number of vertices. At each iteration, the testing process is executed. SSP will be terminated when the number of vertices increases but the test error changes a little. This method is efficient in some cases, but it cannot measure the degree of approximation of the current convex hull. In our algorithm, a parameter ϵ is used to measure the approximation degree of the current approximate convex hull.

III. Proposed Algorithm

In this section, the proposed algorithm is presented in detail. A convex hull decomposition method is introduced in Section III-A to restrict the number of convex hull vertices in the high-dimensional case. The projection process is proposed in Section III-B to reduce the computational complexity of computing the distance between a point and a convex hull and to improve the efficiency of the process of the convex hull vertices selection. In Section III-C, the proposed algorithm is further improved by considering the outliers. In Section III-D, the proposed algorithm is extended to deal with nonlinearly separable problems, based on the kernel trick and a dimension conversion technique. The termination condition of the proposed algorithm is discussed in Section III-E. A theoretical analysis of the proposed algorithm is provided in Section III-F.

A. Convex Hull Decomposition

Given a set $P = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$ and a point $x \in \mathbb{R}^d$, the convex hull of the set P is denoted by $\text{conv}(P)$, where $\text{conv}(P) = \{\sum_{i=1}^n \alpha_i x_i \,|\, x_i \in P, \sum_{i=1}^n \alpha_i = 1, \alpha_i \geq 0, i = 1, \ldots, n\}$. The Euclidean distance between x and $\text{conv}(P)$ can be computed by solving the quadratic optimization problem

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

where $e = [1, 1, \ldots, 1]^T$, $Q = X^T X$ and $c = X^T x$ with $X = [x_1, \ldots, x_n]$. Denote by α^* the solution of (1). Then, the Euclidean distance between x and $\text{conv}(P)$ can be given by

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

In the CHVS algorithm in [30], the number of the convex hull vertices will continually increase, which makes the quadratic optimization problem difficult to solve. In this paper, we propose a convex hull decomposition method to reduce the scale of the quadratic optimization problem. The process is presented as follows.

Step 1: The $d + 1$ vertices of a d-simplex are selected.

Step 2: The samples are divided into several parts.

Step 3: The sample, which is the furthest to the current convex hull, is selected at each iteration.

Step 4: When a new sample is selected, we divide the old part including the new samples into several subconvex hulls. Thus, the number of each subconvex hull vertices is equal to the dimension of the data.

The difference between our algorithm and the CHVS algorithm is reflected in Step 4. In our algorithm, the scale of the quadratic optimization problem never changes, which can reduce the computational time of the quadratic optimization

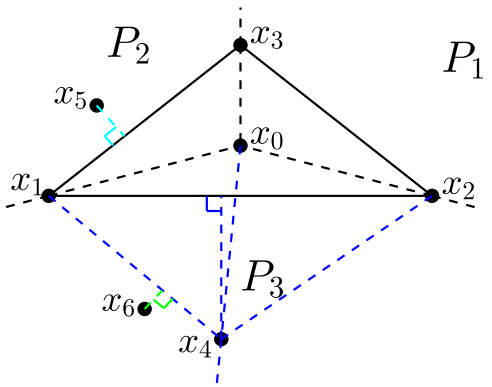


Fig. 2. Difference of the CHVS processes between our algorithm and the VS-OSVM algorithm.

problem. Step 4 of our algorithm is similar to the data partition process of CHVS, but the two procedures have a little difference. In the data partition process of CHVS, the sample which is the furthest to the corresponding facet (but not always the furthest to the current convex hull) is selected. However, in our algorithm, the sample which is the furthest to the current convex hull is selected. This process in our algorithm can guarantee that the selected samples keep the most information of the training data, especially when we select an approximate convex hull. Fig. 2 shows the difference of the CHVS processes between our algorithm and the CHVS algorithm. Let the initial set $S = \{x_1, x_2\}$. In Step 1 of our algorithm, x_3 is selected and $\{x_1, x_2, x_3\}$ is the set of vertices of a 2-simplex. In Step 2, the samples are divided into three parts: P_1, P_2, and P_3. Then, in our algorithm, x_4, the furthest sample to the current convex hull, is selected, and the part P_3 is divided into two small parts. Next, x_5 and x_6 are selected consecutively. Thus, the order of the selected samples is $x_1 \to x_2 \to x_3 \to x_4 \to x_5 \to x_6$. However, in the data partition process of the CHVS algorithm, the sample, which is the furthest to the corresponding facet, is selected after the samples are divided into three parts. In this case, x_4 and x_5 are, respectively, the furthest points to their corresponding facets: line x_1x_2 and line x_1x_3, which results in the fact that the order of selecting the samples x_4 and x_5 is random. Therefore, in CHVS, the order of the selected samples may be $x_1 \to x_2 \to x_3 \to x_5 \to x_4 \to x_6$. It can be observed that if we just select four samples constituting an approximate convex hull, the proposed method would be more reliable than the CHVS algorithm proposed in [30]. This is because the maximum distance between the training samples and the approximate convex hull $\text{conv}(\{x_1, x_2, x_3, x_4\})$ is less than that between the training samples and the approximate convex hull $\text{conv}(\{x_1x_2x_3x_5\})$. Thus, the convex hull $\text{conv}(\{x_1, x_2, x_3, x_4\})$ is a better approximation to the convex hull $\text{conv}(\{x_1x_2x_3x_4x_5x_6\})$ compared with the convex hull $\text{conv}(\{x_1x_2x_3x_5\})$.

The detailed process of convex hull decomposition is presented in Algorithm 1, where M is the number of the divided parts and I is the index of the divided parts. The parameter ϵ in Algorithm 1 is used to control the termination condition, which will be discussed in detail in Section III-E. In addition,

Algorithm 1 CHVS1: CHVS Based on the Convex Hull Decomposition

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

1: Randomly select $x_0 \in P$. Let $x_{j_0} = \text{argmax}_{x \in P} \|x_0 - x\|_2$ and $x_{j_1} = \text{argmax}_{x \in P} \|x_{j_0} - x\|_2$. Initialize $S = \{x_{j_0}, x_{j_1}\}$, $I = \{1, \ldots, d\}$ and $M = d + 1$.
2: Selecting $d + 1$ vertices of a d-Simplex in R^d by using Algorithm 1 of [30]. $S = \{x_i\}_{i=1}^{d+1}$ is the set of selected vertices.
3: Divide the samples into several parts by using the Constrained Data Partition Algorithm 2 of [30] and obtain the sets P_{oc} and S_{oc}, where $P_{oc} = \{P_i\}_{i=1}^{d+1}$ with P_i the ith divided part of P obtained by Algorithm 2 of [30] and $S_{oc} = \{S_i\}_{i=1}^{d+1}$ with $S_i = \{x_k\}_{\substack{k=1 \\ k \neq i}}^{d+1}$.
4: **for** $i = 1; i \leq d + 1; i++$ **do**
5: Find the point $x_i^* \in P_i$, which is furthest from the convex hull $\text{conv}(S_i)$, by using (2). Denote by m_i the distance between x_i^* and $\text{conv}(S_i)$.
6: **end for**
7: Find the maximum distance: $m = \max_{i \in I} m_i$.
8: **while** $m > \varepsilon$ **do**
9: Let $j_0 = \arg\max_{i \in I} m_i$ and $x_{j_0}^{(d+1)} = x_{j_0}^*$.
10: $S = S \cup \{x_{j_0}\}$, $I = I \setminus \{j_0\}$.
11: Let $E_i = \{x_{j_0}^{(t)}\}_{\substack{t=1 \\ t \neq i}}^{d+1}$ for $i = 1, \ldots, d$.
12: Divide P_{j_0} into d parts and gain B_i, $i = 1, \ldots, d$, where B_i is the ith part.
13: Let $P_{M+i} = B_i$, $S_{M+i} = E_i$ for $i = 1, \ldots, d$.
14: **for** $i = 1, i \leq d, i++$ **do**
15: Find the point $x_i^* \in P_{M+i}$, which is furthest from the convex hull $\text{conv}(S_{M+i})$, by using (2). Denote by m_{M+i} the distance between x_{M+i}^* and $\text{conv}(S_{M+i})$.
16: **end for**
17: Let $I = I \cup \{M + 1, \ldots, M + d\}$, $M = M + d$.
18: Find the maximum distance: $m = \max_{i \in I} m_i$.
19: **end while**

Output: S.

Steps 2 and 3 are the same as in [30, Algorithms 1 and 2], respectively. The aim of [30, Algorithm 1] is to select the $d+1$ samples that can construct an as large d-simplex as possible. Taking Fig. 2 for example, we select three samples by the following strategies: 1) initialize $S = \{x_1, x_2\}$ by step 2 of Algorithm 1 and 2) select the furthest point (that is x_3) to the straight line passing through x_1 and x_2. Then, $\{x_1, x_2, x_3\}$ are vertices of $2-$Simplex in R^2. Reference [30, Algorithm 2] is used to divide the samples into several parts. For example, in Fig. 2, suppose $S = \{x_1, x_2, x_3\}$ and x_0 is the mean of samples $\{x_1, x_2, x_3\}$. Then, the lines x_0x_1, x_0, x_2, and x_0, x_3 divide the samples into three parts P_1, P_2, and P_3. The detail can be found in Algorithm 2 of [30]. The process of computing the distance between a point and a convex hull is used in Steps 5 and 15 of Algorithm 1. Next, we will use a new and effective method to compute this distance.

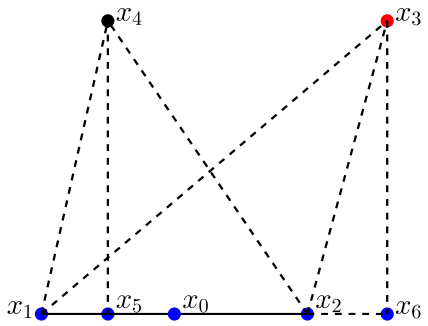


Fig. 3. Example of projection.

B. Projection Method

Though the convex hull decomposition method can improve the computational efficiency of the CHVS algorithm, the cost of solving a quadratic optimization problem is still huge when the dimension of data is high. To overcome this difficulty, we now propose a projection method to compute the distance between a sample and a subconvex hull.

Given a set $S = \{x_i\}_{i=1}^d \subset \mathbb{R}^d$, the convex hull of S is defined by $\text{conv}(S) = \{\sum_{i=1}^d \alpha_i x_i \mid x_i \in S, \sum_{i=1}^d \alpha_i = 1, \alpha_i \geq 0, i = 1, \ldots, d\}$, and the hyperplane composed of S is defined by $H = \{\sum_{i=1}^d \alpha_i x_i \mid x_i \in S, \sum_i^d \alpha_i = 1, \alpha_i \in \mathbb{R}, i = 1, \ldots, d\}$.

Given a point $x \in \mathbb{R}^d$, let d_c be the Euclidean distance between x and $\text{conv}(S)$ and let d_p be the Euclidean distance between x and H. From the definition of projection, $d_c = d_p$ if and only if the projection of x onto H is in $\text{conv}(S)$. For example, in Fig. 3, let $S = \{x_1, x_2\}$. If $x = x_4$, then $d_c = d_p$, since the projection of x_4 onto the line x_1x_2 is x_5, which is actually on the line x_1x_2. However, if $x = x_3$, we have $d_c = \|x_2 - x_3\|_2$ and $d_p = \|x_6 - x_3\|_2$, where $d_c \neq d_p$. Thus, d_c can be computed by two steps: 1) calculating the location of the projection and 2) computing d_p and the distance between x and x_i $(i = 1, \cdots, d)$.

Consider the case in Fig. 3 again. Suppose x_0 is the center of the line x_1x_2 and $x \in \mathbb{R}^2$. According to the property of projection, the projection of x onto the line x_1x_2 is on the line x_1x_2 if and only if $\angle x x_1 x_0 \leq 90°$ and $\angle x x_2 x_0 \leq 90°$. This is also true in the high-dimensional case. Thus, if the projection of x onto H is in $\text{conv}(S)$, d_c can be computed as follows:

$$d_c(x, \text{con}(S)) = d_p(x, H) = \|\beta^T(x - x_1)\|_1 / \|\beta\|_2 \quad (3)$$

where $\|\cdot\|_1$ is the l_1-norm, $\|\cdot\|_2$ is the l_2-norm, and β is the normal vector of the hyperplane H satisfying the constraint

$$\beta^T x_i - 1 = 0, \quad i = 1, \ldots, d. \quad (4)$$

However, if the projection of x onto H is not in $\text{conv}(S)$, then d_c can be computed as

$$d_c(x, \text{conv}(S)) = \min_i \|x - x_i\|_2. \quad (5)$$

The above process of computing the distance between a point and a hyperplane is presented in Algorithm 2. The method of distance calculation in Steps 5 and 15 in Algorithm 1 (CHVS1) can be replaced by Algorithm 2, and the correspondingly modified version of Algorithm 1 is named CHVS2.

Algorithm 2 Computation of the Distance Between a Point and a Hyperplane

Input: $S = \{x_i, i = 1, \ldots, n\}$, x.
1: Let $A = (x_1, .., x_n)$, $b = (1, \ldots, 1)^T$, where $\|b\|_1 = n$.
2: Let $\beta = A^{-T}b$, $x_0 = (1/n)\sum_{i=1}^n x_i$.
3: Let $z_0 = x - x_0$, $z_i = x_i - x_0$, $i = 1, \ldots, n$
4: Let $\widetilde{x} = x - x_1$, $k = 0$.
5: **for** $i = 1; i \leq n; i++$ **do**
6: **if** $z_0^T z_i \geq 0$ **then**
7: $k = k + 1$.
8: **end if**
9: **end for**
10: **if** $k < n$ **then**
11: $d_c(x, S) = \min_{i \in 1, \ldots, n} \|x - x_i\|_2$.
12: **else**
13: $d_p(x, S) = \|\beta^T \widetilde{x}\|_1 / \|\beta\|_2$.
14: **end if**
Output: $d_c(x, S)$, $d_p(x, S)$.

In Algorithm 2, the process of determining the location of the projection point (Steps 5–12) can be removed. Wang *et al.* [30] proved that if the furthest point to the current convex hull was selected in each step, then the projection of a point x onto the current convex hull is on the corresponding facet. This means that if the point, which is selected next, is the furthest to the current convex hull, then the projection process can be executed directly. Therefore, we just need to modify the initial steps, Steps 5 and 10 in CHVS2. In the initial steps, the furthest two samples among the training samples are used to initialize the set S. The calculation of the distance in Steps 5 and 15 in CHVS2 is replaced by the computation of d_p in Algorithm 2. The CHVS2 algorithm such modified is called CHVS3.

Though we can use the projection method directly in CHVS3, the computational time of the maximum distance in the initial steps can be very large if the dimension of data is high and the number of the data samples is very large. For simplicity, we use the random method to initialize the set S and the projection method to compute the distance between a point and a convex hull. The details are presented in Algorithm 3. It should be noted that the convex hull vertices obtained by Algorithm 3 are not the exact vertices of the convex hull, but Algorithm 3 (that is, CHVS4) is the fastest one among CHVS1–4. The differences of CHVS1–4 are presented in Table I. The effectiveness and the validity of CHVS4 will be further verified by experiments in Section IV.

C. Deleting Outliers

Outliers are generally isolated points, which are far away from most of the other samples in the same class and thus will deteriorate the performance of the SVM classifier. In this paper, a method is designed to delete the isolated samples in each class. To do this, we give a definition of an isolated point.

Algorithm 3 CHVS4

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

1: Randomly select $x_0 \in P$. Let $x_{j_0} = \arg\max_{x \in P} \|x_0 - x\|_2$ and $x_{j_1} = \arg\max_{x \in P} \|x_{j_0} - x\|_2$. Initialize $S = \{x_{j_0}, x_{j_1}\}$, $I = \{1, \ldots, d\}$ and $M = d + 1$.
2: Steps 2-3 of CHVS1.
3: **for** $i = 1; i \le d + 1; i++$ **do**
4: Compute $m_i = \max_{x \in P_i} d_p(x, S_i)$ by Algorithm 2 and $x_i^* = \arg\max_{x \in P_i} d_p(x, S_i)$, where $x_i^{(t)}$ is the ith element of S_i.
5: **end for**
6: Steps 7-13 of CHVS1.
7: **for** $i = 1, i \le d, i++$ **do**
8: Compute $m_{M+i} = \max_{x \in P_{M+i}} d_p(x, S_{M+i})$ and $x_{M+i}^* = \arg\max_{x \in P_{M+i}} d_p(x, S_{M+i})$.
9: **end for**
10: Steps 17-19 of CHVS1.

Output: S.

TABLE I
DIFFERENCES OF CHVS1–4. RANDOM: RANDOM METHOD IS USED TO GENERATE THE INITIAL ELEMENTS OF THE SET S (SEE STEP 1 OF ALGORITHM 1). ACTUAL: FURTHEST TWO POINTS OF THE TRAINING SET ARE TAKEN AS THE INITIAL ELEMENTS OF THE SET S

Method	Initialization	Method of Computing Convex Hull Distance
CHVS1	Random	Quadratic Programming
CHVS2	Random	d_c of Algorithm 2
CHVS3	Actual	d_p of Algorithm 2
CHVS4	Random	d_p of Algorithm 2

Given a set $S = \{x_i\}_{i=1}^N \subset \mathbb{R}^d$, a data point $x \in S$ is said to be an isolated point if the condition $|P_\delta| < n$ is satisfied for a small positive number δ, where n is a given positive integer, $P_\delta = \{x_j | d(x_j, x) < \delta\} \subset S$, and $|P_\delta|$ is the number of the elements of the set P_δ.

The process of deleting the outliers is presented in Algorithm 4 and called Deoutlier. Algorithm 4 will be executed in the initial step of the convex hull selection process. This means that if we obtain the initial set $S = \{x_{j_0}, x_{j_1}\}$, then we execute Deoutlier(x_{j_0}) and Deoutlier(x_{j_1}). If either x_{j_0} or x_{j_1} is deleted in this process, the set S will be initialized again from the remaining samples. The Deoutlier process will be executed on S until there is no outlier in S under the conditions. For convenience, the CHVS4 algorithm with the Deoutlier process is called CHVS5.

Algorithm 4 Deoutlier(x)

Input: $S = \{x_i, i = 1, \ldots, N\}$, δ, n, and $x \in S$.

1: $k = 0$.
2: **for** $i = 1; i \le N; i++$ **do**
3: $d_i = \|x_i - x\|_2$.
4: **if** $d_i < \delta$ **then**
5: $k = k + 1$.
6: **end if**
7: **end for**
8: **if** $k < n$ **then**
9: Delete x.
10: **end if**

D. Dimension Conversion

For linearly separable problems, CHVS1–4 can be used directly. However, for nonlinearly separable problems, the process of the CHVS is not applicable in the original space. This difficulty can be overcome by the kernel trick [28] to map the data into a feature space, where the data are linearly separable. However, the dimension of the feature space may be infinite, so the algorithms CHVS1–4 cannot be applied in the feature space directly. We will first use the dimension conversion technique introduced in [30] to convert the samples in the feature space into a finite dimensional space and then apply the algorithms CHVS1–4 to the converted samples in the finite dimensional space.

Let $\phi(\cdot)$ be the mapping from $\mathbb{R}^d$ to the feature space $\mathcal{H}$. The kernel function $k(x, x')$ in $\mathbb{R}^d \times \mathbb{R}^d$ is then given by

$$k(x, x') = \phi(x)^T \phi(x').$$

By using the dimension conversion Algorithm 4 of [30], each sample $\phi(x_i)$ in the feature space can be converted into a new sample z_i in the d_z-dimensional space, where d_z is obtained in [30, Algorithm 4] and $i \in \{1, \cdots, n\}$. It was proved in [30, Th. 4] that, for the Gaussian kernel, the dimension d_z of the converted samples $\{z_1, \cdots, z_n\}$ is finite. The algorithms CHVS1–4 will then be applied to the converted samples $\{z_1, \cdots, z_n\}$ in the d_z-dimensional space. It was shown in [30, Th. 3] that the index set of vertices of both $\text{conv}(\{\phi(x_i)\}_{i=1}^n)$ and $\text{conv}(\{z_i\}_{i=1}^n)$ is the same.

Let H be the hyperplane involving the points $\{z_i\}_{i=1}^n$ in the d_z-dimensional space. Similar to (4), we have

$$\beta^T z_i - 1 = 0, \quad i = 1, \ldots, n \tag{6}$$

where $\beta = \sum_{i=1}^n \alpha_i z_i$ is the normal vector to H and α_i is the coefficient to be determined.

We now show that (6) can be solved directly using the kernel function and the d_z-dimensional samples $\{z_1, \cdots, z_n\}$.

Let $U^z = (u_{j_1}, \cdots, u_{j_{d_z}})T$, where u_{j_t}, $t = 1, \cdots, d_z$, and T are obtained in [30, Algorithm 4]. From [30, Algorithm 4, Th. 3], it follows that $\phi(x_i) = U^z z_i$ and $(U^z)^T U^z = I$, so $(U^z Z)^T U^z Z = Z^T Z$, where $Z = (z_1, \cdots, z_n)$ and I is the identity matrix of order d_z. Let $A = (K_{ij})_{n \times n}$, where $K_{ij} = k(x_i, x_j) = \phi(x_i)^T \phi(x_j)$. Then, we have $A = (U^z Z)^T U^z Z = Z^T Z$.

Let $\alpha = (\alpha_1, \ldots, \alpha_n)^T$, $\mathbf{1} = (1, \ldots, 1)^T$. Then, (6) can be rewritten as

$$A\alpha = \mathbf{1}. \tag{7}$$

Furthermore, let $\mathbf{a} = (k(x, x_1), \ldots, k(x, x_n))^T$ and $\mathbf{a}_1 = (k(x_1, x_1), \ldots, k(x_1, x_n))^T$. By the definition of k, and since $\phi(x_i) = U^z z_i$, we have $\mathbf{a}_1 = Z^T z_1$ and $\mathbf{a} = Z^T z$ for

Algorithm 5 Computing Distance Between a Point and a Hyperplane via the Kernel Trick

Input: $S = \{x_i, i = 1, \ldots, n\}, x$;
$H = \{z_i, i = 1, \cdots, n\}, z$.
1: Let $A = (K_{ij})_{n\times n}$, where $K_{ij} = k(x_i, x_j)$.
2: Let $\alpha = (\alpha_1, \ldots, \alpha_n)^T$, $\mathbf{1} = (1, \ldots, 1)^T$.
3: Let $\alpha = A^{-1}\mathbf{1}$.
4: Let $\mathbf{a} = (k(x, x_1), \ldots, k(x, x_n))^T$,
$\mathbf{a}_1 = (k(x_1, x_1), \ldots, k(x_1, x_n))^T$.
5: $d_h(z, H) = \|\alpha^T(\mathbf{a}\text{-}\mathbf{a}_1)\|_1(\alpha^T A\alpha)^{-1/2}$.
Output: $d_h(z, H)$.

any $x \in \{x_1, \cdots, x_n\}$ satisfying that $\phi(x) = U^z z$ for some $z \in \{z_1, \cdots, z_n\}$. Then

$$\alpha^T(\mathbf{a}\text{-}\mathbf{a}_1) = \alpha^T Z^T(z - z_1) = \beta^T(z - z_1).$$

Now, suppose α^* is the solution of the equation (7). Then, the distance between z and H can be computed by

$$\begin{aligned} d_h(z, H) &= \|\beta^{*T}(z - z_1)\|_1/\|\beta^*\|_2 \\ &= \|\alpha^{*T}(\mathbf{a}\text{-}\mathbf{a}_1)\|_1(\alpha^{*T} A\alpha^*)^{-1/2} \end{aligned} \tag{8}$$

where $\beta^* = Z\alpha^*$, so $\|\beta^*\|_2 = (\alpha^{*T} A\alpha^*)^{1/2}$. The above process is presented in the following algorithm.

E. Termination Condition

For linearly separable problems, the algorithms CHVS1–4 will be terminated when all the convex hull vertices are selected. However, if the dimension of the data is high, then the number of the convex hull vertices is too large to select. In this case, we use an approximate convex hull. The parameter ϵ in the proposed algorithms is used to control the approximation degree of the current convex hull. The algorithms CHVS1–4 will then be terminated when the maximum distance between the data and the current convex hull is smaller than ϵ. The method of choosing the appropriate parameter ϵ is presented as follows.

Step 1: Given a small step size δ, the training data $S_1 = S_1^+ \cup S_1^-$ and the validation data $S_2 = S_2^+ \cup S_2^-$, where S_1^+ and S_2^+ are the sets of positive data and S_1^- and S_2^- are the sets of negative data. Set $k = 1$, $\mathrm{TA}_0 = 0$, and $\epsilon_1 = 0$.

Step 2: Execute CHVS1–4 with data $S_1^\pm$ and obtain the set of convex hull vertices, $V_1^\pm$.

Step 3: Train an SVM classifier with data $V = V_1^+ \cup V_1^-$ and obtain the SVM classifier C.

Step 4: Test the validation data S_2 with the classifier C and obtain the test accuracy TA_k.

Step 5: If $\mathrm{TA}_k < \mathrm{TA}_{k-1}$, then go to Step 7. Otherwise, go to Step 6.

Step 6: Set $k := k + 1$ and $\epsilon_k := \epsilon_{k-1} + \delta$. Go to Step 2.

Step 7: Select ϵ_{k-1} as the best value of ϵ.

Detailed discussion on the impact of ϵ on the performance of the proposed algorithm will be presented in Section IV-C.

For nonlinearly separable problems, the algorithms CHVS1–4 will be implemented on the converted d_z-dimensional samples $\{z_1, \cdots, z_n\}$ obtained by the dimension conversion [30, Algorithm 4]. The termination condition of the algorithms is then similar to the case for linearly separable problems.

F. Theoretical Analysis

Let $x_i \in \mathbb{R}^d, i = 1, \cdots, l$, be the training samples and let $y_i \in \{-1, 1\}$ be the label of x_i. The primal problem of SVM is

$$\min_{w,b} \; F_1(w, b) = \frac{1}{2}\|w\|^2 + \frac{C}{l}\sum_{i=1}^{l} V(y_i f(x_i)) \tag{9}$$

where w is the normal vector of the hyperplane, $V(y_i f(x_i)) = \max\{0, 1 - y_i f(x_i)\}$, $f(x_i) = w^T\phi(x_i) + b$, b is the bias term, $\phi(x_i)$ is the mapping from the input space to the feature space, and C is the penalty factor.

For simplicity, we assume that $\phi(x_k), k = 1, \ldots, m$, are the approximate convex hull vertices of the training samples in the feature space. Then, the primal problem of SVM based on the vertices $\phi(x_k), k = 1, \ldots, m$, is

$$\min_{w,b} \; F_2(w, b) = \frac{1}{2}\|w\|^2 + \frac{C}{l}\sum_{i=1}^{m} V(y_i f(x_i)). \tag{10}$$

For any $i \in \{m+1, \cdots, l\}$ let γ_{it} be such that $0 \le \gamma_{it} \le 1$, $\sum_{z_t \in Z^*} \gamma_{it} = 1$ and

$$z_i = \sum_{z_t \in Z^*} \gamma_{it} z_t + \tau_i \tag{11}$$

where $z_i = \phi(x_i), i = m + 1, \ldots, l$, $Z^* = \{z_t = \phi(x_t) | t = 1, \ldots, m\}$ is the set of the approximate convex hull vertices obtained in the Sections III-A and III-B. τ_i is a vector that accounts for the approximation error of the approximate convex hull and satisfies that

$$\|\tau_i\|^2 \le \epsilon, \quad i = m + 1, m + 2, \cdots, l \tag{12}$$

for some small constant $\epsilon > 0$. The relation (11) means that all the training samples z_i, $i = 1, \cdots, l$, in the feature space can be approximately expressed in terms of the approximate convex hull vertices z_t, $t = 1, \cdots, m$. Since the sample z_i is expressed in terms of the samples from the same class, we conclude that if $y_i y_t < 0$ then $\gamma_{it} = 0$.

Suppose the SVM problems (9) and (10) are linearly separable in the feature space. Then, we have the following results.

Theorem 1: Let (w_2^*, b_2^*) be the solution of the problem (10). If the condition (12) is satisfied, then

$$F_1(w_2^*, b_2^*) - F_2(w_2^*, b_2^*) \le \frac{l - m}{l} C\sqrt{C\epsilon}.$$

Proof: From the definition of F_1 and F_2, it follows that:

$$F_1(w_2^*, b_2^*) - F_2(w_2^*, b_2^*) = \frac{C}{l}\sum_{i=m+1}^{l} V(y_i f^*(x_i))$$

where $f^*(x_i) = w_2^{*T}\phi(x_i) + b_2^* = w^{*T} z_i + b_2^*$. Since the problem (10) is linearly separable in the feature space and (w_2^*, b_2^*) is the solution of the problem (10), we have

$$1 - y_i(w_2^{*T} z_i + b_2^*) \le 0 \quad \text{for} \quad i = 1, \ldots, m.$$

This, together with the relation (11), implies that for $i = m+1, \ldots, l$

$$\begin{aligned}
&1 - y_i(w_2^{*T} z_i + b_2^*) \\
&\quad = 1 - y_i \left(w_2^{*T} \sum_{z_t \in Z^*} \gamma_{it} z_t + w_2^{*T} \tau_i + b_2^* \right) \\
&\quad = 1 - y_i \left(\sum_{z_t \in Z^*} \gamma_{it} w_2^{*T} z_t + b_2^* \right) - y_i w_2^{*T} \tau_i \\
&\quad = \sum_{z_t \in Z^*} \gamma_{it} \left[1 - y_i \left(w_2^{*T} z_t + b_2^* \right) \right] - y_i w_2^{*T} \tau_i \\
&\quad \le -y_i w_2^{*T} \tau_i \le \|w_2^*\| \|\tau_i\|.
\end{aligned}$$

Now, arguing similarly as in the proof of [21, Th. 1], we can obtain that $\|w_2^*\| \le \sqrt{mC/l} \le \sqrt{C}$, so

$$\begin{aligned}
\frac{C}{l} \sum_{i=m+1}^{l} V\left(y_i f^*(x_i)\right) &= \frac{C}{l} \left(1 - y_i \left(w_2^{*T} z_i + b_2^*\right)\right) \\
&\le \frac{C}{l} \sum_{i=m+1}^{l} \left\|w_2^*\right\| \|\tau_i\| \\
&\le \frac{C}{l} \sum_{i=m+1}^{l} \sqrt{C\epsilon} \\
&= \frac{l-m}{l} C \sqrt{C\epsilon}.
\end{aligned}$$

The proof is thus complete. ∎

Based on Theorem 1, the following theorem can be derived.

Theorem 2: Let (w_1^*, b_1^*) and (w_2^*, b_2^*) be the solutions of the problems (9) and (10), respectively. If the condition (12) is satisfied, then

$$F_1\left(w_2^*, b_2^*\right) - F_1\left(w_1^*, b_1^*\right) \le \frac{l-m}{l} C \sqrt{C\epsilon}.$$

Proof: By Theorem 1 and noting that

$$F_2\left(w_2^*, b_2^*\right) - F_1\left(w_1^*, b_1^*\right) \le F_2\left(w_1^*, b_1^*\right) - F_1\left(w_1^*, b_1^*\right) \le 0$$

we have

$$\begin{aligned}
&F_1\left(w_2^*, b_2^*\right) - F_1\left(w_1^*, b_1^*\right) \\
&\quad = F_1\left(w_2^*, b_2^*\right) - F_2\left(w_2^*, b_2^*\right) + F_2\left(w_2^*, b_2^*\right) - F_1\left(w_1^*, b_1^*\right) \\
&\quad \le \frac{l-m}{l} C \sqrt{C\epsilon}.
\end{aligned}$$

∎

Theorem 2 gives an upper bound of the difference between the solution of the problem (8) which is based on the original training set and the solution of the problem (9) which is based on the approximate convex hull vertices under the measure of function F_1. Since F_1 is convex and continuous, Theorem 2 indicates that the solution (w_2^*, b_2^*) of the proposed method is a good approximation to the solution (w_1^*, b_1^*) of the primal SVM problem.

IV. Experimental Results

In this section, the effectiveness and the validity of the proposed algorithms are demonstrated by several experiments on synthetic and real-world data. The efficiency of the CHVS algorithms is compared in Section IV-A. In Section IV-B, the classification ability is tested by applying the proposed algorithms to several linear classification tasks. In Section IV-C, the relation between the number of the convex hull vertices and the value of ϵ is explored through several experiments on a real-word data set. Section IV-D gives an analysis on the impact of the kernel parameters. The proposed algorithms are used for solving several online classification problems in Section IV-E.

All experiments are performed on a Desktop PC with Inter Core 3.2 GHz CPU, 4G RAM and Windows 7 operating system. All algorithms are implemented in MATLAB. The parameters of the SVM classifier are obtained via a tenfold cross validation. The parameter n in the Deoutlier algorithm is set to be 3, and the parameter δ in the Deoutlier algorithm is chosen by using the following process.

Step 1: Denote by $d_{\max}$ and $d_{\min}$ the maximum and minimum distances between the data set, respectively, and let $D = \{d_{\min} + i(d_{\max} - d_{\min})/N | i = 1, 2, \ldots, N\}$, where $N = 20$ in our experiments.

Step 2: Randomly select an element from D as the value of δ (the parameter in the Deoutlier algorithm).

Step 3: To use cross validation to choose the parameter, the data set is randomly divided into ten equal parts B_i, $i = 1, \ldots, 10$, and set $F = \{1, 2, \ldots, 10\}$.

Step 4: Randomly select an element from F which is denoted by t; let B_t be the validation set and let the union of the remaining nine sets be the training set; delete t from the set F.

Step 5: An SVM classifier is trained on the training set with deleting the outliers by the Deoutlier algorithm; the SVM classifier is tested on the validation set and the error rate e_t is calculated.

Step 6: Repeat steps 4) and 5) ten times and calculate the total error rate $e_\delta = \sum_{t=1}^{10} e_t$.

Step 7: Choose the δ with the minimal error rate e_δ as the best value of the parameter δ of the Deoutlier algorithm.

A. Efficiency of the CHVS Algorithms

To test the efficiency of our CHVS algorithms, CHVS1–4 are compared with the CHVS algorithm of [30] (denoted by CHVS) and the Qhull function of MATLAB. These algorithms are executed on several different data sets: 1) a d-dimensional data set of 500 samples generated from the d-dimensional unitary uniform distribution and 2) a d-dimensional data set of 500 samples generated from the d-dimensional standard Gaussian distribution, where $d = 2, 4, \ldots, 10$. In CHVS1–4, the value of ϵ is set to be zero. The results are presented in Fig. 4 and Tables II and III.

From the results, we have the following conclusions.

1) CHVS4 has the least time cost among these methods when the dimension is greater than 7.

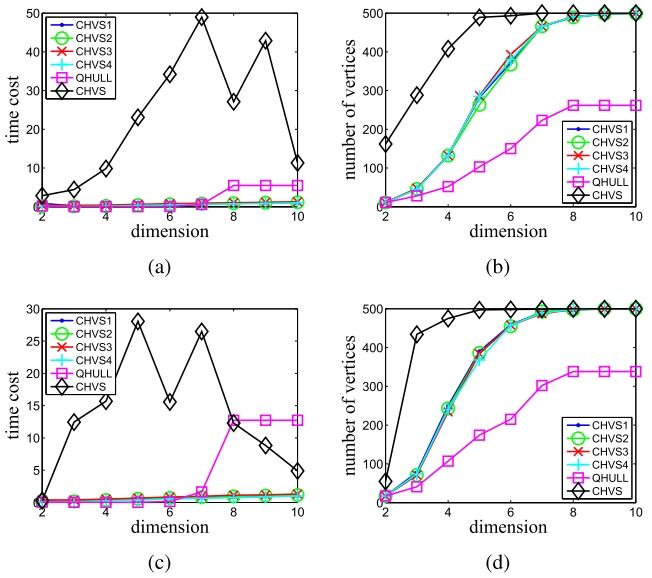


Fig. 4. Efficiency of different methods. The data used in (a)–(b) obeys a uniform distribution, while the data used in (c)–(d) obeys a Gaussian distribution. (a) and (c) Time cost of different methods as the dimension of the data changes. (b) and (d) Number of vertices selected by different methods as the dimension of the data changes.

TABLE II

TIME COST OF DIFFERENT METHODS. D-u: SAMPLES GENERATED FROM D-DIMENSIONAL UNIFORM DISTRIBUTIONS. D-g: SAMPLES GENERATED FROM D-DIMENSIONAL GAUSSIAN DISTRIBUTIONS. NULL: OUT OF MEMORY

Date Sets	CHVS1	CHVS2	CHVS3	CHVS4	QHULL	CHVS
2-u	0.8738	0.0975	0.3857	**0.0921**	0.1808	2.8389
4-u	0.2096	0.2176	0.4225	0.1340	**0.0220**	9.8567
6-u	0.5848	0.6106	0.7956	0.4787	**0.0682**	34.2022
8-u	0.8651	0.9045	1.0398	**0.7435**	5.4902	27.1196
10-u	1.0864	1.1424	1.2686	**0.9895**	NULL	11.3202
2-g	0.1861	0.1067	0.3910	0.0864	**0.0532**	0.2837
4-g	0.3063	0.3662	0.5157	0.2230	**0.0042**	15.6905
6-g	0.6104	0.6954	0.8409	0.5268	**0.1484**	15.5610
8-g	0.8334	0.9386	1.1236	**0.7539**	12.7314	12.2921
10-g	1.0530	1.1622	1.3066	**1.0024**	NULL	4.8788

2) The Qhull function selected the least number of vertices, but it is very expensive when the dimension of the data is greater than 7.
3) The number of convex hull vertices selected by CHVS1–4 is less than that selected by CHVS, so the online updating process of the SVM based on CHVS1–4 is faster than that of the SVM based on CHVS (that is, VS-OSVM in [30]).
4) CHVS1–4 are more stable compared with CHVS and Qhull.
5) CHVS4 is the fastest one among these methods, since solving a system of linear equations is faster than solving a quadratic programming problem.

In Section IV-B, we will evaluate the classification ability of the CHVS4-based SVM classifier.

B. Test Accuracy

In this section, four methods are used to select the convex hull vertices in the original space: CHVS4, CHVS5 (that is, CHVS4 with deleting the outliers), CHVS, and Qhull. Four data sets are used in the experiments: 3-U (a 3-D data set of 4000 samples generated from a 3-D uniform distribution), 6-U (a 6-D data set of 4000 samples generated from a 6-D uniform distribution), 3-G (a 3-D data set of 4000 samples generated from a 3-D Gaussian distribution), and 6-G (a 6-D data set of 4000 samples generated from a 6-D Gaussian distribution). In addition, Gaussian noise is added to the data sets 3-U and 6-U.

TABLE III

NUMBER OF VERTICES SELECTED BY DIFFERENT METHODS. D-u: SAMPLES GENERATED FROM D-DIMENSIONAL UNIFORM DISTRIBUTIONS. D-g: SAMPLES GENERATED FROM D-DIMENSIONAL GAUSSIAN DISTRIBUTIONS. NULL: OUT OF MEMORY

Data Sets	CHVS1	CHVS2	CHVS3	CHVS4	QHULL	CHVS
2-u	11	11	11	11	11	162
4-u	133	132	132	132	52	408
6-u	372	367	393	380	150	493
8-u	489	489	491	490	262	500
10-u	500	499	500	500	NULL	500
2-g	17	17	17	17	17	55
4-g	250	243	234	240	107	475
6-g	459	455	458	458	215	498
8-g	497	497	498	498	338	500
10-g	500	500	500	500	NULL	500

The experiments consist of four steps: 1) randomly dividing the data sets into two equal sets: the training sets and testing sets; 2) selecting the convex hull vertices of each class from the training sets; 3) training a linear SVM classifier with the selected samples; and 4) testing the classifier with the testing sets. The above steps are executed ten times, and the results are presented in Table IV.

From Table IV, it is seen that among the four compared methods the CHVS5-based SVM has the highest test accuracy and the CHVS-based SVM consumes the most time. In general, CHVS5 can delete some outliers, so the CHVS5-based SVM is more efficient than the CHVS4-based SVM and the CHVS-based SVM. Furthermore, CHVS5 is faster than CHVS, since the projection method is used to compute the distance between a point and a convex hull. Qhull is the fastest algorithm for the four data sets; however, the Qhull-based SVM has the worst classification ability among these methods. In conclusion, CHVS5 is the most effective algorithm.

C. Impact of Parameter ϵ

If the data set is large and the data distribution is complicated, most of the samples may be the convex hull vertices. To reduce more samples in the off-line process, an approximate convex hull should be selected. In this section, the parameter ϵ in CHVS4 and CHVS5 will be used to select an appropriate convex hull. The impact of the parameter ϵ on the effectiveness of the two algorithms will be studied through several experiments on the real data set Cod-rna. Cod-rna is a biomedical data set introduced in [27],

TABLE IV

EFFICIENCY OF DIFFERENT METHODS. SE#: NUMBER OF SELECTED SAMPLES. PR: PROPORTION OF THE SELECTED SAMPLES. TC: TIME COST OF THE VERTICES SELECTION. TA: TEST ACCURACY. D-U: SAMPLES GENERATED FROM D-DIMENSIONAL UNIFORM DISTRIBUTIONS. D-G: SAMPLES GENERATED FROM D-DIMENSIONAL GAUSSIAN DISTRIBUTIONS

Method	CHVS4-based SVM				CHVS5-based SVM			
	SE#	PR(%)	TC(s)	TA(%)	SE#	PR(%)	TC(s)	TA(%)
3-U	109±7	5.5±0.3	0.2284±0.0148	99.31±0.17	198±12	9.9±0.6	0.3568±0.0254	**99.46±0.19**
6-U	1363±40	68.2±2.0	2.0882±1.3113	100±0	1486±24	74.3±0.1	1.8258±0.0996	**100±0**
3-G	142±14	7.2±0.7	0.3104±0.0084	94.38±1.16	314±18	15.7±0.9	0.4980±0.0288	**95.61±0.35**
6-G	1358±26	68.9±1.3	1.6240±0.0724	99.19±0.1667	1490±22	74.5±1.1	1.7401±0.0521	**99.21±0.2166**
Method	CHVS-based SVM				QHULL-based SVM			
	SE#	PR(%)	TC(s)	TA(%)	SE#	PR(%)	TC(s)	TA(%)
3-U	730±53	36.5±2.76	8.6207±3.2060	99.42±0.16	63±4	3.1±0.2	0.0060±0.0115	98.94±0.73
6-U	1914±20	95.7±1.0	114.60±52.96	100±0	436±19	21.8±0.9	0.2591±0.0759	99.98±0.02
3-G	1032±162	51.6±8.1	20.004±11.904	94.88±0.96	75±7	3.7±0.3	0.0047±0.0050	94.35±1.47
6-G	1920±14	96.0±0.7	178.09±195.35	99.17±0.1619	426±18	21.3±0.9	0.2240±0.0393	99.15±0.2303

TABLE V

CHVS4 ON "cod-rna." SE#: NUMBER OF SELECTED SAMPLES. PR: PROPORTION OF THE SELECTED SAMPLES. TA: TEST ACCURACY. TC: TIME COST OF TRAINING SVM CLASSIFIER. SC: TIME COST OF SELECTING VERTICES. $\epsilon = 0$: THE METHOD OF LIBSVM

ϵ	SE#	PR(%)	TA(%)	TC(s)	SC(s)
0	244283	100	94.7184	53379.127	0
0.0002	8877	3.6339	94.4380	840.6592	70.7153
0.0009	1624	0.6648	93.9173	117.6786	56.3983
0.0016	585	0.2395	93.1870	33.4743	53.2030
0.0023	306	0.1253	87.4412	17.1131	51.1704
0.003	191	0.0782	83.8027	9.0845	50.8680
0.0037	110	0.0450	82.7351	0.9410	49.4147
0.0044	63	0.0258	77.6038	4.1379	49.0777
0.0051	54	0.0221	81.4542	1.0331	48.0669

TABLE VI

CHVS5 ON "cod-rna." SE#: NUMBER OF SELECTED SAMPLES. PR: PROPORTION OF THE SELECTED SAMPLES. TA: TEST ACCURACY. TC: TIME COST OF TRAINING SVM CLASSIFIER. SC: TIME COST OF SELECTING VERTICES. $\epsilon = 0$: THE METHOD OF LIBSVM

ϵ	SE#	PR(%)	TA(%)	TC(s)	SC(s)
0	244283	100	94.7184	53379.127	0
0.0002	9931	4.0654	94.8068	908.9844	79.2378
0.0009	2042	0.8359	94.8715	160.8947	65.3094
0.0016	846	0.3463	92.7739	51.6903	60.5605
0.0023	441	0.1805	90.4074	24.3850	56.2977
0.003	247	0.1011	78.6890	5.0869	54.7462
0.0037	94	0.0385	74.4230	2.1257	53.5846
0.0044	105	0.0430	60.5706	1.5034	53.5376
0.0051	57	0.0233	85.5470	0.8845	52.5639

which includes 488 585 samples with eight attributes. The classification task is to detect whether or not a sample is RNA of cod fish. We randomly decompose the data set into two equal parts: the training and testing samples.

The results are obtained by executing Steps 1–6 of Section III-E and presented in Tables V and VI. The case $\epsilon = 0$ represents that the SVM classifier is trained with all the training samples.

From Tables V and VI, it can be seen that with the value of ϵ increasing, the number of the selected vertices decreases quickly. It is also found that when the number of the selected vertices is very small, the SVM classifier is not reliable due to its low test accuracy and unstable performance. The reason for this is that when the number of the selected vertices is very small, the selected vertices may not include all the most important samples for the SVM classifier. In order to determine the best value for ϵ, we use the method developed in Section III-E in our experiments. From Tables V and VI, it is found that 0.0009 is the best value of ϵ for CHVS4 and CHVS5 on the data set Cod-rna.

D. Impact of the Kernel Parameters

We now analyze the impact of the kernel parameters on two data sets, the Letter database [2] and the a3a database [7]. The classes "A" and "B" of the Letter database are used for our experiments. In addition, the Gaussian kernel $k(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$ and the polynomial kernel $k(x_i, x_j) = (x_i x_j)^d$ are used for the Letter and a3a databases, respectively. The data are standardized, and the termination condition parameter ϵ is set to be 10^{-4}. We randomly decompose the data set into two equal parts: the training and testing samples. The results are presented in Tables VII and VIII.

From Table VII, it can be seen that the number of the selected samples increases with γ increasing. This is because the distance between the training samples increases when γ increases. Table VIII presents the impact of the parameter d of the polynomial kernel. It can be observed that the number of samples selected by CHVS4 decreases as d increases. In addition, the results show that the algorithm Deoutlier has a positive impact on the test accuracy by comparing CHVS5

TABLE VII

EXPERIMENT ON "LETTER." SE#: NUMBER OF SELECTED SAMPLES. PR: PROPORTION OF THE SELECTED SAMPLES. TA: TEST ACCURACY. TC: TIME COST OF TRAINING SVM CLASSIFIER. SC: TIME COST OF SELECTING VERTICES

a	CHVS4					CHVS5				
γ	SE#	PR(%)	TA(%)	TC(s)	SC(s)	SE#	PR(%)	TA(%)	TC(s)	SC(s)
0.0032	56	7.2979	51.9949	0.0031	0.1500	167	21.4650	48.0051	0.0018	0.7795
0.0100	73	9.3830	51.9949	0.0008	0.1833	286	36.7610	94.4659	0.0044	2.5114
0.0316	141	18.1230	51.9949	0.0015	0.7719	571	73.3930	97.0399	0.0058	13.0884
0.1000	299	38.4320	99.3565	0.0033	3.0250	758	97.4290	100	0.0055	22.8254
0.3162	463	59.5120	51.9949	0.0081	12.1324	758	97.4290	99.7426	0.0276	28.4369
1	518	66.5810	48.0051	0.0352	8.2714	758	97.4290	62.8057	0.0671	23.4524

TABLE VIII

EXPERIMENT ON "a3a." SE#: NUMBER OF SELECTED SAMPLES. PR: PROPORTION OF THE SELECTED SAMPLES. TA: TEST ACCURACY. TC: TIME COST OF TRAINING SVM CLASSIFIER. SC: TIME COST OF SELECTING VERTICES

a	CHVS4					CHVS5				
d	SE#	PR(%)	TA(%)	TC(s)	SC(s)	SE#	PR(%)	TA(%)	TC(s)	SC(s)
2	781	49.0270	81.2186	0.0315	6.0648	810	50.8475	82.0980	0.0322	6.4617
3	269	16.8864	74.8744	0.0061	4.0607	332	20.8412	74.8744	0.0076	4.4091
4	269	16.8864	25.1256	0.0091	3.6226	244	15.3170	74.8744	0.0072	4.1939
8	4	0.2511	61.6206	0.0005	0.0291	363	22.7872	74.8744	0.0138	3.6667
16	4	0.2511	60.5528	0.0004	0.0254	379	23.7916	74.8744	0.0151	4.1274

TABLE IX

OVERALL DESCRIPTION OF THE DATA SETS USED AND RESULTS OF SAMPLES SELECTION BY THE PROPOSED ALGORITHM. C IS THE PENALTY PARAMETER, σ IS THE PARAMETER OF THE GAUSSIAN KERNEL $k(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$, AND d IS THE PARAMETER OF THE POLYNOMIAL KERNEL $k(x_i, x_j) = (x_i x_j)^d$. "DIM.": 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, AND "PR": PROPORTION OF SELECTED SAMPLES

Data Sets	Dim.	C	γ	d	ϵ	TR#	EX#	TE#	SE#	PR(%)
Ionosphere	34	1274	0.7848	-	10^{-5}	176	35	140	46	26.1±0
Circle2	2	0.5456	0.2637	-	10^{-1}	2000	400	1600	25±6	1.2±0.3
Satellite	36	1.1288	0.0004	-	10^{-3}	1080	215	864	150±6	13.9±0.6
Letter	16	0.7848	0.0428	-	$10^{-3.5}$	778	155	622	120±0	15.4±0
Mushroom	112	0.7848	0.2637	-	$10^{-2.5}$	4062	815	3247	232±22	5.7±0.5
Shuttle	9	12.7427	0.0001	-	10^{-3}	6085	1215	4870	197±25	3.2±0.4
a3a	123	1	-	3	10^{-2}	1593	320	1272	743±252	46.7±15.8
KDDCUP08	117	0.0001	-	2	$10^{-0.5}$	51147	10230	40917	1800±0	3.5±0

with CHVS4. In our experiments, the best parameters γ and d are chosen by cross validation and presented in Table IX.

E. Efficiency for Online Learning

In this section, we evaluate the efficiency of the proposed algorithms for online learning. All the experiments are carried out ten times independently for all data sets, and the classification results are presented in terms of mean and standard deviation. Each experiment is divided into four steps.

Step 1 (Data Preprocessing): The data are randomly divided into three sets: the training, expanded, and testing sets with a fixed proportion $5:1:4$.

Step 2 (Samples Selection): Four algorithms, CHVS4, CHVS5, random selection (RS) and CHVS, are used to select samples. To compare the four algorithms fairly, the number of the vertices selected is forced to be the same, and the time cost is recorded.

Step 3 (Updating): FVS1-OSVM (the online version of SVM based on CHVS4), FVS2-OSVM (the online version of SVM based on CHVS5), LIBSVM [7] (updating the classifier in batch mode based on both the old samples and the new samples), VS-OSVM [30], LASVM [4], and RS-OSVM (the online version of SVM based on the RS method) are used to update the classifier with the expanded samples. The expanded set is equally divided into ten subsets, and the classifiers are then updated with one subset at a time.

Step 4 (Testing): After the updating process, we apply the classifiers to the classification tasks on the testing sets.

TABLE X

TIME COST (S) OF THE OFF-LINE SAMPLE SELECTION PROCESS OF FVS1-OSVM, FVS2-OSVM, VS-OSVM, AND RS-OSVM, WHERE THE MEAN TIME COST TOGETHER WITH STANDARD DEVIATION IS REPORTED

Data Sets	FVS1-OSVM	FVS2-OSVM	VS-OSVM	RS-OSVM
Ionosphere	0.015±0.004	0.026±0.001	0.034±0.001	0.0001±0.0003
Circle2	0.234±0.019	0.599±0.015	1.129±0.319	0.0001±0.0002
Satellite	3.951±0.698	3.964±0.692	5.673±1.002	0.0001±0.0003
Letter	0.245±0.028	0.237±0.026	0.431±0.048	0.0001±0.0002
Mushroom	2.053±0.159	2.057±0.166	4.601±0.469	0.0001±0.0002
Shuttle	4.035±0.482	4.076±0.504	8.660±1.070	0.0001±0.0001
a3a	6.316±2.337	6.031±4.236	83.093±43.050	0.0001±0.0001
KDDCUP08	64.454±0.302	92.244±0.144	323.104±0.300	0.0001±0.0001

TABLE XI

CLASSIFICATION RATE OF FVS1-OSVM, FVS2-OSVM, VS-OSVM, RS-OSVM, LASVM, AND LIBSVM, WHERE THE MEAN RATE TOGETHER WITH STANDARD DEVIATION IS REPORTED. THE BEST CLASSIFICATION RESULTS ARE MARKED IN BOLDFACE

Data Sets	FVS1-OSVM	FVS2-OSVM	VS-OSVM	RS-OSVM	LASVM	LIBSVM
Ionosphere	**93.786±0.894**	**93.786±0.894**	**93.786±0.894**	89±2.858	91.429±0	93.571±0
Circle2	92.356±0.409	92.725±0.387	92.475±0.690	86.063±1.140	92.681±1.493	**93.419±0.0782**
Satellite	**99.850±0.056**	**99.850±0.056**	99.699±0.124	98.970±0.403	99.653±0.001	99.456±0.056
Letter	99.904±0.112	**99.920±0.114**	**99.920±0.114**	99.470±0.497	99.839±0.076	99.839±1.498e-14
Mushroom	**100±0**	**100±0**	**100±0**	99.861±0.087	99.938±0.174	**100±0**
Shuttle	**99.938±0**	**99.938±0**	**99.938±0**	99.914±0.009	99.897±0.001	**99.938±0**
a3a	81.230±3.204	**83.569±0.491**	81.663±3.081	80.759±4.476	81.745±0.804	83.33±0
KDDCUP08	99.433±0	99.450±0	99.387±0	99.367±0.021	99.407±0.094	**99.462±0**

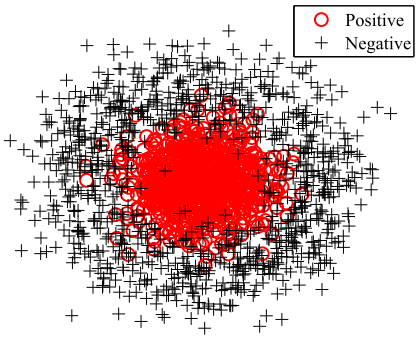


Fig. 5. Situation of a nonlinearly separable problem.

Eight nonlinearly separable data sets are used in the experiments, and the overall description of the data sets is presented in Table IX. The data set Circle2 is shown in Fig. 5, where the red points are positive samples and the black points are negative points. The data are generated from a 2-D uniform distribution in different circles, and Gaussian noise is added to the data. The other seven data sets can be download from the UCI Machine Learning Repository [2], the LIBSVM Web site [7], and the KDDCUP competition site [34]. Note that there are more than two classes in the data sets Satellite, Letter, and Shuttle. The classes "red soil" and "damp gray soil," the classes "A" and "B," and the classes "High" and "Bypass" are used in our experiments, respectively. The parameter ϵ of our algorithms is initialized to 10^{-5} and δ is set to be 0.5. And, at time T, the value of $\log_{10}\epsilon := -5 + \delta T$. The best value of ϵ can be found in Table IX.

According to Steps 2–4 of the experiment in Section IV-E, the proposed algorithms are compared with the other algorithms in three aspects: 1) the time cost of the off-line SSP (Step 2); 2) the online updating time of the classifier (Step 3); 3) the test accuracy (Step 4), as explained in the following.

The time cost of the SSP of the compared algorithms is presented in Table X. As seen from Table X, the SSP of FVS1-OSVM and FVS2-OSVM is almost twice faster than that of VS-OSVM, that is, the proposed CHVS algorithms are much faster than the CHVS process of VS-OSVM.

The online updating time of the six methods is shown in Fig. 6. It can be seen that the updating time of FVS1-OSVM and FVS2-OSVM is less than that of LIBSVM and LASVM but is almost the same as that of VS-OSVM and RS-OSVM. The reason is as follows: 1) many redundant samples are deleted in the off-line process of FVS1-OSVM and FVS2-OSVM, so the SVM based on the remaining samples is faster than the SVM based on all the samples and 2) the number of the selected samples is the same for FVS1-OSVM, FVS2-OSVM, VS-OSVM, and RS-OSVM.

The classification results of the six methods on the testing samples are shown in Table XI. It can be seen that the test accuracy of FVS2-OSVM is comparable with that of LIBSVM and is higher than that of the other four methods. This is because the samples, which keep the most information of the training samples, are selected and the outliers are deleted from the training samples in the proposed FVS2-OSVM algorithm.

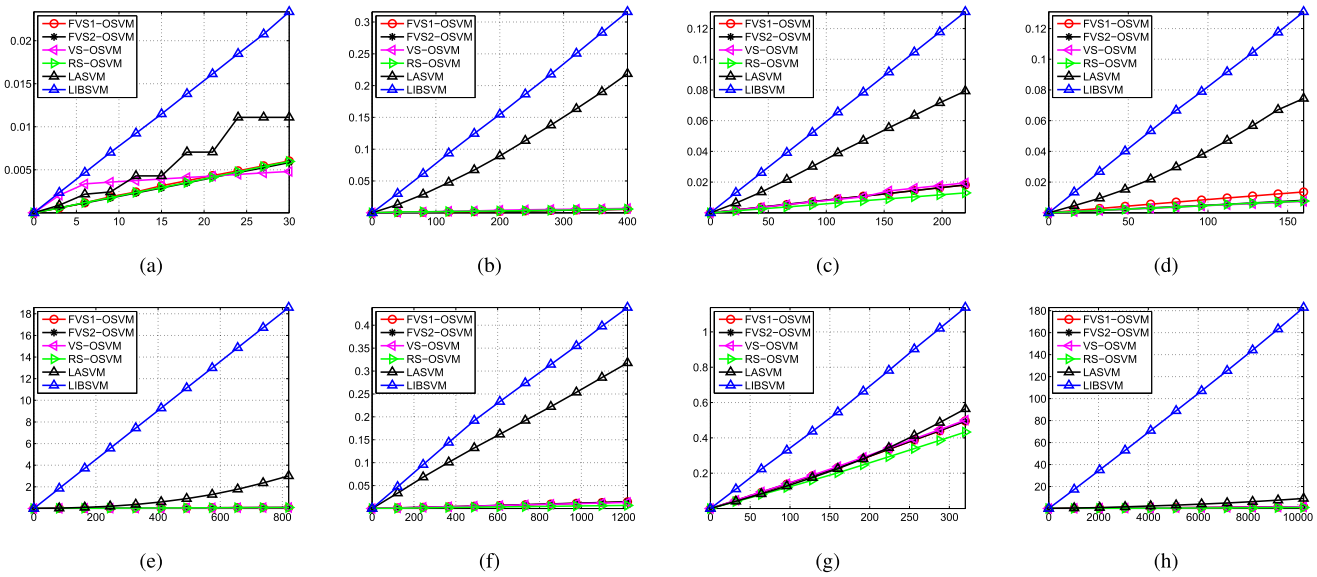


Fig. 6. Time cost of the online updating process of the six comparing algorithms. The lateral axis represents the number of expanded samples and the longitudinal axis represents the time cost (s). (a) Ionosphere. (b) Circle2. (c) Statlog. (d) Letter. (e) Mushroom. (f) Shuttle. (g) a3a. (h) KDDCUP08.

TABLE XII

VALUES OF h OF THE WILCOXON SIGNED RANK TEST AT THE 5% SIGNIFICANCE LEVEL BETWEEN THE TEST ACCURACIES OF FVS2-OSVM AND THE OTHER COMPARING ALGORITHMS. THE CASE $h = 0$ MEANS THAT THE PRIMAL HYPOTHESIS OF THE WILCOXON SIGNED RANK TEST IS ACCEPTED, THAT IS, THE DIFFERENCE BETWEEN THE TEST ACCURACY OF FVS2-OSVM AND THAT OF THE OTHER COMPARING ALGORITHMS IS NOT SIGNIFICANCE, AND THE CASE $h = 0$ MEANS THAT THE PRIMAL HYPOTHESIS OF WILCOXON SIGNED RANK TEST IS REJECTED. A1: FVS1-OSVM. A2: VS-OSVM. A3: RS-OSVM. A4: LASVM. A5: LIBSVM

Data Sets	A1	A2	A3	A4	A5
Ionosphere	0	0	1	1	1
Circle2	1	0	1	0	1
Satellite	0	1	1	1	1
Letter	0	0	1	0	0
Mushroom	0	0	1	0	0
Shuttle	0	0	1	1	0
a3a	1	1	1	1	0
KDDCUP08	1	1	1	1	1

Additionally, we analyze the accuracy with a Wilcoxon signed rank test at the 5% significance level. Note that all the experiments are carried out ten times independently. Let x_i be the test accuracy obtained by FVS2-OSVM and let y_i be the test accuracy obtained by one of the other compared algorithms (FVS1-OSVM, VS-OSVM, RS-OSVM, LASVM, and LIBSVM), where $1 \leq i \leq 10$. Then, the difference between $X = [x_1, x_2, \ldots, x_{10}]$ and $Y = [y_1, y_2, \ldots, y_{10}]$ is analyzed by using the Wilcoxon signed rank test. The values of h of the Wilcoxon signed rank test are presented in Table XII, where the case $h = 0$ means that the primal hypothesis of the Wilcoxon signed rank test is accepted, that is, the difference between X and Y is not significance, and the case $h = 1$ means that the primal hypothesis of the Wilcoxon signed rank test is rejected. From Tables XI and XII, it can be observed that FVS2-OSVM is better than RS-OSVM with respect to the test accuracy since: 1) the average of the test accuracy of FVS2-OSVM is higher than that of RS-OSVM, 2) in terms of the test accuracy, the difference between FVS2-OSVM and RS-OSVM is significance based on the Wilcoxon signed rank test; and 3) in terms of the standard deviation of the test accuracy, FVS2-OSVM is more stable than RS-OSVM.

In conclusion, FVS2-OSVM is the most efficient algorithm for online classification among the six comparing methods. In addition, in FVS2-OSVM, the outliers are deleted from the training samples, so FVS2-OSVM is more stable than the other comparing methods.

V. CONCLUSION

In this paper, a novel fast algorithm of CHVS has been proposed for online classification, which is based on the convex hull decomposition and the property of projection. In the proposed algorithm, the distance between a point and a convex hull can be computed by solving a linear equation problem, which is much faster than solving a quadratic optimization problem as used previously in [30]. Therefore, our algorithm is much faster than the CHVS algorithm proposed in [30]. Furthermore, the outliers are deleted in the initial process by using the designed Deoutlier algorithm, so our algorithm is more stable compared with that in [30]. Moreover, our algorithm allows the convex hull to be just approximated, instead of computed exactly, by setting an appropriate termination condition, which plays an important role in deleting more nonimportant samples for high-dimensional data. We have also given a rigorous analysis on the upper bound between the SVMs based on the convex hull vertices selected and all the training samples. Experimental results have been carried out on synthetic and real data sets and demonstrated that the proposed algorithm can solve the online classification problems

efficiently, since the SSP of our FVS2-OSVM algorithm is almost twice faster than that of VS-OSVM proposed in [30] with the test accuracy being higher than that of VS-OSVM.

ACKNOWLEDGMENT

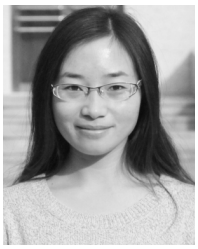
The authors would like to thank the anonymous reviewers for their valuable comments and constructive suggestions.

REFERENCES

[1] S. Agarwal, V. V. Saradhi, and H. Karnick, "Kernel-based online machine learning and support vector reduction," *Neurocomputing*, vol. 71, pp. 1230–1237, Sep. 2008.
[2] K. Bache and M. Lichman, "UCI machine learning repository," Dept. Inf. Comput. Sci., Univ. California, Irvine, CA, USA, Tech. Rep., 2013. [Online]. Available: http://archive.ics.uci.edu/ml
[3] 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.
[4] 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, Oct. 2005.
[5] C. J. C. Burges, "Simplified support vector decision rules," in *Proc. 13th Int. Conf. Mach. Learn.*, 1996, pp. 71–77.
[6] E. Carrizosa, B. Martin-Barragan, and D. Romero Morales, "Binarized support vector machines," *INFORMS J. Comput.*, vol. 22, no. 1, pp. 154–167, 2010.
[7] C. C. Chang and C. J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–27, 2011.
[8] K. De Brabanter, J. De Brabanter, J. A. K. Suykens, and B. De Moor, "Optimized fixed-size least squares support vector machines for large data sets," *Comput. Statist. Data Anal.*, vol. 54, no. 6, pp. 1484–1504, Jun. 2010.
[9] S. Ertekin, L. Bottou, and C. L. Giles, "Nonconvex online support vector machines," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 2, pp. 368–381, Feb. 2011.
[10] Y. Freund and R. E. Schapire, "Large margin classification using the perceptron algorithm," *Mach. Learn.*, vol. 37, no. 3, pp. 277–296, 1999.
[11] G. Fung and O. L. Mangasarian, "A feature selection Newton method for support vector machine classification," *Comput. Optim. Appl.*, vol. 28, no. 2, pp. 185–202, 2004.
[12] 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.
[13] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. New York, NY, USA: Springer, 2001.
[14] S. Jayadeva, "Learning a hyperplane regressor by minimizing an exact bound on the VC dimension," *Neurocomputing*, vol. 149, pp. 683–689, Oct. 2015.
[15] 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.
[16] Y. J. Lee and O. L. Mangasarian, "RSVM: Reduced support vector machines," in *Proc. 1st SIAM Int. Conf. Data Mining*, 2001, pp. 325–361.
[17] N. Mezghani, A. Mitiche, and M. Cheriet, "Bayes classification of online arabic characters by Gibbs modeling of class conditional densities," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 7, pp. 1121–1131, Jul. 2008.
[18] D. Nguyen and T. Ho, "An efficient method for simplifying support vector machines," in *Proc. 22nd Int. Conf. Mach. Learn.*, 2005, pp. 617–624.
[19] G. Pillonetto, F. Dinuzzo, and G. D. Nicolao, "Bayesian online multitask learning of Gaussian processes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 2, pp. 193–205, Feb. 2010.
[20] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychol. Rev.*, vol. 65, pp. 386–407, Sep. 1958.
[21] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, "Pegasos: Primal estimated subgradient solver for SVM," *Math. Program.*, vol. 127, pp. 3–30, Mar. 2011.
[22] S. Shalev-Shwartz, "Online learning: Theory, algorithms, and applications," Ph.D. dissertation, School Comput. Sci. Eng., Hebrew Univ. Jerusalem, Jerusalem, Israel, 2007. [Online]. Available: http://www.cs.huji.ac.il/~shais
[23] B. Schölkopf, A. J. Smola, *Learning With Kernels*. Cambridge, MA, USA: MIT Press, 2002.
[24] N. Slonim, E. Yom-Tov, and K. Crammer, "Active online classification via information maximization," in *Proc. 22nd Int. Joint Conf. Artif. Intell.*, p. 1489, 2011.
[25] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle, *Least Squares Support Vector Machines*, Singapore: World Scientific, 2002.
[26] D. Tax and P. Laskov, "Online SVM learning: From classification to data description and back," in *Proc. 13th IEEE Workshop Neural Netw. Signal Process.*, 2003, pp. 499–508.
[27] A. V. Uzilov, J. M. Keegan, and D. H. Mathews, "Detection of noncoding RNAs on the basis of predicted secondary structure formation free energy change," *BMC Bioinform.*, vol. 7, p. 173, Dec. 2006.
[28] V. Vapnik, *The Nature of Statistical Learning Theory*. New York, NY, USA: Springer, 1995.
[29] D. Wang, B. Zhang, P. Zhang, and H. Qiao, "An online core vector machine with adaptive MEB adjustment," *Pattern Recognit.*, vol. 43, pp. 3469–3482, Jun. 2010.
[30] D. Wang, H. Qiao, B. Zhang, and M. Wang, "Online support vector machine based on convex hull vertices selection," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 4, pp. 593–609, Apr. 2013.
[31] J. Wang, P. Zhao, and S. C. H. Hoi, "Cost-sensitive online classification," *IEEE Trans. Knowl. Data Eng*, vol. 26, no. 10, pp. 2425–2438, Sep. 2014.
[32] R. C. Wilson, M. R. Nassar, and J. I. Gold, "Bayesian online learning of the hazard rate in change-point problems," *Neural. Comput.*, vol. 22, pp. 2452–2476, Jan. 2010.
[33] M. Wu, B. Scholkopf, and G. Bakir, "Building sparse large margin classifiers," in *Proc. 22nd Int. Conf. Mach. Learn.*, 2005, pp. 996–1003.
[34] (2008). *KDDCup 2008*. [Online]. Available: http://www.kdd.org/kdd-cup/view/kdd-cup-2008/Data

Shuguang Ding received the B.Sc. degree in mathematics from Central South University, Changsha, China, in 2012. He is currently pursuing the Ph.D. degree in computational mathematics with the Institute of Applied Mathematics, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China.

His current research interests include online kernel learning, online semi-supervised learning and online multiview learning.



Xiangli Nie (M'15) received the B.Sc. degree in mathematics from Shandong University, Jinan, China, and the Ph.D. degree in computational mathematics from the Institute of Applied Mathematics, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China, in 2010 and 2015, respectively.

She is currently an Assistant Professor with the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences. Her current research interests include (polarimetric) synthetic aperture radar image understanding and online classification.

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 with the Department of Manufacturing Engineering and Engineering Management, City University of Hong Kong, Hong Kong, as a Research Assistant Professor from 1997 to 2000 and an Assistant Professor from 2000 to 2002. Since 2002, she has been a Lecturer with the School of Informatics, The 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, which has successfully been applied by herself in robot assembly, robot grasping, and part recognition. The work has been 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 currently a member of the Administrative Committee of the IEEE Robotics and Automation Society (RAS), the IEEE Medal for Environmental and Safety Technologies Committee, the Early Career Award Nomination Committee, the Most Active Technical Committee Award Nomination Committee, and the Industrial Activities Board for RAS. She is currently an Associate Editor of the IEEE TRANSACTION ON CYBERNETICS and IEEE TRANSACTION ON AUTOMATION SCIENCE AND ENGINEERING, and an Editor-in-Chief of *Assembly Automation*.

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, Glasgow, U.K., in 1983, 1985, and 1992, respectively.

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 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 promoted to Reader in applied mathematics in 2000 and to Professor of Applied Mathematics in 2003. He is currently a Professor with the Institute of Applied Mathematics, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China. His current research interests include direct and inverse scattering problems, radar and sonar imaging, machine learning, and pattern recognition.

Dr. Zhang is currently an Associate Editor of the IEEE TRANSACTION ON CYBERNETICS and *Applicable Analysis*.