Pattern Recognition 43 (2010) 3468–3482

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

An online core vector machine with adaptive MEB adjustment

Di Wang[a], Bo Zhang[a,*], Peng Zhang[a], Hong Qiao[b]

[a] LSEC and Institute of Applied Mathematics, AMSS, Chinese Academy of Sciences, Beijing 100190, China
[b] Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China

ARTICLE INFO

Article history:
Received 28 September 2009
Received in revised form
18 April 2010
Accepted 15 May 2010

Keywords:
Minimum enclosing ball
Online classifier
Core vector machine
Support vector machine
Machine learning

ABSTRACT

Support vector machine (SVM) is a widely used classification technique. However, it is difficult to use SVMs to deal with very large data sets efficiently. Although decomposed SVMs (DSVMs) and core vector machines (CVMs) have been proposed to overcome this difficulty, they cannot be applied to online classification (or classification with learning ability) because, when new coming samples are misclassified, the classifier has to be adjusted based on the new coming misclassified samples and all the training samples. The purpose of this paper is to address this issue by proposing an online CVM classifier with adaptive minimum-enclosing-ball (MEB) adjustment, called online CVMs (OCVMs). The OCVM algorithm has two features: (1) many training samples are permanently deleted during the training process, which would not influence the final trained classifier; (2) with a limited number of selected samples obtained in the training step, the adjustment of the classifier can be made online based on new coming misclassified samples. Experiments on both synthetic and real-world data have shown the validity and effectiveness of the OCVM algorithm.

1. Introduction

So far, SVMs have been widely used in many real-world applications due to their good performance. These applications include face recognition [1,2], gene expression data clustering [3], pedestrian detection [4], handwriting recognition [5], as well as the classification tasks of text [6,7], fingerprint [8] and texture [9]. The main advantages of the SVMs can be summarized as follows [10]:

- a compromise between minimizing empirical risk and preventing overfitting is taken by implementing the structural risk minimization;
- the process of computing the classification hyperplane involves a convex quadratic optimization problem which can be solved efficiently and has a global solution;
- the obtained classifier is completely determined by the support vectors and the type of kernel functions which are used for training.

Despite the above advantages, SVMs have also the following two main disadvantages which limit their application in real-time pattern recognition problems:

- the convex quadratic optimization problem arising in SVMs is a large-scale problem for very large data sets, so it is difficult for SVMs to deal with very large data effectively;
- SVMs handle training samples in a batch mode; when a new training sample arrives, the whole training process has to be implemented once again to adjust the classifier; thus, it is not practical for SVMs to be used for online learning.

Recently, many algorithms have been proposed to address the fast computation issue of large-scale SVMs (see [18,19] for a good literature survey on this). Among these algorithms are the decomposed SVMs (DSVMs) [11–15] and core vector machines (CVMs) [16,17]. Their main ideas can be briefly summarized as follows.

- DSVMs essentially repeats two operations until some optimality condition is satisfied: one is to select a working set and the other is to minimize the original objective function of the quadratic programming (QP) problem arising in SVMs by updating the variables only associated with the working set. The key step in DSVMs is how to select a suitable working set at each iteration.
- CVMs reformulate SVMs as the minimum enclosing ball (MEB) problems in computational geometry. Then an approximate optimal solution to the original optimization problem can be obtained by utilizing efficient approximate MEB approaches. Reported experimental results on very large data sets have shown that classification results obtained by CVMs are as

[*] Corresponding author. Tel.: +86 10 6265 1358.
E-mail addresses: wangdi@amss.ac.cn (D. Wang), b.zhang@amt.ac.cn (B. Zhang), zhangpeng@amss.ac.cn (P. Zhang), hong.qiao@ia.ac.cn (H. Qiao).

0031-3203/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2010.05.020

accurate as those obtained by SVMs, while the computation speed of the former is much faster than that of the latter since the computational complexity of the MEB problem is independent of the dimension and number of data samples.

DSVMs and CVMs have been successfully applied to solve many large-scale classification problems. However, the online learning issue of the DSVM and CVM classifiers is still not addressed. In these two algorithms, data are processed in a batch mode. When a new training sample arrives, the whole training process should be implemented once again to adjust the classifier. Thus, online adjustment of the classifier is impossible.

Online learning ability of a classifier is very important in real-time pattern recognition systems such as pedestrian detection system and aircraft visual navigation system. In such systems, data are input in a consecutive sequence. The classifier needs to be adjusted online with misclassified samples to achieve more accurate classification results. Recently, several successful approaches have been proposed to address the online learning issue of SVMs [21–33] which will be reviewed in the next section. However, very few work is concerned with deleting training samples effectively without influencing the final trained classifier. Efficient samples deletion is very important for online classification. For very large data sets, a very large amount of training samples will be used in order to get a good trained classifier, so if no samples are deleted then online adjustment of the classifier would be very difficult since all training samples will be needed to re-train the classifier.

In this paper, we propose an online CVM classification algorithm with adaptive MEB adjustment, based on an efficient redundant samples deletion technique. An advantage of our approach over the existing ones is that our online CVM algorithm, called OCVM, can be applied to deal with very large data sets efficiently, as shown by the experimental results on both synthetic and real-world data sets. Our OCVM consists of the following two steps:

- Off-line samples deletion: an upper bound is given of the distance between the center of the approximate MEB at each iteration and the accurate MEB of all the training samples and then used to identify training samples which definitely lie in the final computed MEB; such data samples are permanently deleted from the set of training samples to accelerate the speed of MEB computation.
- Online classifier adjustment: the selected training samples after the off-line samples deletion step, together with new coming misclassified samples, are used to compute the new classifier coefficients; then online updating of the classifier can be achieved since only very limited training samples are maintained in this process due to the efficient samples deletion.

Experimental results on both synthetic and real-world data sets have been presented to illustrate the validity and effectiveness of the proposed method.

The rest of the paper is organized as follows. A literature review on the existing online learning algorithms for SVMs is presented in Section 2. In Section 3, the classical CVM algorithm is briefly reviewed. The redundant samples deletion algorithm is described in Section 4, and the new OCVM algorithm is presented in Section 5. In Section 6, experiments are conducted on both synthetic and real-world data to illustrate the validity and effectiveness of the proposed method. Some concluding remarks are given in Section 7.

2. Literature review on existing online SVM algorithms

In this section, we briefly review the recent progress on the online learning issue of SVMs. Cheng and Shih [20] proposed an incremental training algorithm of SVMs by using active query. A subset of training samples is first selected by the K-Means clustering to compute the initial separating hyperplane. At each iteration, the classifier is updated based on the training samples selected by active query. The iteration stops until there are no unused informative training samples. Syed et al. [21] proposed a so-called support vector incremental algorithm where the SVM classifier is re-trained based on both the new samples and the support vectors from the trained SVM. Then the SVM classifier can be incrementally updated. Peng et al. [22] designed a questing process in which the machine quests the interrogative instance by the distance between the point and the hyperplane. Through this process, some samples are selected from the incremental training set, which are used, together with the support vectors from previous steps, to update the classifier. Bordes et al. [23] proposed a fast algorithm, called LaRank, for efficiently solving multi-class SVMs. Different from the traditional methods which rely on the full gradient, the LaRank algorithm computes and updates the gradient in a random pattern. Thus, the computational complexity can be greatly reduced, and online learning can be achieved.

In recent years, the stochastic gradient descent (SGD) method has been introduced to online learning of SVMs [24–27]. To solve the primal problem of SVMs, gradient-based methods compute the gradient using all the training samples, which is very expensive for very large data sets. Alternatively, SGD-based methods compute the gradient with respect to only one randomly selected sample. Therefore, these methods run significantly faster than the gradient descent methods if the number of training samples is very large, so online adjustment of the SVM classifier with newly arrived samples can be achieved. Although both the SGD-based methods and our proposed OCVM algorithm only utilize part of the training samples in online learning, they are quite different in motivation and algorithm design. Our OCVM algorithm aims to delete most redundant samples safely and permanently so that the adjustment of the classifier only involves a limited number of training samples. Moreover, our OCVM solves the dual problem using the sequential minimal optimization (SMO) method.

On the other hand, a series of online learning algorithms for SVMs have been proposed, which are mainly based on analyzing the change of the Karush–Kuhn–Tucker (KKT) conditions while updating the classifier. Note that the KKT conditions are the optimality criteria for the solution of the SVM quadratic programming problems. Cauwenberghs and Poggio [28] analyzed the change of the KKT conditions when one data point is added in or removed from the training set. Then a so-called bookkeeping step is used to compute the new coefficients of the classifier to achieve online updating. Agarwal et al. [29] introduced the concept of S-span and constructed a mechanism to remove/add support vectors. Through this mechanism, a new sample is first determined whether to be a support vector or not. If the new sample is a support vector, then a judgement has to be made to see if the support vector should be removed. The coefficients of the classifier are then adjusted by the method of [28]. Davy et al. [30] adopted a similar idea as in [28] to compute the change of the support vector decision function and designed an online novelty detection algorithm as well as an online abnormality detection algorithm for online updating of the classifier. Based on the traditional least square SVM model, Awad et al. [31] modified the SVM for multi-class classification. They utilized the change of the KKT conditions, in the case when a new sample is added, to compute the perturbation of the classifier coefficients. Then online updating of the classifier is achieved. Lau and Wu [32] presented an online support vector classifier (OSVC) for pattern classification problems where data are input in sequence rather than in batch mode. New coming data, together with the data points violating the KKT conditions and the support vectors from the last iteration, are used to form the new training data set to train the classifier at the current step. The iteration stops only if no data point violates the KKT conditions.

After submission of this paper, we noticed, on pointing out by an anonymous referee, that an online learning algorithm, called online independent SVMs (OISVMs), was proposed in [33], based on the idea of selecting a limited number of samples to achieve online learning. In the OISVM algorithm, a hypothesis is constructed via a subset of the samples seen so far called *basis*, and new samples are added into the basis only if they are linearly independent in the feature space of the current basis. Since the number of samples in the basis is far less than the number of the original samples, the time complexity for learning the classifier can be greatly reduced so that online learning of SVMs can be achieved. It was proved both theoretically and empirically in [33] that the size of the basis does not grow linearly with the training set, but converges to a limit size and then stops growing. This is different from the proposed OCVM algorithm, which is based on the safe deletion of redundant samples other than the selection of a limited number of samples.

3. Brief review of core vector machines

In this section, the CVM algorithm [16] and the generalized CVM (GCVM) algorithm [17] will be reviewed, respectively. Both of these two algorithms utilize the relationship between MEB and SVM to solve the QP problem arising in SVM with the approximate MEB method. In this section, we first review the approximate MEB method and then introduce the CVM and GCVM algorithms. We also depict the kernel-related problems which CVM and GCVM can be applied to.

3.1. The approximate minimum enclosing ball method

For a given set of data samples, $P=\{x_i\}_{i=1}^m$, $x_i \in \mathbb{R}^d$, the minimum enclosing ball of P, denoted by MEB(P), is the smallest ball which contains all the data samples in P. Let k be a kernel function, let φ be the associated feature map and let $\{\varphi(x_i)\}_{i=1}^m$ be the set of $\varphi-$mapped points in the kernel space induced by k. Denote by $B(c, R)$ the MEB of $\{\varphi(x_i)\}_{i=1}^m$, where c is its center and R is its radius. Traditional methods for computing the exact MEB of a data set are not efficient for large d (for example, $d>30$). A possible solution is to use the $(1+\varepsilon)-$approximation MEB algorithm. For a given small value $\varepsilon>0$, a subset $S \subset P$ is called a $\varepsilon-$core set if $P \subset B(c,(1+\varepsilon)r_{\mathrm{MEB}(S)})$, where $r_{\mathrm{MEB}(S)}$ denotes the radius of MEB(S). The ball $B(c,(1+\varepsilon)r_{\mathrm{MEB}(S)})$ is an $(1+\varepsilon)-$approximation of MEB(S).

An important progress has been made by Bădoiu and Clarkson who introduced an efficient iterative algorithm in [34] to find the $(1+\varepsilon)-$approximation MEB of a data set. The key idea is to expand the ball $B(c_t, r_t)$ at the current iteration by including the furthest point which is outside the ball $B(c_t,(1+\varepsilon)r_t)$. The iteration will stop if $B(c_t,(1+\varepsilon)r_t)$ covers all data points in S. The most important fact is that, the size of the $\varepsilon-$core set computed through this algorithm depends only on ε and is independent of d and m. Therefore, the algorithm proposed in [34] can be applied to high-dimensional data sets.

3.2. The CVM algorithm

In [16], it is shown that the exact MEB problem is equivalent to a hard-margin support vector data description (SVDD) [35]. It can also be applied to soft-margin one-class and two-class SVMs if

$$k(x,x) \equiv \kappa, \tag{1}$$

where κ is a constant. In the following subsections, the hard-margin SVDD, one-class SVMs and two-class SVMs will be reviewed, respectively.

3.2.1. Hard-margin SVDD

Formally, the hard-margin SVDD can be formulated as

$$\begin{aligned} \min_{c,R} \quad & R^2 \\ \text{s.t.} \quad & \|-\varphi(x_i)\|^2 \le R^2, \quad i=1,\ldots,m. \end{aligned} \tag{2}$$

Its dual problem is equivalent to a simpler QP problem

$$\begin{aligned} \max_{\alpha} \quad & -\alpha^T K \alpha \\ \text{s.t.} \quad & \alpha^T e = 1, \quad \alpha \ge 0, \end{aligned} \tag{3}$$

if the kernel satisfies (1). Here, e is a column vector of all ones. Then any QP problem of the form (3) can be regarded as an MEB problem if the kernel k satisfies (1).

3.2.2. One-class SVMs

The primal one-class L2-SVM can be described as

$$\begin{aligned} \min_{w,\rho,\xi_i} \quad & \|w\|^2 - 2\rho + C\sum_{i=1}^m \xi_i^2 \\ \text{s.t.} \quad & w^T\varphi(x_i) \ge \rho - \xi_i, \quad i=1,\ldots,m. \end{aligned} \tag{4}$$

Its dual problem is

$$\begin{aligned} \max_{\alpha} \quad & -\alpha^T \tilde{K} \alpha \\ \text{s.t.} \quad & \alpha^T e = 1, \quad \alpha \ge 0, \end{aligned} \tag{5}$$

where $\tilde{K} = [\tilde{k}_{ij}]$ with $\tilde{k}_{ij} = k(x_i,x_j) + \delta_{ij}/C$ and $\delta_{ij}=1$ if $i{=}j$ and 0 otherwise. It is obvious that $\tilde{k}(x,x) = \kappa + 1/C$ is a constant. Then the one-class L2-SVM can also be regarded as an MEB problem.

3.2.3. Two-class SVMs

The primal two-class L2-SVM can be written as

$$\begin{aligned} \min_{w,b,\rho,\xi_i} \quad & \|w\|^2 + b^2 - 2\rho + C\sum_{i=1}^m \xi_i^2 \\ \text{s.t.} \quad & y_i(w^T\varphi(x_i) + b) \ge \rho - \xi_i, \quad i=1,\ldots,m. \end{aligned} \tag{6}$$

Its dual problem is

$$\begin{aligned} \max_{\alpha} \quad & -\alpha^T \tilde{K} \alpha \\ \text{s.t.} \quad & \alpha^T e = 1, \quad \alpha \ge 0, \end{aligned} \tag{7}$$

where $\tilde{K} = [\tilde{k}_{ij}]$ with $\tilde{k}_{ij} = y_i y_j k(x_i,x_j) + y_i y_j + \delta_{ij}/C$. Then the modified kernel $\tilde{k}$ satisfies that $\tilde{k}(x,x) = \kappa + 1 + 1/C$ which is also a constant if k satisfies (1). Therefore, the two-class L2-SVM can be regarded as an MEB problem.

3.2.4. The CVM algorithm

Given $\varepsilon > 0$, the CVM algorithm is summarized in Algorithm 1, where S_t, c_t and r_t denote the core set, the center and the radius of the approximate MEB computed at the t-th iteration, respectively.

Algorithm 1. CVM algorithm.

```
   Input: Set of input samples $P \subset R^d$, parameter $\varepsilon \in (0,1)$.
1  Randomly select $x_0 \in P$. Let $x' = \mathrm{argmax}_{x \in P}\|\varphi(x_0)-\varphi(x)\|$, and
   $x'' = \mathrm{argmax}_{x \in P}\|\varphi(x')-\varphi(x)\|$.
2  $t=0$, $S_t = \{x',x''\}$.
3  Let $B(c_t,r_t)$ denote the MEB($S_t$) obtained via (3).
4  if $P \subset B(c_t,(1+\varepsilon)r_t)$ then
5    |Return $B(c_t,(1+\varepsilon)r_t),S_t$
6  else
7    |$x^* = \mathrm{argmax}_{x \in P}\|c_t-\varphi(x)\|$
8  end
9  $S_{t+1} \leftarrow S_t \cup \{x^*\}$, $t=t+1$. Go to Step 3.
```

3.3. The GCVM algorithm

Although CVM has achieved its success on very large data sets [16], there are limitations when applying CVM to other kernel methods. The reason mainly lies in two aspects:

- $k(x,x) \equiv \kappa$ which is a constant,
- the QP problem is of the form in (3).

Tsang et al. [17] proposed the generalized CVM algorithm (GCVM) to resolve these two problems. The details can be found in [17], which will not be reviewed here due to lack of space.

4. The redundant samples deletion algorithm

In the CVM algorithm, it is necessary to compute the distance $\|c_t-\varphi(x)\|$ for each $x \in P$ in order to find the furthest point from c_t, where $c = \sum_{x_i \in S_t} \alpha_i \varphi(x_i)$. This can be done by noting that

$$\|c_t-\varphi(x)\|^2 = \left\|\sum_{x_i \in S_t} \alpha_i \varphi(x_i)-\varphi(x)\right\|^2 = \sum_{x_i,x_j \in S_t} \alpha_i \alpha_j k(x_i,x_j) -2\sum_{x_i \in S_t} \alpha_i k(x_i,x)+k(x,x). \tag{8}$$

In the GCVM algorithm, it is also needed to calculate the distance

$$\left\| \begin{bmatrix} c_t \\ 0 \end{bmatrix} - \begin{bmatrix} \varphi(x) \\ \delta \end{bmatrix} \right\|$$

for each $x \in P$ in order to find the furthest point from $[{c_t \atop 0}]$. It can be seen that

$$\left\| \begin{bmatrix} c_t \\ 0 \end{bmatrix} - \begin{bmatrix} \varphi(x) \\ \delta \end{bmatrix} \right\|^2 = \left\|\sum_{x_i \in S_t} \alpha_i \varphi(x_i)-\varphi(x)\right\|^2 + \delta^2 = \sum_{x_i,x_j \in S_t} \alpha_i \alpha_j k(x_i,x_j) -2\sum_{x_i \in S_t} \alpha_i k(x_i,x)+k(x,x)+\delta^2. \tag{9}$$

From (8) and (9), it can be seen that it needs $O(|S_t|^2+m|S_t|)$ operations to compute the distances between the current center and all the m samples in the set P at the t-th iteration, where $|S_t|$ denotes the cardinality of the set S_t. This complexity is very high when m is large.

However, most of the samples in the set P are redundant in the process of computing the approximate MEB of P. If these redundant samples can be deleted, a very large amount of computational time can be saved in the MEB computation. Based on this idea, a redundant samples deletion algorithm is proposed in the following subsection.

4.1. Redundant samples deletion algorithm

The overall description of the proposed algorithm is given in Algorithm 2, where Ω is set manually and decides whether or not to start the deletion of data samples. When $\Omega = 1$, the algorithm never deletes data samples. When $\Omega = 0$, the algorithm detects data samples which need to be deleted. Therefore, Algorithm 2 is equivalent to Algorithm 1 when $\Omega = 1$. L determines how many data samples will finally be kept. The larger L is, the less data samples are kept. Step 9 identifies samples in the ball $B(c_t,r_{d_t})$ which can be deleted at the t-th iteration. The deleted samples will not affect the final solution of the $(1+\varepsilon)-$approximation MEB of the set P, which will be proved in the next subsection. It is clear that the time complexity of the RSD algorithm is $O(|S_t|^2+|P_r||S_t|)$, where $|S_t|$ denotes the cardinality of the set S_t and $|P_r|$ is the cardinality of the preserved samples set P_r.

Algorithm 2. Redundant samples deletion (RSD) algorithm.

```
    Input: Input set $P \subset \mathbb{R}^d$, parameter $\varepsilon \in (0,1)$, $\Omega \in \{0,1\}$ and real number $L > 1$.
1   Randomly select $x_0 \in P$ and set $x' = \mathrm{argmax}_{x \in P}\|\varphi(x_0)-\varphi(x)\|$, $x'' = \mathrm{argmax}_{x \in P}\|\varphi(x')-\varphi(x)\|$.
2   $t=0$, $S_t = \{x',x''\}$, $flag\ 1=0$, $N$=size($P$) and $P_r$=$P$.
3   Let $B(c_t,\ r_t)$ denote the MEB($S_t$) obtained from (3).
4   if $P_r \subset B(c_t,(1+\varepsilon)r_t)$ then
5     |Return $B(c_t,(1+\varepsilon)r_t),S_t$
6   else
7
8     |$x^* = \mathrm{argmax}_{x \in P_r}\|c_t-\varphi(x)\|,D_{t_1} = \|c_t-\varphi(x^*)\| flag = \max\{(r_t-\sqrt{D_{t_1}^2-r_t^2})/r_t, flag1\}$.
9     |if $flag \geq \Omega$ and $size(P_r) \geq N/L$ then
      ||$P_d = \{x|x \in P_r, \|c_t-\varphi(x)\| < r_d\}$, where $r_d$ is given in Theorem 1,$P_r = P_r \backslash P_d$ and $flag1 \leftarrow 1$.
10    |end
11  end
12  $S_{t+1} \leftarrow S_t \cup \{x^*\}$, $t=\ t+1$. Go to Step 3.
```

4.2. Theoretical analysis of the proposed algorithm

In this subsection, we will give a detailed theoretical analysis of the proposed algorithm. To do this, denote by $B(c, R)$ and $B(c_t, r_t)$ the MEBs of the set P and the core-set S_t, respectively. Our main results in this subsection are presented in Theorems 1 and 2. These results can be proved through several lemmas which are given and proved in the appendix.

Theorem 1. *Let*

$$d=\min\left\{\sqrt{D_{t_1}^2-\frac{(D_{t_1}^2+r_t^2)^2}{4D_{t_1}^2}},\sqrt{\frac{(D_{t_1}+D_{t_2})^2}{4}-r_t^2}\right\}$$

and let $r_d=(D_{t_1}^2+r_t^2)/2D_{t_1}-d$. Then all points in the ball $B(c_t, r_d)$ can be deleted at the t-th iteration without affecting the MEB of the set P.

Proof. From Lemmas 3 and 4, it can be seen that $\|c-c_t\| \le d$. Then, for any $p \in B(c_t,r_d)$,

$$\|p-c\| \le \|p-c_t\|+\|c-c_t\| \le r_d+d=(D_{t_1}^2+r_t^2)/2D_{t_1}-d+d \le R.$$

It thus follows that any point in the ball $B(c_t, r_d)$ is also in the ball $B(c, R)$. Therefore, we can delete all points in the ball $B(c_t, r_d)$. □

Theorem 2. *If the RSD algorithm stops at the t-th iteration, then $d \le \sqrt{2\varepsilon+\varepsilon^2}R$.*

Proof. First, it is clear that $(D_{t_1}^2+r_t^2)^2/4D_{t_1}^2 \ge r_t^2$. Then, by Theorem 1 and (12) in the appendix, we have

$$d \le \sqrt{D_{t_1}^2-(D_{t_1}^2+r_t^2)^2/(4D_{t_1}^2)} \le \sqrt{D_{t_1}^2-r_t^2}$$
$$\le \sqrt{(1+\varepsilon)^2r_t^2-r_t^2} \le \sqrt{2\varepsilon+\varepsilon^2}R. \qquad \square$$

Theorem 2, together with Theorem 1 and the definition of r_d in Theorem 1, means that, when the algorithm stops at the t-th iteration, the ball $B(c_t, r_d)$ is very close to the final approximate MEB $B(c_t, r_t)$. This implies that most of the points in the final approximate MEB $B(c_t, r_t)$ are redundant and therefore can be deleted.

5. Online CVM with adaptive MEB adjustment

Based on the redundant samples deletion (RSD) algorithm proposed in Section 4, online updating of the CVM classifier can be achieved. In this section, we first discuss how MEB can be learned online based on the selected samples after samples deletion, that is, those samples which are preserved after deletion, and then show how the online approximate MEB algorithm can be implemented in CVM to achieve online adjustment of the classifier.

5.1. Online approximate MEB algorithm

The diagram of the online approximate MEB algorithm is shown in Fig. 1. The whole algorithm can be decomposed into two steps stated below.

Step 1: Compute the current $(1+\varepsilon)-$approximation of MEB of the existing data samples. Through Algorithm 2, the approximate MEB, $B(c,(1+\varepsilon)R)$, the current core set S and the set P_r of the preserved data samples are computed, which are stored for the following online adjustment step.

Step 2: Expand the current approximate MEB based on the new coming samples: when a new sample x_{new} arrives, we use (8) with $x=x_{\text{new}}$, $c_t=c$, $S_t=S$ to determine whether it lies within the $(1+\varepsilon)-$ball $B(c,(1+\varepsilon)R)$; if yes then the current approximate MEB will not be expanded, and if no then the current approximate MEB will be expanded with the following iteration process:

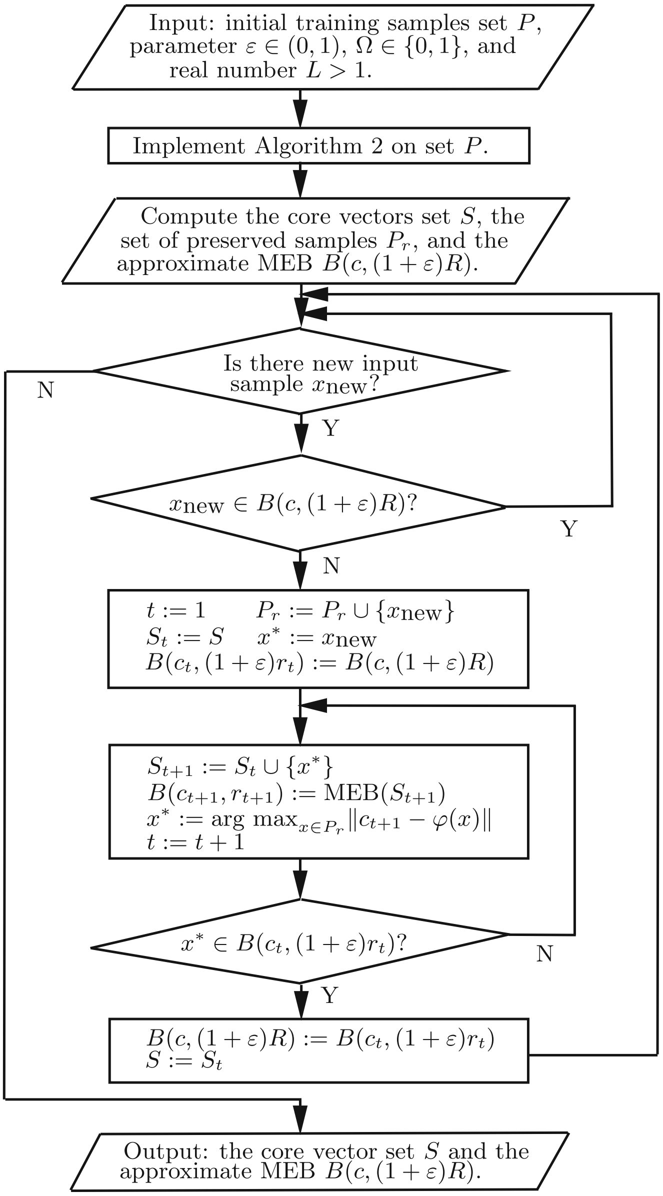
Input: initial training samples set P, parameter $\varepsilon \in (0,1)$, $\Omega \in \{0,1\}$, and real number $L>1$.
Implement Algorithm 2 on set P.
Compute the core vectors set S, the set of preserved samples P_r, and the approximate MEB $B(c,(1+\varepsilon)R)$.
Is there new input sample $x$new?
N
Y
$x$new $\in B(c,(1+\varepsilon)R)$?
Y
N
$t:=1$ $P_r := P_r \cup \{x\text{new}\}$
$S_t := S$ $x^* := x$new
$B(c_t,(1+\varepsilon)r_t) := B(c,(1+\varepsilon)R)$
$S_{t+1} := S_t \cup \{x^*\}$
$B(c_{t+1}, r_{t+1}) := \text{MEB}(S_{t+1})$
$x^* := \arg\max_{x\in P_r} \|c_{t+1}-\varphi(x)\|$
$t := t+1$
$x^* \in B(c_t,(1+\varepsilon)r_t)$?
N
Y
$B(c,(1+\varepsilon)R) := B(c_t,(1+\varepsilon)r_t)$
$S := S_t$
Output: the core vector set S and the approximate MEB $B(c,(1+\varepsilon)R)$.

Fig. 1. Diagram of the online CVM algorithm.

- initialize $t=1$, $x^*=x_{\text{new}}$, $P_r=P_r \bigcup \{x^*\}$, $B(c_t,(1+\varepsilon)r_t)=B(c,(1+\varepsilon)R)$ and the core set $S_1=S$;
- at the t-th iteration, expand the current $B(c_t, r_t)$ incrementally by including the furthest sample x_* which lies in P_r but outside $B(c_t,(1+\varepsilon)r_t)$, that is, compute $B(c_{t+1}, r_{t+1})=\text{MEB}(S_{t+1})$ with $S_{t+1}=S_t \bigcup \{x^*\}$; set $t=t+1$;
- the iteration is repeated until $B(c_t,(1+\varepsilon)r_t)$ covers all points in P_r.

Finally, set $B(c,(1+\varepsilon)R)=B(c_t,(1+\varepsilon)r_t)$ and $S=S_t$. Repeat this step until no new samples are input.

5.2. Online CVM algorithm

As mentioned in Section 3, many kernel-related problems can be viewed as MEB problems or center-constrained MEB problems. Therefore, the key step of the online CVM is the online adjustment of MEB. In this section, we take the two-class L2-CVM as an example to depict the online updating process of the CVM classifier coefficients based on the online MEB algorithm. For other CVMs including the GCVMs the online updating process of the classifier coefficients is similar.

The procedure of the online CVM (OCVM) can be decomposed into three steps: (1) off-line training of the existing samples to obtain the initial classifier coefficients, (2) online adjustment of the MEB based on the new coming samples, and (3) online adjustment of the classifier coefficients. These three steps are explained in details as follows.

Step 1: From (6) and (7), the two-class classification problem is formulated as an MEB problem. Then by Algorithm 2, the initial approximate MEB, $B(C,(1+\varepsilon)R)$, the initial classifier coefficients and the current core set S are computed. The set P_r of the preserved samples is also stored for the online adjustment Step 2.

Step 2: When a new sample x arrives, $B(C,(1+\varepsilon)R)$ is adjusted online according to Step 2 of the online MEB algorithm depicted in Section 5.1. The kernel function k appeared in (8) in Step 2 of the online MEB algorithm is replaced with $\tilde{k}$ which is defined as (see Section 3.2.3):

$$\tilde{k}(x_i,x_j)=y_iy_jk(x_i,x_j)+y_iy_j+\delta_{ij}/C.$$

Step 3: The CVM classifier coefficients are adjusted based on the new approximate MEB, $B(c,(1+\varepsilon)R)$, as follows:

$$w=\sum_{x_i\in S}\alpha_i\varphi(x_i),\quad b=\sum_{x_i\in S}\alpha_iy_i,$$

where the coefficients α_i and the core set S are obtained by solving the QP problem (7) during the online adjustment of the MEB in Step 2.

Repeat Steps 2 and 3 until no new samples are input.

6. Experiments

In this section, we carry out experiments on both synthetic and real-world data sets to show the effectiveness of the proposed algorithms.

We first consider the one-class two Gaussian data to illustrate the online learning procedure of the OCVM algorithm, which is shown in Fig. 2.

As the training data, a set of 400 data samples are first generated from a mixture of two two-dimensional Gaussian distributions, which are marked with blue disks. Then the off-line CVM with the RSD algorithm is performed to obtain the one-class classification boundary. The preserved data samples and the support vectors are denoted by red squares without and with frames, respectively. The classification boundary is plotted with black line.

In the online updating procedure, 15 independent data samples from a different mixture of two two-dimensional Gaussian distributions are generated. They are designed to lie outside of the trained classification boundary. Then the CVM classifier is updated by inputting the testing data samples one by one. The updated classification boundary obtained by the OCVM algorithm is plotted by pink line. The same task is also tested with the standard CVM in a batch mode, that is, a new coming data sample and all the previous data samples are combined together to train a new boundary. The updated boundary with this procedure is plotted by black line. It can be seen that the two boundaries almost coincide, which shows that the updating results of the OCVM algorithm and the standard CVM algorithm are almost the same. However, the time consumed by OCVM is 0.11 s, while that consumed by the standard CVM is 0.66 s.

We now consider several two-class synthetic and real-world data sets to illustrate the online classification ability of the OCVM algorithm. The experimental results show that OCVM can meet the demand of online classification of data streams. Details of the experiments are stated below.

6.1. Overall description of experiment design and results

All experiments are performed on a Desktop PC with Intel DuoCore 1.7 GHz CPU, 2 G Ram, and Windows XP OS. The 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*. Each data set is decomposed into three parts: training samples, expanded samples and testing samples, which are independent of and different from each other.

- For synthetic data, the testing samples are first generated randomly from the distribution and fixed throughout the whole experiment. Then 20 independent sets of training and expanded samples are generated from the distribution by running 20 random runs.
- For real-world data, the testing samples are randomly selected from the data set first and fixed throughout the whole experiment. Then 20 independent sets of training and expanded samples are generated by running 20 random divisions of the remaining data samples with a fixed proportion.

After the generating process, we have 20 independent sets of training/expanded/testing samples. The next three steps are repeated on each of these sets.

Step 2: *Training process*. We use the training samples to train five classifiers: OCVM, CVM with updating in batch mode (CVMu), SVM using the full QP with updating in batch mode (SVMu), CVM with no updating (CVM), and SVM with no updating (SVM). OCVM calls the RSD algorithm first for redundant samples deletion and then use the preserved samples to train the classifier. During this process, the trained classifiers by CVMu and SVMu are the same as those obtained by CVM and SVM.

Step 3: *Updating process*. We use the expanded samples to update OCVM, CVMu and SVMu. The expanded samples are equally divided into 10 subsets, and the classifiers are updated with one subset in one time. Each time when the classifiers are updated, we first record the new classifier coefficients w and b and then record the correct classification rates of the classifiers on the testing samples.

Step 4: *Testing process*. After the OCVM, CVMu, SVMu, CVM and SVM classifiers are updated using all the expanded samples, they are all applied to the classification tasks on the testing samples.

An overall description of the data sets and parameters is presented in Table 1. The number of preserved samples by RSD and the number of support vectors are also given in this table. The classifier coefficients of OCVM, CVMu and SVMu, which are recorded after all expanded samples are used to update the classifiers, are compared in Table 2. The correct classification rates of OCVM, CVMu, SVMu, CVM and SVM, which are recorded in Step 4, are shown in Table 3. All experimental results reported in the tables are presented in the form: mean value ± standard deviation. The time cost of the classifiers is shown in Figs. 6 and 7 for all nine data sets. The time cost is the time consumed in classifying the newly coming samples and re-computing the classifier based on the misclassified samples. In all experiments, the correct classification rates and the time cost of CVM and SVM are used as benchmarks.

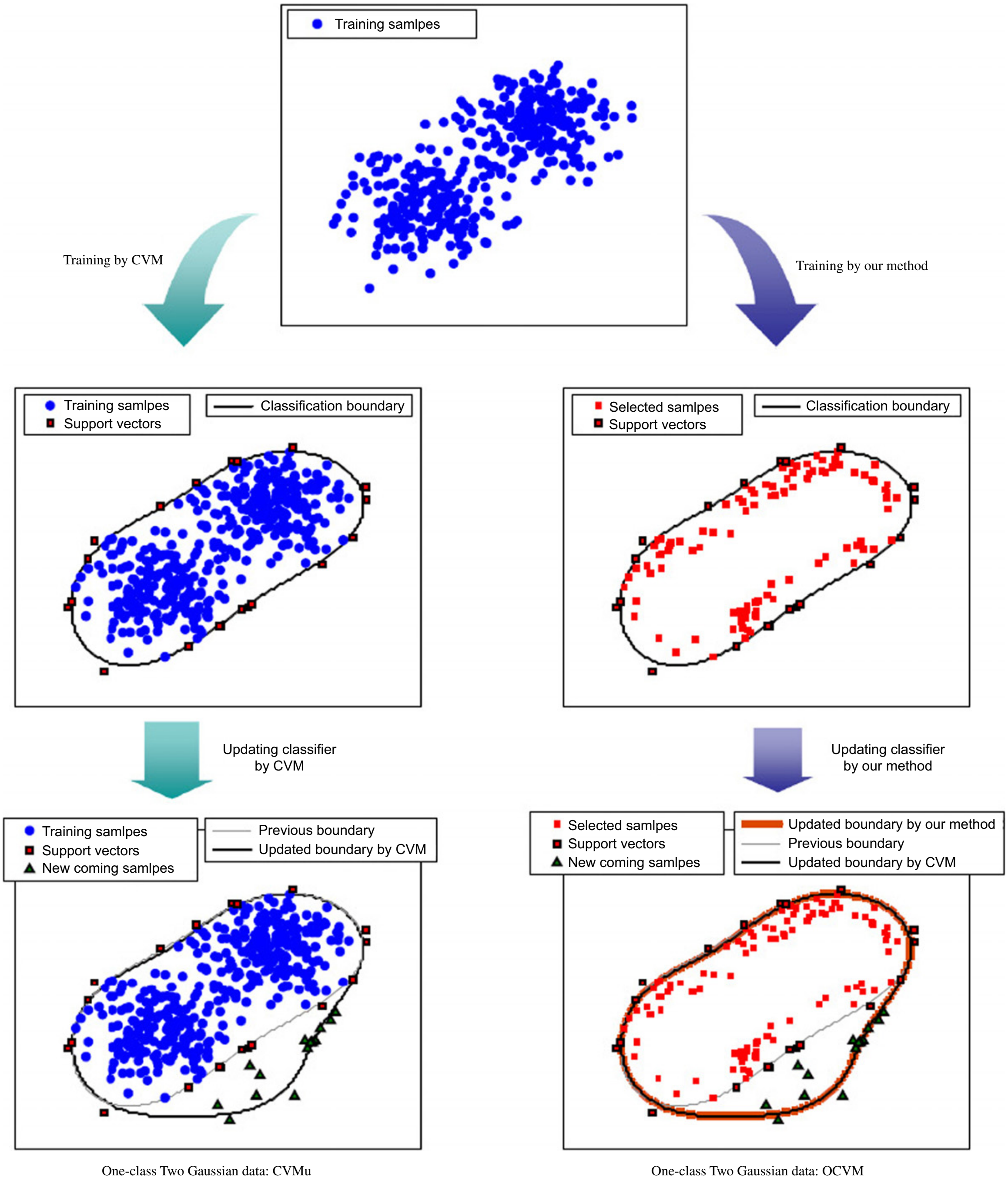

Fig. 2. Online learning process of OCVM and CVMu on one-class two Gaussian data. It can be seen that the updated classification boundaries obtained by OCVM and CVMu almost coincide, which shows that the updating results of OCVM and CVMu are almost the same.

6.2. Experiments on synthetic data

In this subsection, the proposed OCVM algorithm is tested on three synthetic data sets: Two Circles data, Two Moons data and Checkerboard Data.

6.2.1. Two Circles data

In the first experiment, the proposed OCVM algorithm is tested on the Two Circles data. The positive and negative samples lie on a disk and a ring with the same center, respectively. The disk is surrounded by the ring, but their boundaries do not coincide with each other.

An intuitive illustration of the online learning process of OCVM and CVMu is shown in Fig. 3(a) and (b). From Fig. 3 and Table 1, it can be seen that more than half of the training samples are deleted. Nevertheless, Table 2 shows that the errors between the classifier coefficients of OCVM and CVMu and of OCVM and SVMu are very small which can be neglected. The errors between the classifier coefficients during the updating process are given in Table 4, which show that during the whole updating process, the errors are very small (in the order of 10^{-5}). Meanwhile, it follows from Table 3 that the correct classification rate of OCVM is as high as that of CVMu but higher than that of SVMu. The correct classification rates of OCVM, CVMu and SVMu, recorded in Step 3 of Section 6.1, are plotted in Fig. 5(a). We can see that as the number of the expanded samples increases, the correct classification rate of OCVM also goes up and coincides with that of CVMu and SVMu. However, compared with CVMu and SVMu,

Table 1
Overall description of data sets used in Section 6 and results of samples deletion by RSD.

Data sets	Dim.	C	L	TR#	EX#	TE#	PR#	SV#
Tow Circles	2	20	2	800	200	600	331 ± 38	41 ± 3
Two Moons	2	20	2	600	180	600	261 ± 26	55 ± 3
Checkerboard	2	20	2	800	200	600	332 ± 41	70 ± 5
Building	50[a]	14	2	900	300	600	393 ± 40	173 ± 6
USPS	42[a]	20	3	800	300	800	264 ± 97	51 ± 3
UCI-Letter	16	10	2	768	256	512	344 ± 24	148 ± 5
Statlog	36	20	2	1443	481	967	650 ± 215	85 ± 8
MNIST	784	20	10	60 000	20 000	8710	10 117 ± 9139	352 ± 30
CBCL Face	361	24	2	6000	977	2000	2575 ± 242	560 ± 8

C and L are the two parameters used in the RSD algorithm. “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. “PR#”: number of preserved samples by RSD. “SV#”: number of support vectors.

[a] After dimensionality reduction, not the original dimensions.

Table 2
Comparison of classifier coefficients for all 9 experiments in Section 6.

Data sets	$\lVert w_1-w_2\rVert/\lVert w_2\rVert$	$\lvert b_1-b_2\rvert/\lvert b_2\rvert$	$\lVert w_1-w_3\rVert/\lVert w_3\rVert$	$\lvert b_1-b_3\rvert/\lvert b_3\rvert$	Order
Two Circles	6.20 ± 2.17	1.52 ± 1.36	5.48 ± 1.88	1.10 ± 1.09	10^{-5}
Two Moons	1.05 ± 0.20	2.57 ± 1.85	0.79 ± 0.22	42.83 ± 33.17	10^{-5}
Checkerboard	3.84 ± 1.15	5.30 ± 1.51	5.44 ± 0.94	18.03 ± 6.63	10^{-3}
Building	5.91 ± 1.49	2.82 ± 2.17	5.56 ± 1.06	1.94 ± 1.29	10^{-5}
USPS	0.022 ± 0.005	0.036 ± 0.020	5.40 ± 3.33	1.27 ± 0.44	10^{-5}
UCI-Letter	1.01 ± 0.22	4.71 ± 2.59	0.91 ± 0.11	2.83 ± 2.11	10^{-4}
Statlog	0.66 ± 0.20	1.11 ± 0.83	0.57 ± 0.03	0.42 ± 0.29	10^{-4}
MNIST	1.02 ± 0.18	1.67 ± 0.48	–	–	10^{-4}
CBCL Face	1.43 ± 0.30	0.62 ± 0.42	–	–	10^{-4}

$\{w_1, b_1\}$, $\{w_2, b_2\}$, and $\{w_3, b_3\}$ are obtained by OCVM, CVMu and SVMu, respectively. The errors are recorded after all expanded samples are used to update the classifiers. The relative errors are the values in the table multiplied by the order in the last column.

Table 3
Correct classification rates of OCVM, CVMu, SVMu, CVM and SVM.

Data sets	OCVM	CVMu	SVMu	CVM	SVM
Tow Circles	**99.75 ± 0.23**	**99.75 ± 0.23**	99.70 ± 0.25	89.20 ± 1.30	89.20 ± 1.30
Two Moons	98.18 ± 0.65	98.18 ± 0.65	**98.30 ± 0.67**	87.23 ± 1.82	87.23 ± 1.82
Checkerboard	93.81 ± 1.16	**93.82 ± 1.16**	93.50 ± 0.11	90.85 ± 1.09	90.85 ± 1.09
Building	95.78 ± 0.08	95.78 ± 0.08	**95.79 ± 0.07**	94.28 ± 0.56	94.28 ± 0.56
USPS	**99.75 ± 0**	**99.75 ± 0**	**99.75 ± 0**	98.85 ± 0.09	98.85 ± 0.09
UCI-Letter	**99.02 ± 0**	**99.02 ± 0**	**99.02 ± 0**	98.39 ± 0.19	98.39 ± 0.19
Statlog	**99.17 ± 0**	**99.17 ± 0**	**99.17 ± 0**	98.89 ± 0.09	98.89 ± 0.09
MNIST	**99.973 ± 0.021**	**99.973 ± 0.021**	–	99.915 ± 0.032	–
CBCL Face	**83.27 ± 0.02**	**83.27 ± 0.02**	–	83.06 ± 0.23	–

The mean of the correct rates together with the standard deviation are reported. The best classification results are marked in boldface.

the time cost of OCVM is significantly reduced, as illustrated in Fig. 6(a).

6.2.2. *Two Moons data*

The second experiment is conducted on the Two Moons data. The positive and negative samples are drawn from two half rings which do not intersect each other.

The online learning process of OCVM and CVMu is illustrated in Fig. 3(c) and (d). It can be seen that OCVM has almost the same classification boundary as CVMu, which is also supported by the errors between the classifier coefficients presented in Table 2. The correct classification rate of OCVM is the same as that of CVMu but slightly lower than that of SVMu. However, the time consumed by OCVM is much lower than that of CVMu and SVMu, as shown in Fig. 6(b).

6.2.3. *Checkerboard data*

The last set of synthetic data is the 3 × 3 Checkerboard data. The positive and negative samples are drawn from alternating squares on a checkerboard.

The online learning process of OCVM and CVMu is shown in Fig. 3(e) and (f), and the time cost consumed by OCVM, CVMu, CVM and SVM is shown in Fig. 6(c). From Tables 2 and 3 and Fig. 3 it is seen again that OCVM is much faster than CVMu and SVMu with accurate updating classifier coefficients and almost the same correct classification rate.

6.3. *Experiments on real-world data*

Fast and accurate online classifier is an important component for detection or classification tasks in real-time pattern classification

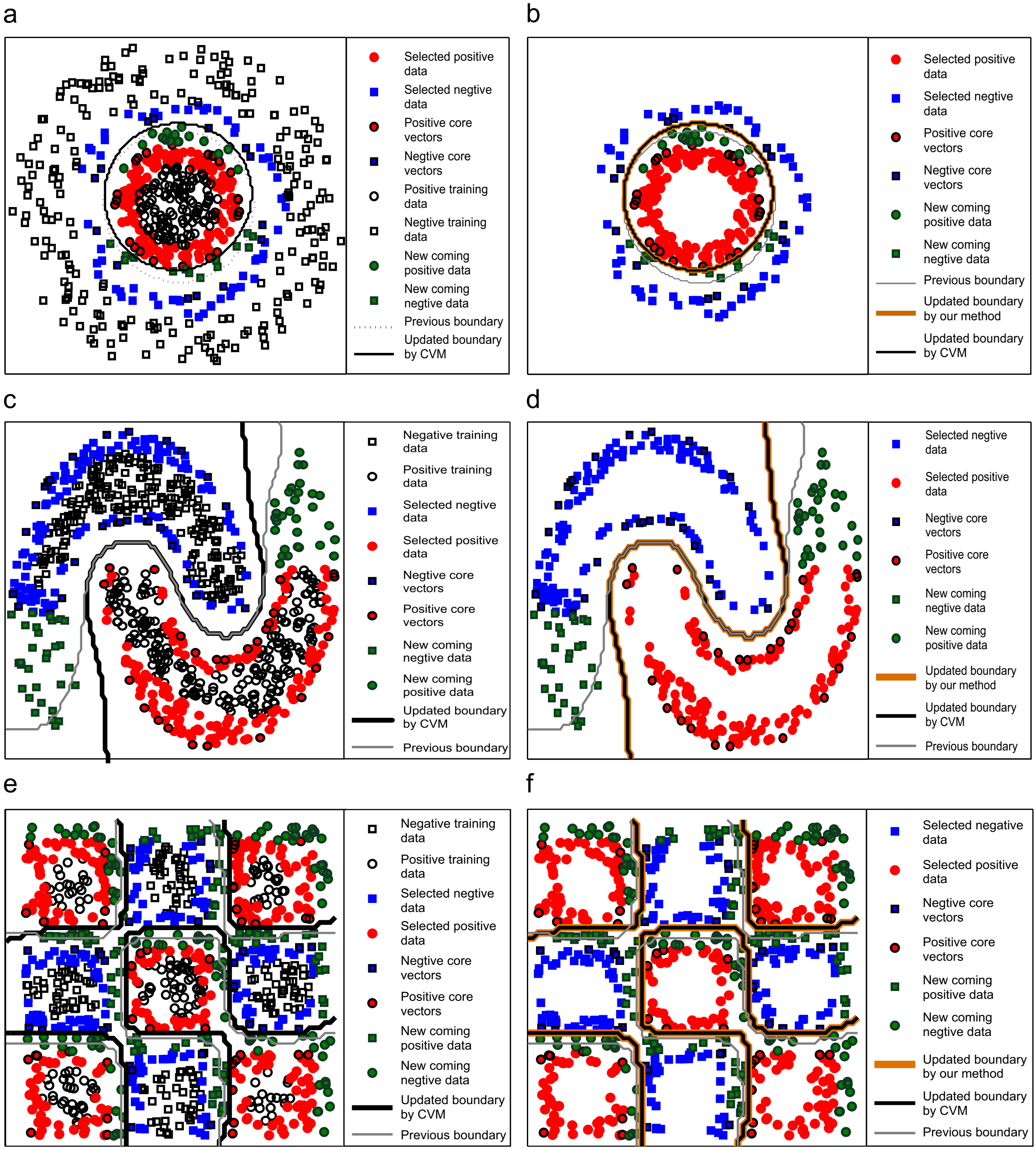

Fig. 3. Online learning process of OCVM and CVMu on synthetic data sets. For each data set, it can be seen that the updated classification boundaries obtained by OCVM and CVMu almost coincide, which shows that the updating results of OCVM and CVMu are almost the same. (a) Two Circles: CVMu. (b) Two Circles: OCVM. (c) Two Moons: CVMu. (d) Two Moons: OCVM. (e) Checkerboard: CVMu. (f) Checkerboard OCVM.

systems such as pedestrian/building detection in driver assistant systems (DAS) and zip-code recognition in accommodation distribution systems. In this subsection, the proposed OCVM classifier is applied to six real-world data sets. Some samples from these data sets are shown in Fig. 4. Details of the experimental results are stated below.

6.3.1. *Online classification of building data*

In the first experiment, the proposed OCVM classifier is applied to building detection. The building data are from a traffic database which is built up by our lab for vision-based perceptual module in DAS. Some training samples are shown in Fig. 4(a) and (b). The resolution of the data images is 32 × 16.

Before training, dimensionality reduction is first performed as a preprocessing step to remove redundant information and accelerate computation speed. Here, the locality preserving projections (LPP) [36,37] algorithm is adopted since it can handle nonlinear data and provide an explicit mapping relationship. The target dimension is set to keep 50% of the information in the sense of reconstruction error in the LPP computation.

From Table 1, it can be seen that more than half of the training samples are deleted. The errors between the classifier coefficients during the updating process are given in Table 5. Tables 5 and 2 show that OCVM has almost the same updating classifier coefficients as CVMu and SVMu (the errors are in the order of 10^{-5}). Meanwhile, Table 3 shows that the correct classification rate of OCVM is the same as that of CVMu but slightly lower than that of SVMu. The correct classification rates of OCVM, CVMu and SVMu, recorded in Step 3 of Section 6.1, are also plotted in Fig. 5(b), which means that the performance of OCVM, CVMu and SVMu is almost the same. However, compared with CVMu and SVMu, the time consumed by OCVM is significantly reduced, as illustrated in Fig. 7(a).

Table 4
Comparison of classifier coefficients for the Two Circles data.

EX#	$\|w_1-w_2\|/\|w_2\|$	$\|b_1-b_2\|/\|b_2\|$	$\|w_1-w_3\|/\|w_3\|$	$\|b_1-b_3\|/\|b_3\|$
10	3.05 ± 1.48	0.99 ± 0.75	2.85 ± 0.85	0.98 ± 0.57
20	4.79 ± 2.10	1.46 ± 1.41	4.05 ± 0.89	0.95 ± 0.70
30	4.21 ± 1.61	1.20 ± 0.82	4.13 ± 1.31	1.30 ± 1.04
40	5.21 ± 2.38	1.33 ± 1.24	3.39 ± 1.07	0.81 ± 0.55
50	4.75 ± 2.00	1.09 ± 0.70	3.71 ± 1.24	1.08 ± 0.87
60	6.20 ± 2.09	2.11 ± 1.50	4.44 ± 0.73	1.10 ± 1.01
70	6.33 ± 1.78	1.88 ± 1.51	5.34 ± 1.61	1.52 ± 1.10
80	6.53 ± 3.24	1.35 ± 0.91	5.47 ± 2.41	1.31 ± 1.03
90	5.25 ± 1.47	1.57 ± 0.92	4.43 ± 1.65	1.58 ± 0.92
100	6.20 ± 2.17	1.52 ± 1.36	5.48 ± 1.88	1.10 ± 1.09

$\{w_1, b_1\}$, $\{w_2, b_2\}$, and $\{w_3, b_3\}$ are obtained by OCVM, CVMu and SVMu, respectively. The relative errors are the values in the table multiplied by a factor 10^{-5}.

6.3.2. *Online classification of* USPS *data*

In the second experiment, the proposed OCVM classifier is applied to zip-code recognition. The training data are from the USPS data set [39]. The experiment is conducted to classify handwritten numbers '0' and '1'. Some selected samples are shown in Fig. 4(c). The resolution of the data images is 16 × 16.

Before training, we also perform LPP as a preprocessing step to remove redundant information and accelerate the computation speed. The target dimension is set to keep 50% of the information in the sense of reconstruction error in the LPP computation.

From Tables 2 and 3, we can see that OCVM has accurate updating classifier coefficients and the same correct classification rate as CVMu and SVMu. But the time cost of OCVM is much lower than that of CVMu and SVMu, as shown in Fig. 7(b).

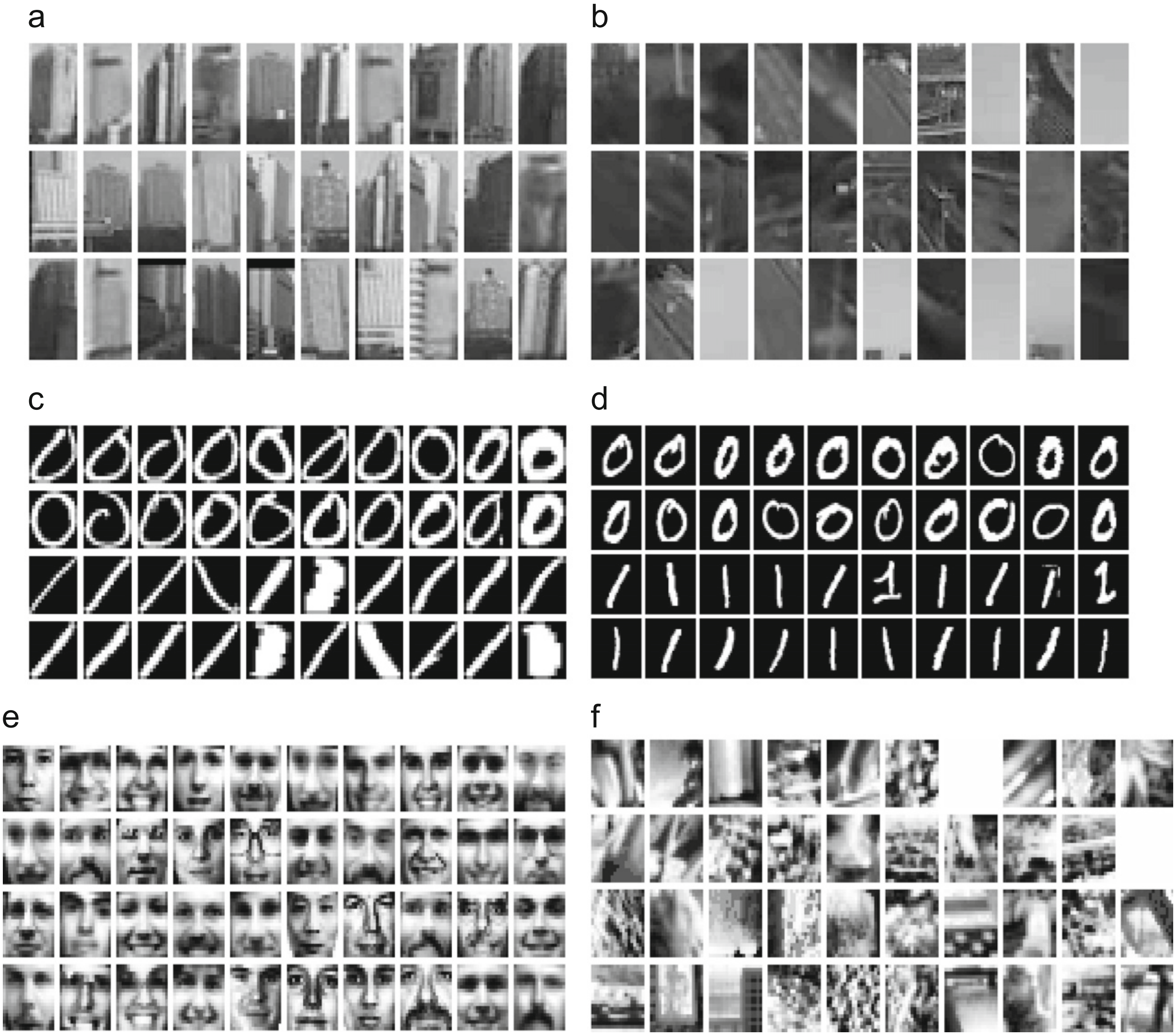

Fig. 4. Some preserved training samples used in Section 6.3. (a) Positive training samples of building data. (b) Negative training samples of building data. (c) USPS data. (d) MNIST data. (e) Positive training samples of CBCL data. (f) Negative training samples of CBCL data.

6.3.3. *Online classification of* UCI-Letter *data*

In the third experiment, the proposed OCVM classifier is applied to the UCI letter recognition data set [39], which contains 26 English letters based on 20 different fonts. Each sample in the data set consists of 16 numerical attributes that record statistical moments and edge counts of the images of letters. Our experiment is designed to classify letters ‘A’ and ‘B’.

Table 5
Comparison of classifier coefficients for the building data.

EX#	$\|w_1-w_2\|/\|w_2\|$	$\|b_1-b_2\|/\|b_2\|$	$\|w_1-w_3\|/\|w_3\|$	$\|b_1-b_3\|/\|b_3\|$
10	4.45 ± 1.08	1.51 ± 0.88	4.76 ± 0.83	1.77 ± 0.62
20	5.06 ± 1.37	1.66 ± 1.15	4.35 ± 0.78	1.99 ± 1.11
30	10.43 ± 1.53	3.85 ± 7.33	4.75 ± 0.80	1.49 ± 0.81
40	4.96 ± 0.52	2.01 ± 1.37	4.92 ± 0.65	1.95 ± 1.05
50	4.78 ± 1.11	2.57 ± 1.74	4.99 ± 0.71	1.60 ± 0.66
60	6.34 ± 1.13	2.35 ± 1.85	5.16 ± 1.05	2.26 ± 0.84
70	6.04 ± 1.64	2.38 ± 1.63	4.89 ± 0.84	2.05 ± 1.04
80	5.67 ± 1.91	2.30 ± 1.91	4.73 ± 0.56	2.06 ± 0.93
90	5.96 ± 1.03	3.72 ± 2.13	5.35 ± 0.96	2.30 ± 1.74
100	5.91 ± 1.49	2.82 ± 2.17	5.56 ± 1.06	1.94 ± 1.29

$\{w_1,\ b_1\}$, $\{w_2,\ b_2\}$, and $\{w_3,\ b_3\}$ are obtained by OCVM, CVMu and SVMu, respectively. The relative errors are the values in the table multiplied by a factor 10^{-5}.

From Tables 2 and 3 and Fig. 7(c), it is seen that OCVM is much faster than CVMu and SVMu with accurate updating classifier coefficients and the same classification ability.

6.3.4. *Online classification of* Statlog *data*

The fourth experiment is conducted on the UCI Statlog (Landset Satellite) data set [39], which contains images of six different types of soil. The attributes of the samples consist of the multi-spectral values of pixels in 3 × 3 neighborhoods in a satellite image. The classification task aims to predict the type of soil given the multi-spectral values.

The errors between the classifier coefficients, the correct classification rates and the time cost of the classifiers are shown in Tables 2 and 3 and Fig. 7(d), respectively. The OCVM algorithm has the same classification result as CVMu and SVMu on this data set with much less time cost.

6.3.5. *Online classification of* MNIST *data*

In the fifth experiment, the proposed OCVM classifier is tested on a very large data set taken from the MNIST database [38]. The experiment is designed for online classification of digits ‘0’ and ‘1’. Some training samples are shown in Fig. 4(d). The resolution of the training images is 28 × 28.

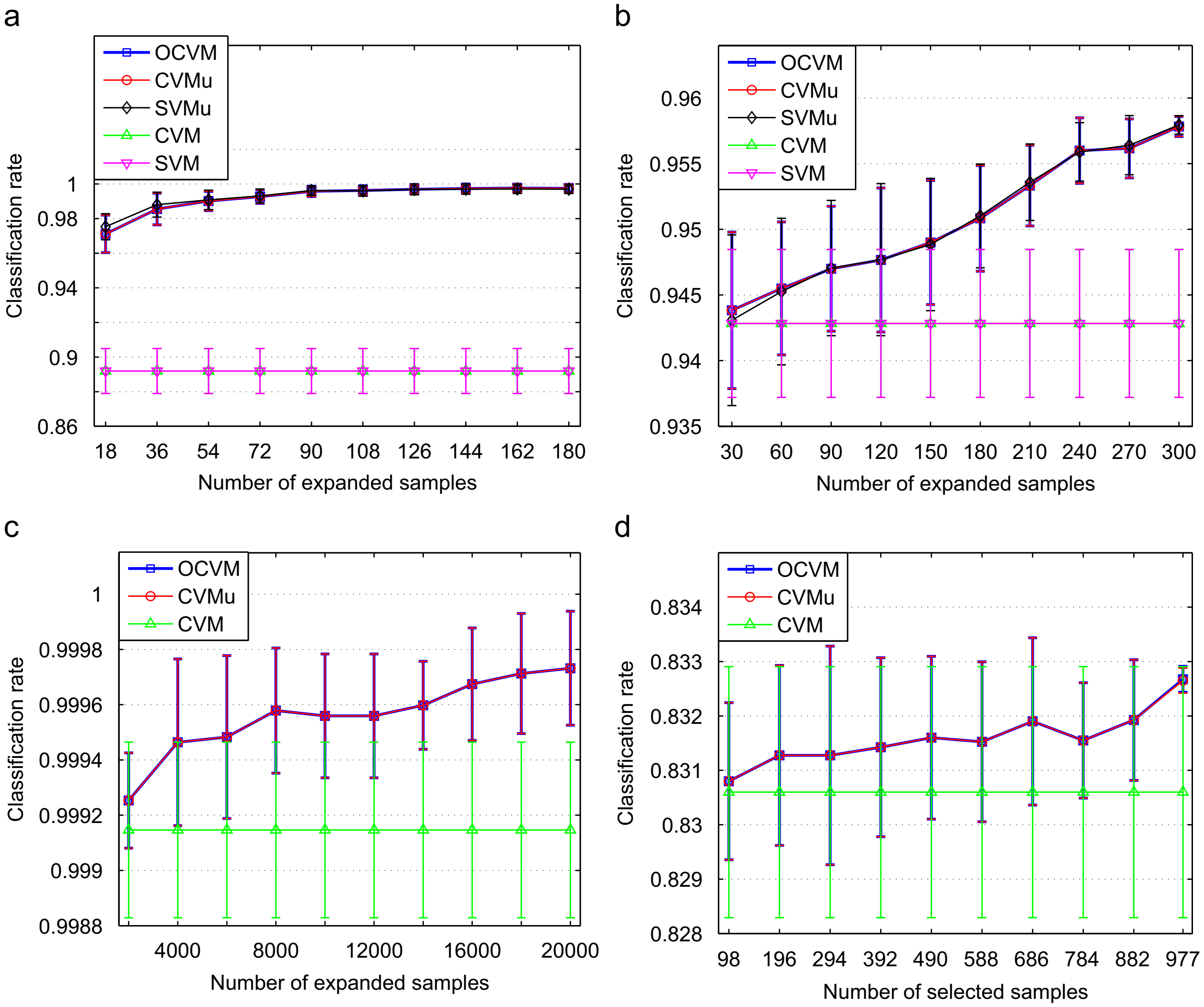

Fig. 5. Correct classification rates in the updating process on four testing data sets. (a) Two Circles data. (b) Building data. (c) MNIST. (d) CBCL Face.

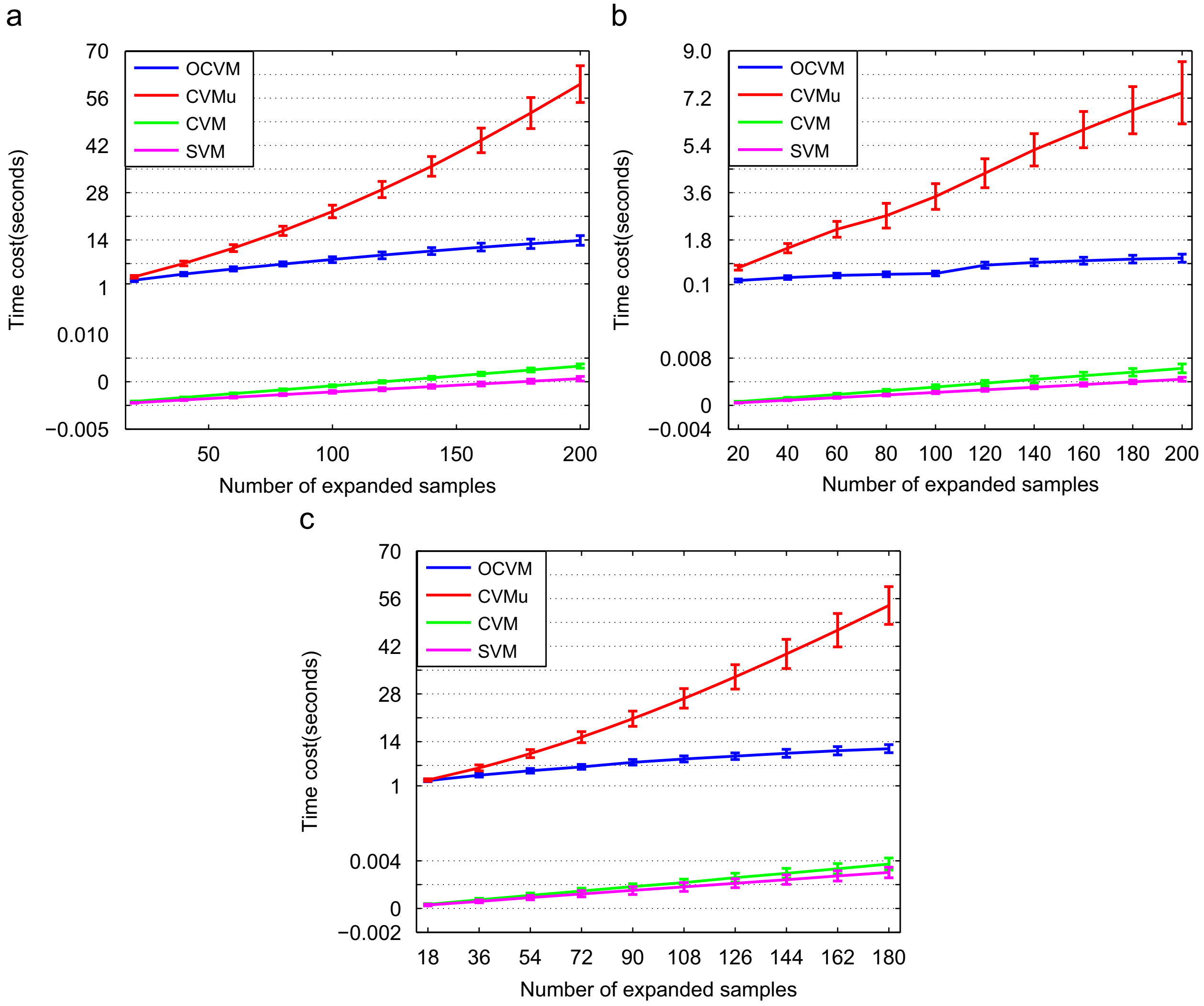

Fig. 6. Time cost on synthetic data sets. (a) Checkerboard. (b) Two Moons. (c) Two circles.

From Tables 1, 2 and 6 it is seen that although nearly 70% of the training samples are deleted, the errors between the classifier coefficients of the OCVM and CVMu classifiers during and after the updating process are very small. Meanwhile, the correct classification rate of OCVM is as high as CVM both during and after the updating process, as shown in Table 3 and Fig. 5(c). However, Fig. 7(e) shows that the time cost of OCVM is significantly reduced. The time consumed by OCVM is less than 5000 s to update with 20 000 samples and classify 8710 samples, while the time consumed by CVMu for the same operations is more than 24 000 s. It should be noted that the experimental results with SVMu and SVM are not available for such a very large data set due to the limited computational resources.

6.3.6. *Online classification of* CBCL Face *data*

The last experiment is conducted on a large data set, the CBCL Face data set [40] which consists of 2901 human face images and 6076 non-face images.

From Tables 1, 2 and 7, we can see again that OCVM obtains accurate updating results for the classifier coefficients through efficient samples deletion. The classification results are illustrated in Table 3 and Fig. 5(d) and show that OCVM has the same classification results as CVMu. However, the time cost of CVMu is 15 times that of OCVM, as shown in Fig. 7(f). The experimental results with SVMu and SVM are again not available for such a large data set due to the limited computational resources.

6.4. *Remarks*

There are five user-defined parameters in the proposed OCVM classifier: the kernel parameter σ, the cost coefficient C, the radius parameter ε, Ω and L. In our experiments, the five parameters are determined based on the following principles.

- In the experiments on real world data, σ is taken as the mean of the pairwise distances among the training samples:

$$\frac{1}{m^2}\sum_{i,j=1}^{m}\|x_i - x_j\|^2.$$

- C is selected within an user-defined interval such that the RSD algorithm can delete as many training samples as possible.
- ε is fixed to be 10^{-6}.
- Ω has a Boolean value for the implementation of the code: $\Omega = 0$ means that the RSD algorithm will be used to delete the

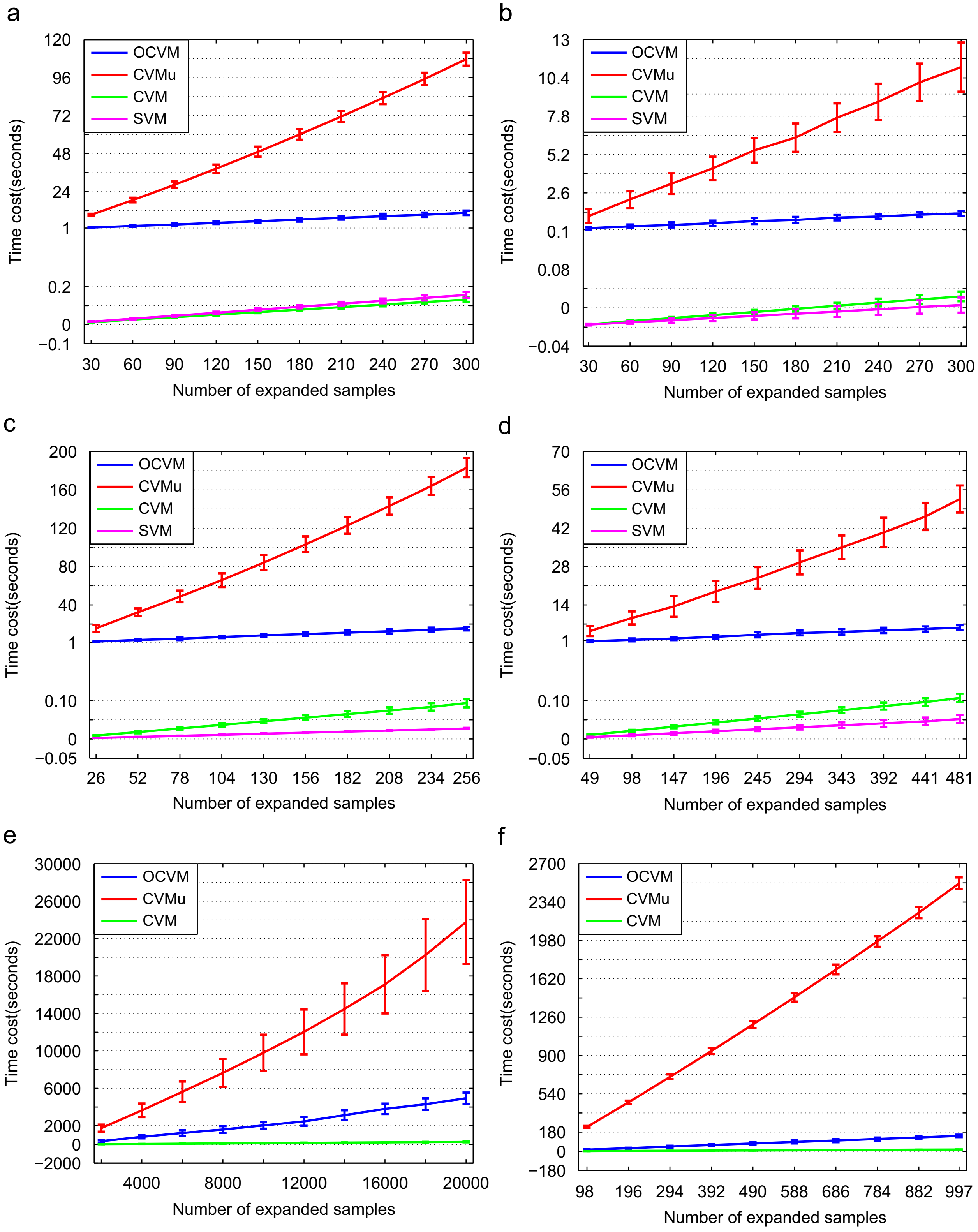

Fig. 7. Time cost on real-world data sets. (a) Building. (b) USPS. (c) UCI Letter. (d) Statlog. (e) MNIST. (f) CBCL Face.

redundant samples and $\Omega = 1$ indicates that the RSD algorithm will not be utilized. In all the experiments, Ω is set to be 0.

- L is selected empirically depending on the size of the data sets. When the size of data sets is large (e.g., over 10 000 data points), L should be set to be a large value (e.g., around 10); otherwise, L is set to be small (e.g., 2 or 3).

7. Conclusion

In this paper, an online CVM classifier with online MEB adjustment is proposed. During the training process, many redundant training samples are efficiently removed, which leads to a significant saving of training time of the CVM classifier.

Table 6
Comparison of classifier coefficients for the MNIST data.

EX#	$\|w_1-w_2\|/\|w_2\|(\times 10^{-5})$	$\|b_1-b_2\|/\|b_2\|(\times 10^{-5})$
10	0.89 ± 0.12	2.24 ± 1.40
20	0.83 ± 0.17	1.52 ± 1.50
30	0.92 ± 0.26	1.67 ± 0.87
40	0.88 ± 0.20	1.73 ± 2.20
50	0.64 ± 0.14	1.28 ± 1.46
60	0.93 ± 0.23	1.54 ± 0.93
70	0.84 ± 0.19	1.11 ± 1.08
80	0.98 ± 0.06	1.78 ± 1.13
90	1.01 ± 0.08	0.58 ± 0.49
100	1.02 ± 0.18	1.67 ± 0.48

$\{w_1, b_1\}$ and $\{w_2, b_2\}$ are obtained by OCVM and CVMu, respectively.

Table 7
Comparison of classifier coefficients for the CBCL Face data.

EX#	$\|w_1-w_2\|/\|w_2\|(\times 10^{-5})$	$\|b_1-b_2\|/\|b_2\|(\times 10^{-5})$
10	1.48 ± 0.40	0.50 ± 0.49
20	1.72 ± 0.37	0.61 ± 0.40
30	1.46 ± 0.20	0.61 ± 0.29
40	1.49 ± 0.26	0.46 ± 0.41
50	1.42 ± 0.27	0.38 ± 0.30
60	1.54 ± 0.25	0.61 ± 0.29
70	1.56 ± 0.24	0.51 ± 0.38
80	1.62 ± 0.22	0.49 ± 0.37
90	1.52 ± 0.41	0.60 ± 0.43
100	1.43 ± 0.30	0.62 ± 0.42

$\{w_1, b_1\}$ and $\{w_2, b_2\}$ are obtained by OCVM and CVMu.

During the online classification process, the classifier is updated online based on the preserved samples and the newly coming misclassified samples.

From the theoretical point of view, the deleted samples are proved to be enclosed in the accurate MEB so that the deletion of samples will not affect the final trained classifier. From the application point of view, online adjustment of the CVM classifier is achieved with the proposed adaptive MEB learning method with a similar classification accuracy. Experimental results on both synthetic and real-world data have illustrated the effectiveness of the proposed method.

From our experiments, it is found that the proposed method is robust to all the parameters except the cost coefficient C. If C is too large or too small, the performance of the proposed method will be poor. How to find an optimal C automatically will be one of our future work. Furthermore, in this paper we only consider the Gaussian kernel which fits the proposed method well. In the future, we will also test the performance of the online CVM classifier with other kernel functions such as the linear and polynomial kernels.

Acknowledgements

This work was partly supported by the NNSF of China grant no. 90820007, 60975002, the Outstanding Youth Fund of the NNSF of China Grant no. 60725310, the 863 Program of China Grant no. 2007AA04Z228 and the 973 Program of China Grant no. 2007CB311002. The authors thank the referees for their invaluable comments and suggestions which helped improve the paper greatly.

Appendix A. Lemmas and their proofs

We first have the following basic result which was proved in [34].

Lemma 1. *Let $B(c, R)$ be the minimum enclosing ball of the set $P \subset \mathbb{R}^d$. Then any closed half-space which contains the center c also contains at least one point in P which is at distance R from the center c. It follows that for any point q at distance K from c there is a point $q' \in P$ at distance at least $\sqrt{R^2+K^2}$ from q.*

Using Lemma 1, the following two lemmas can be proved.

Lemma 2. *Let D_{t_1} be the furthest distance between data points in P and the center c_t at the t-th iteration. Then $r_{t+1} \leqslant (D_{t_1}^2+r_t^2)/2D_{t_1}$.*

Proof. The lemma is proved by contradiction. Suppose the conclusion is not true, that is,

$$r_{t+1} < (D_{t_1}^2+r_t^2)/2D_{t_1}.$$

Let p be the furthest point in the set P from the center c_t. Then we have $\|c_{t+1}-p\| \le r_{t+1}$ and $\|c_t-p\| = D_{t_1}$. By the triangle inequality

$$\|c_t-c_{t+1}\| + \|c_{t+1}-p\| \ge \|c_t-p\|,$$

it follows that

$$\|c_t-c_{t+1}\| \ge D_{t_1}-r_{t+1} > D_{t_1}-(D_{t_1}^2+r_t^2)/2D_{t_1} = (D_{t_1}^2-r_t^2)/2D_{t_1}.$$

For any $q' \in S_t$, we have $\|c_{t+1}-q'\| \le r_{t+1}$, so

$$\|c_{t+1}-q'\|^2 \le r_{t+1}^2 < (D_{t_1}^2+r_t^2)^2/4D_{t_1}^2 = (D_{t_1}^2-r_t^2)^2/4D_{t_1}^2+r_t^2 < \|c_t-c_{t+1}\|^2+r_t^2. \quad (10)$$

Consider c_{t+1} as the point q mentioned in Lemma 1 and let $\|c_t-c_{t+1}\| = K$. By Lemma 1 we know that there is a point $q_1 \in S_t$ such that $\|c_{t+1}-q_1\| \ge \sqrt{K^2+r_t^2}$. On the other hand, from (10) we have $\|c_{t+1}-q'\| < \sqrt{K^2+r_t^2}$ for any $q' \in S_t$. This is a contradiction. □

Lemma 3.

$$\|c-c_t\| \le \min\{\sqrt{D_{t_1}^2-R^2}, \sqrt{R^2-r_t^2}\}.$$

Proof. We first prove that $\|c-c_t\| \le \sqrt{D_{t_1}^2-R^2}$ by contradiction. Suppose this is not true, that is,

$$\|c-c_t\| > \sqrt{D_{t_1}^2-R^2}.$$

Then for any $q' \in \partial B(c,R)$, that is, $\|q'-c\| = R$, we have

$$\|q'-c_t\| \le D_{t_1} = \sqrt{(D_{t_1}^2-R^2)+R^2} < \sqrt{\|c-c_t\|^2+R^2}. \quad (11)$$

Consider c_t as the point p mentioned in Lemma 1. Then $\|q'-c_t\| \ge \sqrt{R^2+\|c-c_t\|^2}$, which contradicts (11). Similarly, we can prove that $\|c-c_t\| \le \sqrt{R^2-r_t^2}$. The lemma is thus proved. □

Lemma 4. *Let D_{t_2} be the second furthest distance between data points in P and the center c_t at the t-th iteration. Then $(D_{t_1}^2+r_t^2)/2D_{t_1} \le R \le (D_{t_1}+D_{t_2})/2$.*

Proof. The left inequality is obvious by Lemma 2. We now prove the right inequality. If $D_{t_1} = D_{t_2}$, then the conclusion is straightforward. Otherwise, suppose p is the furthest point in P from the center c_t, that is, $\|p-c_t\| = D_{t_1}$ and let

$$c' = c_t + \frac{D_{t_1}-D_{t_2}}{2\|p-c_t\|}(p-c_t).$$

For any $q \in P \backslash \{p\}$, $\|q-c_t\| \le D_{t_2}$, so we have

$$\|q-c'\| \le \|q-c_t\| + \|c'-c_t\| \le D_{t_2} + (D_{t_1}-D_{t_2})/2 = (D_{t_1}+D_{t_2})/2.$$

By a direct calculation we get that $\|p-c'\| = (D_{t_1}+D_{t_2})/2$. Therefore, it is clear that all points in P are in the ball $B(c',(D_{t_1}+D_{t_2})/2)$. The proof is complete. □

Lemma 5. *If the RSD algorithm stops at the t-th iteration, then* $\|c-c_t\| \le \sqrt{2\varepsilon+\varepsilon^2}R$.

Proof. If the RSD algorithm stops at the t-th iteration, then we have

$$r_t \le R \le D_{t_1} \le (1+\varepsilon)r_t. \qquad (12)$$

Thus,

$$R^2 - r_t^2 \le (\varepsilon^2+2\varepsilon)r_t^2 \le (\varepsilon^2+2\varepsilon)R^2,$$

$$D_{t_1}^2 - R^2 \le (1+\varepsilon)^2 r_t^2 - R^2 \le (\varepsilon^2+2\varepsilon)R^2.$$

From this and Lemma 3 it follows that $\|c-c_t\| \le \sqrt{2\varepsilon+\varepsilon^2}R$. □

Lemma 5 and (12) mean that, when the RSD algorithm stops at the t-th iteration, the approximate MEB $B(c_t, r_t)$ is very close to the accurate MEB $B(c, R)$ since the center c_t and the radius r_t are very close to c and R, respectively, noting that ε is a very small tolerance number given initially.

References

[1] E. Osuna, R. Freund, F. Girosi, Training support vector machines: an application to face detection, in: Proceedings of the 1997 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97), Puerto Rico, June 1997, pp. 130–136.

[2] P. Shih, C. Liu, Face detection using discriminating feature analysis and support vector machine, Pattern Recognition 39 (2) (2006) 260–276.

[3] A. Mukhopadhyay, U. Maulik, Towards improving fuzzy clustering using support vector machine: application to gene expression data, Pattern Recognition 42 (11) (2009) 2744–2763.

[4] X.B. Cao, H. Qiao, A low-cost pedestrian detection system with a single optical camera, IEEE Transactions on Intelligent Transportation Systems 9 (1) (2008) 58–67.

[5] M. Adankon, M. Cheriet, Model selection for the LS-SVM. Application to handwriting recognition, Pattern Recognition 42 (12) (2009) 3264–3270.

[6] T. Joachims, Text categorization with support vector machines: learning with many relevant features, in: Proceedings of the European Conference on Machine Learning, vol. 1398, 1998, pp. 137–142.

[7] K. Kim, K. Jung, S. Park, H. Kim, Support vector machine-based text detection in digital video, Pattern Recognition 34 (2) (2001) 527–529.

[8] J. Hong, J. Min, U. Cho, S. Cho, Fingerprint classification using one-vs-all support vector machines dynamically ordered with naïve Bayes classifiers, Pattern Recognition 41 (2) (2008) 662–671.

[9] S. Li, J. Kwok, H. Zhu, Y. Wang, Texture classification using the support vector machines, Pattern Recognition 36 (2003) 2883–2893.

[10] A. Shilton, M. Palaniswami, D. Ralph, A.C. Tsoi, Incremental training of support vector machines, IEEE Transactions on Neural Networks 16 (1) (2005) 114–131.

[11] T. Joachims, Making large-scale SVM learning practical, in: Advances in Kernel Methods: Support Vector Learning, 1999, pp. 169–184.

[12] J.C. Platt, Fast training of support vector machines using sequential minimal optimization, in: Advances in Kernel Methods: Support Vector Learning, 1999, pp. 185–208.

[13] C.J. Lin, On the convergence of the decomposition method for support vector machines, IEEE Transactions on Neural Networks 12 (6) (2001) 1288–1298.

[14] R.E. Fan, P.H. Chen, C.J. Lin, Working set selection using second order information for training support vector machines, Journal of Machine Learning Research 6 (2005) 1889–1918.

[15] H. Qiao, Y. Wang, B. Zhang, A simple decomposition algorithm for support vector machines with polynomial-time convergence, Pattern Recognition 40 (2007) 2543–2549.

[16] I.W. Tsang, J.T. Kwok, P.M. Cheung, Core vector machines: fast SVM training on very large data sets, Journal of Machine Learning Research 6 (2005) 363–392.

[17] I.W. Tsang, J.T. Kwok, J.M. Zurada, Generalized core vector machines, IEEE Transactions on Neural Networks 17 (5) (2006) 1126–1140.

[18] L. Bottou, O. Chapelle, D. DeCoste, J. Weston (Eds.), Large Scale Kernel Machines, MIT Press, Cambridge, MA, USA, 2007.

[19] A.K. Menon, Large-scale support vector machines: algorithms and theory, available at 〈http://cseweb.ucsd.edu/~akmenon/ResearchExam.pdf〉.

[20] S. Cheng, F. Shih, An improved incremental training algorithm for support vector machines using active query, Pattern Recognition 40 (3) (2007) 964–971.

[21] N. Syed, H. Liu, K. Sung, Handling concept drifts in incremental learning with support vector machines, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'99), San Diego, 1999, pp. 317–321.

[22] B. Peng, Z. Sun, X. Xu, SVM-based incremental active learning for user adaptation for online graphics recognition system, in: Proceedings of the First International Conference on Machine Learning and Cybernetics, Beijing, 2002, pp. 1379–1386.

[23] A. Bordes, L. Bottou, P. Gallinari, J. Weston, Solving multiclass support vector machines with LaRank, in: Proceedings of the 24th International Conference on Machine Learning, vol. 227, 2007, pp. 89–96.

[24] J. Kivinen, A.J. Smola, R.C. Williamson, Online learning with kernels, IEEE Transactions on Signal Processing 52 (8) (2004) 2165–2176.

[25] L. Bottou, Online learning and stochastic approximations, in: D. Saad (Ed.), Online Learning in Neural Networks, Cambridge University Press, Cambridge, UK, 1998, pp. 9–42.

[26] L. Bottou, Y. LeCun, Large scale online learning, Advances in Neural Information Processing Systems 16 (2004).

[27] L. Bottou, O. Bousquet, The tradeoffs of large scale learning, Advances in Neural Information Processing Systems 20 (2008).

[28] G. Cauwenberghs, T. Poggio, Incremental and decremental support vector machine learning, Advances in Neural Information Processing Systems 12 (2000) 409–415.

[29] S. Agarwal, V. Saradhib, H. Karnick, Kernel-based online machine learning and support vector reduction, Neurocomputing 71 (7–9) (2008) 1230–1237.

[30] M. Davy, F. Desobry, A. Gretton, C. Doncarli, An online support vector machine for abnormal events detection, Signal Processing 86 (8) (2006) 2009–2025.

[31] M. Awad, X. Jiang, Y. Motai, Incremental support vector machine framework for visual sensor networks, EURASIP Journal on Advances in Signal Processing 1 (2007) 222–236.

[32] K.W. Lau, Q.H. Wu, Online training of support vector classifier, Pattern Recognition 36 (8) (2003) 1913–1920.

[33] F. Orabona, C. Castellini, B. Caputo, L. Jie, G. Sandini, On-line independent support vector machines, Pattern Recognition 43 (2010) 1402–1412.

[34] M. Bădoiu, K. Clarkson, Optimal core-sets for balls, Computational Geometry 40 (1) (2008) 14–22.

[35] D. Tax, R. Duin, Support vector data description, Machine Learning 54 (1) (2004) 45–66.

[36] X. He, P. Niyogi, Locality preserving projections, Advances in Neural Information Processing Systems 15 (2003).

[37] X. He, S. Yan, Y. Hu, P. Niyogi, H. Zhang, Face recognition using Laplacianfaces, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (3) (2005) 328–340.

[38] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.

[39] A. Asuncion, D.J. Newman, UCI Machine Learning Repository, University of California, School of Information and Computer Science, Irvine, CA, 2007 〈http://www.ics.uci.edu/~mlearn/MLRepository.html〉.

[40] CBCL Face Database #1, MIT Center For Biological and Computation Learning 〈http://www.ai.mit.edu/projects/cbcl〉.