Pattern Recognition 49 (2016) 55–64

Contents lists available at ScienceDirect

Pattern Recognition

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

# Smoothly approximated support vector domain description

Songfeng Zheng *

*Department of Mathematics, Missouri State University, 901 S. National Avenue, Springfield, MO 65897, USA*

ARTICLE INFO

*Article history:*
Received 14 February 2013
Received in revised form
21 November 2014
Accepted 4 July 2015
Available online 17 July 2015

*Keywords:*
Support vector domain description
Smooth approximation
Quadratic programming
conjugate gradient

ABSTRACT

Support vector domain description (SVDD) is a well-known tool for pattern analysis when only positive examples are reliable. The SVDD model is often fitted by solving a quadratic programming problem, which is time consuming. This paper attempts to fit SVDD in the primal form directly. However, the primal objective function of SVDD is not differentiable which prevents the well-behaved gradient based optimization methods from being applicable. As such, we propose to approximate the primal objective function of SVDD by a differentiable function, and a conjugate gradient method is applied to minimize the smoothly approximated objective function. Extensive experiments on pattern classification were conducted, and compared to the quadratic programming based SVDD, the proposed approach is much more computationally efficient and yields similar classification performance on these problems.

© 2015 Elsevier Ltd. All rights reserved.

## 1. Introduction

There exist a range of pattern recognition problems, such as novelty detection, where the task is to discriminate the pattern of interest from imposters. In such cases, positive examples for training are relatively easier to obtain and more reliable. However, although negative examples are very abundant, it is usually difficult to sample enough useful negative examples to adequately model the imposters since they may belong to any class. In this situation, it is often reasonable to assume positive examples to cluster in a certain way. Under this assumption, the goal is to accurately describe the class of positive examples as opposed to the very wide range of negative examples, which are not of interest.

To this end, Tax and Duin [30–32] developed a support vector domain description (SVDD) method, which fits a tight hypersphere in the nonlinearly transformed feature space to enclose most of the positive examples. Thus, SVDD could be regarded as a description of the class of interest. Extensive experiments show that SVDD can correctly identify some negative examples even though it has not seen any negative example during the training phase [30–32].

SVDD is, like support vector machine (SVM) [34, chapter 10], a kernel method, thus inherits all the related advantages of SVM. Since it was proposed, SVDD has been applied to various application problems, including image classification [39], remote sensing image analysis [2,23,24], medical image analysis [29], machine diagnostics [33,38], and multi-class classification problems [18,37], among others. Furthermore, SVDD is a preliminary step for support vector clustering [3,19,20].

Similar to SVM, the formulation of SVDD leads us to a quadratic programming problem (see Section 2 for more details). Although the decomposition techniques [25,26] or sequential minimization methods [27] could be employed to solve the quadratic programming problem, the training of SVDD has time complexity about $O(n^3)$ (see the end of Section 2 for details), where $n$ is the training set size. High training cost is undesirable, especially for model selection and some feature selection methods, where the training algorithm often needs to run multiple times. Therefore, it is highly appreciated to develop time-efficient yet accurate enough training algorithms for SVDD.

As an alternative, we can fit the SVDD model by directly optimizing the primal objective function, as the similar work for SVM [9]. However, the primal objective function of SVDD is not differentiable which prevents gradient based methods [4,36] from being applicable, although they are easy to implement, and converge fast to at least a local optimum. As such, we introduce a smooth approximation to the primal objective function of SVDD, which is an upper bound of the primal objective function and converges uniformly to the primal objective function as the controlling smoothing parameter increases. Then, conjugate gradient method is employed to minimize the proposed smoothly approximated objective function. We test the proposed approach on face detection and handwritten digit recognition problems, and detailed performance comparison on these problems demonstrates that the proposed smoothly approximated SVDD (SA-SVDD) often yields testing accuracy very close to that of the quadratic programming based SVDD (QP-SVDD). However, SA-SVDD is much more computationally efficient than QP-SVDD.

* Tel.: +1 417 836 6037; fax: +1 417 836 6966.
*E-mail address:* SongfengZheng@MissouriState.edu

http://dx.doi.org/10.1016/j.patcog.2015.07.003
0031-3203/© 2015 Elsevier Ltd. All rights reserved.

The rest of this paper is organized as follows: Section 2 briefly reviews the formulation of SVDD; Section 3 proposes the smoothly approximated SVDD model, and a conjugate gradient method is presented to minimize the smoothed objective function; a brief computational complexity analysis is also presented in Section 3; Section 4 compares the classification performances and training time of the proposed SA-SVDD algorithm to the original QP-SVDD on two publicly available real-world datasets; finally, Section 5 summarizes this paper and discusses some future research directions.

## 2. Support vector domain description

This section briefly reviews the general formulation of support vector domain description (SVDD) with only positive examples (Section 2.1) and with both positive and negative examples (Section 2.2). Refer to [30–32] for more detailed derivations.

### 2.1. SVDD with positive examples

Given training data $\{\mathbf{x}_i, i=1,\ldots,n\}$ with the feature vector $\mathbf{x}_i \in \mathbf{R}^p$, SVDD is looking for a hypersphere (in a high dimensional Hilbert feature space $\mathcal{H}$ where the examples have been mapped through a nonlinear transformation $\Phi$) of radius $R>0$ and center $\mathbf{c}$ with a minimum volume containing most of the data. Therefore, we have to minimize $R^2$ with constraints $\|\Phi(\mathbf{x}_i)-\mathbf{c}\|^2 \le R^2$, for $i=1,\ldots,n$. In addition, since the training sample might contain outliers, we introduce a set of slack variables $\xi_i \ge 0$, as in the framework of support vector machine (SVM) [34, chapter 10]. The slack variable $\xi_i$ measures how much the squared distance from the $i$th training example to the center exceeds the radius squared. Therefore, the slack variable could be understood as a measure of error. Taking the constraints into account, the problem becomes

$$\min_{R,\mathbf{c},\xi} \quad F(R,\mathbf{c},\boldsymbol{\xi}) = R^2 + C\sum_{i=1}^{n}\xi_i, \tag{1}$$

with constraints

$$\|\Phi(\mathbf{x}_i)-\mathbf{c}\|^2 \le R^2+\xi_i, \quad \xi_i \ge 0 \ \text{ for } i=1,\ldots,n, \tag{2}$$

where $\boldsymbol{\xi}=(\xi_1,\ldots,\xi_n)'$ is the vector of slack variables, and the parameter $C>0$ controls the tradeoff between the volume of the hypersphere and the permitted errors.

The Lagrangian dual of the above optimization problem is

$$\min_{\boldsymbol{\alpha}} \quad L(\boldsymbol{\alpha}) = \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j K(\mathbf{x}_i,\mathbf{x}_j) - \sum_{i=1}^{n}\alpha_i K(\mathbf{x}_i,\mathbf{x}_i), \tag{3}$$

with constraints

$$\sum_{i=1}^{n}\alpha_i = 1, \quad 0 \le \alpha_i \le C \ \text{ for } i=1,\ldots,n, \tag{4}$$

where $\boldsymbol{\alpha}=(\alpha_1,\ldots,\alpha_n)'$ with $\alpha_i$ being the Lagrangian multiplier for the $i$th constraint, and $K(\mathbf{x}_i,\mathbf{x}_j) = <\Phi(\mathbf{x}_i),\Phi(\mathbf{x}_j)>$ is the kernel function which satisfies Mercer's condition [34, chapter 10]. From the Karush–Kuhn–Tucker (KKT) conditions [4, chapter 3] [6, chapter 5], the center of the hypersphere in the high dimensional feature space $\mathcal{H}$ can be represented in terms of the Lagrangian multipliers as

$$\mathbf{c} = \sum_{i=1}^{n}\alpha_i\Phi(\mathbf{x}_i). \tag{5}$$

Once the parameters $\alpha_i$'s are obtained, the radius $R$ can be computed from the set of support vectors.

In decision making stage, if the distance from a new example $\mathbf{x}$ is less than the radius $R$, it is classified as a positive example; otherwise, it is classified as a negative example. Thus, the decision rule is

$$f(\mathbf{x}) = \operatorname{sign}\left(R^2 - \|\Phi(\mathbf{x}) - \sum_{i=1}^{n}\alpha_i\Phi(\mathbf{x}_i)\|^2\right) = \operatorname{sign}\left(2\sum_{i=1}^{n}\alpha_i K(\mathbf{x}_i,\mathbf{x}) - K(\mathbf{x},\mathbf{x}) + b\right), \tag{6}$$

where $b = R^2 - \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j K(\mathbf{x}_i,\mathbf{x}_j)$.

### 2.2. 2-Class SVDD

If negative examples are available, we could integrate this part of information to the formulation of SVDD. In this situation, we would prefer the hypersphere enclosing as many positive examples as possible and excluding as many negative examples as possible, and again, we want the volume of the hypersphere to be as small as possible. Let the training set be $\{(\mathbf{x}_i,y_i), i=1,2,\ldots,n\}$, where $y_i \in \{+1,-1\}$, with $y_i=+1$ for positive examples and $y_i=-1$ for negative examples. As in Section 2.1, we denote the radius of the hypersphere as $R$ and denote its center as $\mathbf{c}$.

Suppose we impose different penalties for misclassifying positive and negative examples, then similar to Section 2.1, the optimization problem could be summarized as

$$\min_{R,\mathbf{c},\xi} \quad F(R,\mathbf{c},\boldsymbol{\xi}) = R^2 + C_{+1}\sum_{i:y_i=+1}\xi_i + C_{-1}\sum_{i:y_i=-1}\xi_i = R^2 + \sum_{i=1}^{n}C_{y_i}\xi_i, \tag{7}$$

where $C_{+1}$ and $C_{-1}$ are the penalties on mistakenly classifying a positive or negative example, respectively. As in Section 2.1, $\xi_i$ is the slack variable on the $i$th example, and it should satisfy the constraints

$$\|\Phi(\mathbf{x}_i)-\mathbf{c}\|^2 \le R^2+\xi_i, \quad \xi_i \ge 0 \ \text{ for } y_i=+1, \tag{8}$$

and

$$\|\Phi(\mathbf{x}_i)-\mathbf{c}\|^2 \ge R^2-\xi_i, \quad \xi_i \ge 0 \ \text{ for } y_i=-1. \tag{9}$$

We compactly rewrite the constraints in one equation as

$$y_i\left(\|\Phi(\mathbf{x}_i)-\mathbf{c}\|^2 - R^2\right) \le \xi_i, \quad \xi_i \ge 0 \ \text{ for } i=1,\ldots,n. \tag{10}$$

By using the Lagrange multiplier method, we get the dual problem as

$$\min_{\boldsymbol{\alpha}} \quad L(\boldsymbol{\alpha}) = \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j) - \sum_{i=1}^{n}\alpha_i y_i K(\mathbf{x}_i,\mathbf{x}_i), \tag{11}$$

with constraints

$$\sum_{i=1}^{n}\alpha_i y_i = 1, \quad 0 \le \alpha_i \le C_{y_i} \ \text{ for } i=1,\ldots,n. \tag{12}$$

By the KKT condition, the center of the hypersphere can be represented as

$$\mathbf{c} = \sum_{i=1}^{n}\alpha_i y_i \Phi(\mathbf{x}_i). \tag{13}$$

Once the parameters $\alpha_i$'s are obtained, the radius $R$ can be computed from the set of support vectors.

Given a new example $\mathbf{x}$, the decision rule is

$$f(\mathbf{x}) = \operatorname{sign}\left(R^2 - \|\Phi(\mathbf{x}) - \sum_{i=1}^{n}\alpha_i y_i\Phi(\mathbf{x}_i)\|^2\right) = \operatorname{sign}\left(2\sum_{i=1}^{n}\alpha_i y_i K(\mathbf{x}_i,\mathbf{x}) - K(\mathbf{x},\mathbf{x}) + b\right), \tag{14}$$

where $b = R^2 - \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j)$.

We should notice that minimizing the objective function in Eq. (7) does not imply strong generalization ability of the resultant

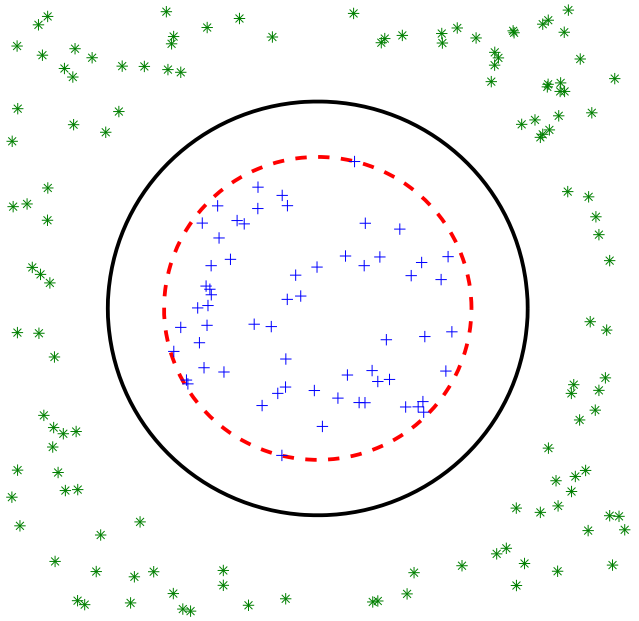

**Fig. 1.** Minimizing Eq. (7) does not imply good generalization ability: the red dashed circle minimizes Eq. (7) while the black solid circle has better generalization ability as a classifier. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

classifier because the radius $R$ is not directly related to the generalization ability. Particularly, 2-class SVDD is different from SVM in which $\|\mathbf{w}\|$ is related to the margin of the classifier which in turn measures the generalization ability. For instance, in Fig. 1, the blue "+" represents positive example and the green "*" represents negative example. The dashed red circle obviously corresponds to the global minimum solution for Eq. (7), but the black circle clearly has better generalization ability even though it does not achieve the smallest objective function defined in Eq. (7). To the best of our knowledge, the relationship between the generalization ability of SVDD and the related optimization problem has not been discussed in literature so far.

As implemented in popular toolboxes [8,12,16], the quadratic programming problems in Eqs. (3) and (11) can be solved by the decomposition methods [25,26] or sequential minimal optimization [27]. Theoretical and empirical studies [5] show that the computational cost for SVM QP problem is between $O(n^2)$ and $O(n^3)$, and since the SVDD QP problem is similar to the SVM QP problem, we expect SVDD to have a similar computational complexity for training. The numerical experiments in Section 4.2 on real world data show that the time complexity for QP based SVDD is approximately $O(n^3)$, at least on the tested datasets.

Thus, as the training set gets large, it is very expensive to train an SVDD model. In model selection, cross validation [14, chapter 7] is often used, in which we need to run the training algorithm multiple times to select the best set of parameters. Also, for feature selection, some well-known methods are designed to run the training algorithm many times, for example, the recursive feature elimination SVM [13,41]. In these situations, the reduction of training time is important for system development and progress making. As such, fast training algorithms for SVDD which can achieve similar accuracy as the quadratic programming method are highly appreciated.

## 3. The proposed approach

This section proposes a smooth approximation to the primal objective functions of SVDD models and presents a conjugate gradient algorithm to minimize the smoothed objective functions. We also provide a brief computational cost analysis for the proposed algorithms.

### 3.1. Smooth approximation to the SVDD models

In the SVDD formulation in Eq. (1), we rewrite the slack variable $\xi_i$ as

$$\xi_i = [\|\Phi(\mathbf{x}_i)-\mathbf{c}\|^2 - R^2]_+ = \max(0, \|\Phi(\mathbf{x}_i)-\mathbf{c}\|^2 - R^2), \tag{15}$$

where the function $[t]_+ = \max(0,t)$ is called hinge loss function [14, chapter 12]. Therefore, we reformulate the optimization problem for SVDD as

$$\min_{R,\mathbf{c}} \quad F(R,\mathbf{c}) = R^2 + C\sum_{i=1}^{n}\left[\|\Phi(\mathbf{x}_i)-\mathbf{c}\|^2 - R^2\right]_+, \tag{16}$$

which absorbs the constraints.

In the minimization problem in Eq. (16), the hinge loss function is not differentiable which prevents us from using gradient based optimization methods which usually converge fast to at least a local optimum point. Chen and Mangasarian [10] introduced a class of smooth functions for nonlinear optimization problems and the idea was applied to fitting SVM classifier [21,40] and quantile regression model [42] in statistics. Enlightened by these successes, we use a differentiable function

$$S_\tau(t) = \frac{1}{\tau}\log\left(1+e^{\tau t}\right) \tag{17}$$

to approximate the hinge loss function $[t]_+$, where $\tau$ is a large positive constant and is called the smoothing parameter. It can be verified that the function $S_\tau(t)$ is continuous, convex, and always dominates the hinge loss function. As the smoothing parameter $\tau$ increases, $S_\tau(t)$ converges uniformly to the hinge loss function. The verification is similar to that in [21,40], thus is omitted.

Replacing the hinge loss function in Eq. (16) by its smoothed counterpart (with $\tau$ fixed at a large positive value), we obtain the smoothed objective function as

$$F_\tau(R,\mathbf{c}) = R^2 + C\sum_{i=1}^{n} S_\tau\left(\|\Phi(\mathbf{x}_i)-\mathbf{c}\|^2 - R^2\right). \tag{18}$$

Obviously, from the properties of $S_\tau(\cdot)$, $F_\tau(R,\mathbf{c})$ is an upper bound of the function $F(R,\mathbf{c})$ defined in Eq. (16), and converges uniformly to $F(R,\mathbf{c})$ as $\tau$ increases.

By using the "kernel trick", we assume

$$\mathbf{c} = \sum_{i=1}^{n}\alpha_i\Phi(\mathbf{x}_i), \tag{19}$$

then the objective function in Eq. (18) becomes

$$\begin{aligned} F_\tau(R,\boldsymbol{\alpha}) &= R^2 + C\sum_{i=1}^{n} S_\tau\left(K(\mathbf{x}_i,\mathbf{x}_i) - 2\sum_{j=1}^{n}\alpha_j K(\mathbf{x}_i,\mathbf{x}_j) \right.\\ &\qquad \left. + \sum_{j=1}^{n}\sum_{k=1}^{n}\alpha_j\alpha_k K(\mathbf{x}_j,\mathbf{x}_k) - R^2\right)\\ &= R^2 + C\sum_{i=1}^{n} S_\tau\left(\mathbf{K}_{ii} - 2\mathbf{K}_{i\cdot}\boldsymbol{\alpha} + \boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha} - R^2\right), \end{aligned} \tag{20}$$

where $\mathbf{K}$ is the $n\times n$ kernel matrix with elements $\mathbf{K}_{ij} = K(\mathbf{x}_i,\mathbf{x}_j)$, $\mathbf{K}_{i\cdot}$ is the $i$th row of the kernel matrix $\mathbf{K}$. It is straightforward to show that the function $F_\tau(R,\boldsymbol{\alpha})$ is convex in $R^2$ and $\boldsymbol{\alpha}$, thus gradient method can give us a global minimum point of $F_\tau(R,\boldsymbol{\alpha})$.

Viewing $R^2$ as a single variable, we have

$$\frac{\partial F_\tau}{\partial R^2} = 1 - C\sum_{i=1}^{n} S'_\tau\left(\mathbf{K}_{ii} - 2\mathbf{K}_{i\cdot}\boldsymbol{\alpha} + \boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha} - R^2\right), \tag{21}$$

where

$$S'_\tau(t)=\frac{e^{\tau t}}{1+e^{\tau t}} \tag{22}$$

is calculated from the definition of $S_\tau(t)$ in Eq. (17). Taking derivative with respect to $\boldsymbol{\alpha}$, yields

$$\frac{\partial F_\tau}{\partial \boldsymbol{\alpha}}=2C\sum_{i=1}^{n}S'_\tau\left(\mathbf{K}_{ii}-2\mathbf{K}_{i\cdot}\boldsymbol{\alpha}+\boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha}-R^2\right)(\mathbf{K}\boldsymbol{\alpha}-\mathbf{K}_{\cdot i}), \tag{23}$$

where $\mathbf{K}_{\cdot i}$ is the $i$th column of the kernel matrix $\mathbf{K}$.

For the 2-class SVDD model, similarly, we introduce the hinge loss function to Eq. (7) to absorb the constraints in Eq. (10), yielding

$$F(R,\boldsymbol{\alpha})=R^2+\sum_{i=1}^{n}C_{y_i}\xi_i=R^2+\sum_{i=1}^{n}C_{y_i}\left[y_i\left(\|\Phi(\mathbf{x}_i)-\mathbf{c}\|^2-R^2\right)\right]_+. \tag{24}$$

By using the "kernel trick", we assume

$$\mathbf{c}=\sum_{i=1}^{n}\alpha_i y_i\Phi(\mathbf{x}_i), \tag{25}$$

and we continue as

$$\begin{aligned}F(R,\boldsymbol{\alpha})&=R^2+\sum_{i=1}^{n}C_{y_i}\left[y_iK(\mathbf{x}_i,\mathbf{x}_i)-2y_i\sum_{j=1}^{n}\alpha_j y_jK(\mathbf{x}_i,\mathbf{x}_j)\right.\\&\qquad\left.+y_i\sum_{j=1}^{n}\sum_{k=1}^{n}\alpha_j y_j\alpha_k y_kK(\mathbf{x}_j,\mathbf{x}_k)-y_iR^2\right]_+\\&=R^2+\sum_{i=1}^{n}C_{y_i}\left[y_i\mathbf{K}_{ii}-2y_i\mathbf{K}_{i\cdot}(\boldsymbol{\alpha}.*\mathbf{y})+y_i(\boldsymbol{\alpha}.*\mathbf{y})'\mathbf{K}(\boldsymbol{\alpha}.*\mathbf{y})-y_iR^2\right]_+\\&=R^2+\sum_{i=1}^{n}C_{y_i}\left[y_i\mathbf{K}_{ii}-2y_i\mathbf{K}_{i\cdot}\boldsymbol{\alpha}_{\mathbf{y}}+y_i\boldsymbol{\alpha}'_{\mathbf{y}}\mathbf{K}\boldsymbol{\alpha}_{\mathbf{y}}-y_iR^2\right]_+,\end{aligned} \tag{26}$$

where $\boldsymbol{\alpha}.*\mathbf{y}=(\alpha_1y_1,\ldots,\alpha_ny_n)'$, that is, the vector formed by element-wise multiplying two vectors. Note that $(\boldsymbol{\alpha}.*\mathbf{y}).*\mathbf{y}=\boldsymbol{\alpha}$ since $y_i=\pm1$. Thus, it is convenient to view $\boldsymbol{\alpha}.*\mathbf{y}$ as one variable, and we introduce the notation $\boldsymbol{\alpha}_{\mathbf{y}}=\boldsymbol{\alpha}.*\mathbf{y}$.

Notice that although the hinge loss function $[t]_+$ is convex in $t$, the function $y_i\mathbf{K}_{ii}-2y_i\mathbf{K}_{i\cdot}\boldsymbol{\alpha}_{\mathbf{y}}+y_i\boldsymbol{\alpha}'_{\mathbf{y}}\mathbf{K}\boldsymbol{\alpha}_{\mathbf{y}}-y_iR^2$ is concave in $\boldsymbol{\alpha}_{\mathbf{y}}$ for $y_i=-1$ since the kernel matrix $\mathbf{K}$ is always positive definite, thus $[y_i\mathbf{K}_{ii}-2y_i\mathbf{K}_{i\cdot}\boldsymbol{\alpha}_{\mathbf{y}}+y_i\boldsymbol{\alpha}'_{\mathbf{y}}\mathbf{K}\boldsymbol{\alpha}_{\mathbf{y}}-y_iR^2]_+$ is not convex in $\boldsymbol{\alpha}_{\mathbf{y}}$ for $y_i=-1$. As a consequence, the objective function in Eq. (26) is not convex in $\boldsymbol{\alpha}_{\mathbf{y}}$.

Replacing the hinge loss function in Eq. (26) by its smoothed counterpart (with $\tau$ fixed at a large positive value), we obtain the smoothed objective function as

$$F_\tau(R,\boldsymbol{\alpha}_{\mathbf{y}})=R^2+\sum_{i=1}^{n}C_{y_i}S_\tau\left(y_i\mathbf{K}_{ii}-2y_i\mathbf{K}_{i\cdot}\boldsymbol{\alpha}_{\mathbf{y}}+y_i\boldsymbol{\alpha}'_{\mathbf{y}}\mathbf{K}\boldsymbol{\alpha}_{\mathbf{y}}-y_iR^2\right). \tag{27}$$

The function $F_\tau(R,\boldsymbol{\alpha}_{\mathbf{y}})$ is not convex in $\boldsymbol{\alpha}_{\mathbf{y}}$, based on the same reason as in last paragraph. As such, the gradient based method can only yield a local minimum point of $F_\tau(R,\boldsymbol{\alpha}_{\mathbf{y}})$.

It is straightforward to calculate that

$$\frac{\partial F_\tau}{\partial R^2}=1-\sum_{i=1}^{n}y_iC_{y_i}S'_\tau\left(y_i\mathbf{K}_{ii}-2y_i\mathbf{K}_{i\cdot}\boldsymbol{\alpha}_{\mathbf{y}}+y_i\boldsymbol{\alpha}'_{\mathbf{y}}\mathbf{K}\boldsymbol{\alpha}_{\mathbf{y}}-y_iR^2\right), \tag{28}$$

and

$$\frac{\partial F_\tau}{\partial \boldsymbol{\alpha}_{\mathbf{y}}}=2\sum_{i=1}^{n}y_iC_{y_i}S'_\tau\left(y_i\mathbf{K}_{ii}-2y_i\mathbf{K}_{i\cdot}\boldsymbol{\alpha}_{\mathbf{y}}+y_i\boldsymbol{\alpha}'_{\mathbf{y}}\mathbf{K}\boldsymbol{\alpha}_{\mathbf{y}}-y_iR^2\right)(\mathbf{K}\boldsymbol{\alpha}_{\mathbf{y}}-\mathbf{K}_{\cdot i}). \tag{29}$$

### 3.2. Conjugate gradient for minimizing the smoothed objective function

In applying the idea of smoothly approximating a nonsmooth objective function, some works [9,21] used the Newton method to minimize the smoothed objective function. However, the Newton method involves estimating and inverting the Hessian matrix, which is time consuming and prone to errors due to the finite precision of floating-point numbers, especially in high dimensional spaces. Compared to the Newton method, conjugate gradient method avoids using the second order derivative information and inverting a matrix, and it only has a simple formula to determine the new search direction. This simplicity makes the method very easy to implement, only slightly more complicated than steepest descent. Other advantages of the conjugate gradient method include its low memory requirements and its convergence speed. Refer to [4, chapter 1] for more details. Algorithm 1 gives the procedure of conjugate gradient method for minimizing a general function $f$ over $\mathbf{R}^n$.

**Algorithm 1.** Conjugate gradient method for minimizing a function $f$ over $\mathbf{R}^n$.

0. Initialization: choose a starting point $\mathbf{x}_0\in\mathbf{R}^n$, compute $\mathbf{g}_0=\nabla f(\mathbf{x}_0)$ and set $\mathbf{d}_0=-\mathbf{g}_0$. Set the iteration number $m$.
1. **for** $t=1$ to $m$ **do**:
2. If $\mathbf{g}_{t-1}=0$, terminate and return $\mathbf{x}_{t-1}$ as the minimum point of $f$.
3. Set $\mathbf{x}_t=\mathbf{x}_{t-1}+\gamma_t\mathbf{d}_{t-1}$, where $\gamma_t$ is the step-size at iteration $t$.
4. Compute $\mathbf{g}_t=\nabla f(\mathbf{x}_t)$, and set $\mathbf{d}_t=-\mathbf{g}_t+\delta_t\mathbf{d}_{t-1}$, with $\delta_t=\frac{\mathbf{g}'_t\mathbf{g}_t}{\mathbf{g}'_{t-1}\mathbf{g}_{t-1}}$.
5. **end for**
6. Return $\mathbf{x}_m$ as the minimum point of $f$.

In the 3rd step, the step-size $\gamma_t$ is chosen as

$$\gamma_t=\arg\min_{\gamma>0}f(\mathbf{x}_{t-1}+\gamma\mathbf{d}_{t-1}). \tag{30}$$

The minimization problem in Eq. (30) can be solved by backtracking line search algorithms [6, chapter 9]. We choose to use Armijo rule [1] for its simplicity, which is given in Algorithm 2 for completeness.

**Algorithm 2.** Armijo rule to determine the step-size.

0. Given the objective function $f(\mathbf{x})$, the current estimation $\mathbf{x}_c$, the current gradient vector $\nabla f(\mathbf{x}_c)$, and the search direction $\mathbf{d}$.
1. Initialize $\gamma=1$.
2. Calculate $D=f(\mathbf{x}_c+\gamma\mathbf{d})-f(\mathbf{x}_c)$.
3. If $D\le\frac{\gamma}{4}\nabla f(\mathbf{x}_c)'\mathbf{d}$, return the current $\gamma$ as the step-size; otherwise, set $\gamma=\gamma/2$, and go back to step 2.

When applying the conjugate gradient method to minimize the objective function in Eq. (20) or (27), according to the properties of the smooth approximation, we should set a large value for $\tau$. However, as studied in [40], if we start with a very large $\tau$, the algorithm will be unstable. Starting from a relatively small $\tau$ and increasing it gradually will stabilize the solution. In practice, we do not increase $\tau$ after each conjugate gradient step, because we should let the conjugate gradient algorithm run several iterations to fully utilize its power in minimizing the objective function at the current $\tau$ value. As such, we adopt a sequential minimization strategy for fitting the smoothly approximated SVDD (SA-SVDD) model. Algorithm 3 sketches the procedure for minimizing the

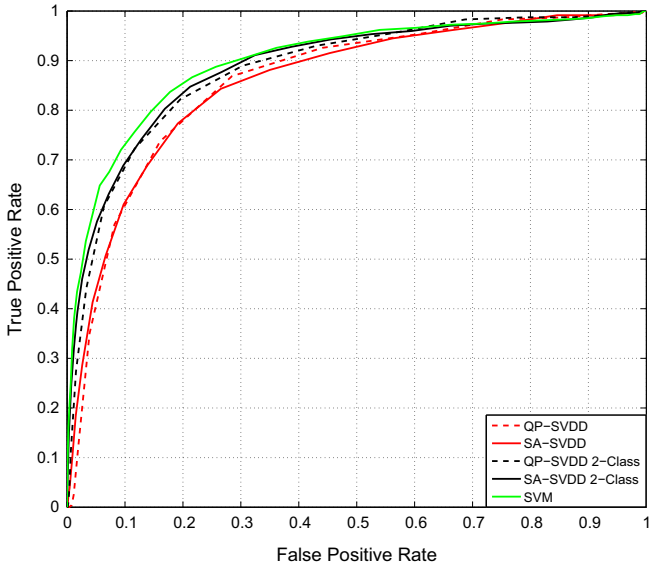


**Fig. 2.** On the CBCL face dataset, the ROC curves of QP-SVDD, SA-SVDD, 2-class QP-SVDD, 2-class SA-SVDD, and SVM.

objective function in Eq. (20), and the algorithm for the 2-class SA-SVDD could be developed similarly, thus is omitted.[1]

**Algorithm 3.** Conjugate gradient for smoothly approximated SVDD.

0. Initialize $R$ and $\boldsymbol{\alpha}$, set the maximum outer iteration number $M$, and the conjugate gradient iteration number $m$.
1. **for** $i=1$ to $M$ **do**:
2. Set $\tau = i$.
3. Minimize $F_\tau(R, \boldsymbol{\alpha})$ with $m$ iterations of conjugate gradient algorithm, with the currently estimated $R^2$ and $\boldsymbol{\alpha}$ as the initial point.
4. **end for**

To get a reasonably good model, from our experience, for the SA-SVDD with only positive examples, we set the outer iteration number $M$ to be 15 and set the inner iteration number $m$ to be 4; for the 2-class SA-SVDD, we set $M$ and $m$ to be 30 and 10, respectively. Thus, the SA-SVDD algorithm with only positive examples will run totally 60 steps of conjugate gradient at most, and the 2-class SA-SVDD will run at most 300 steps.[2] In our experiments, we tried different combinations of $M$ and $m$, and found that the performances of the resultant classifiers are similar. In general, setting the parameters $M$ and $m$ to be larger values will make the algorithm more precise, but it would spend more time as well.

Since we aim to enclose the positive examples using a hypersphere as much as possible, we initialize the diameter of the hypersphere as the largest distance between positive examples. In the feature space, the squared distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ is

$$d^2 = \|\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)\|^2 = K(\mathbf{x}_i, \mathbf{x}_i) - 2K(\mathbf{x}_i, \mathbf{x}_j) + K(\mathbf{x}_j, \mathbf{x}_j) = \mathbf{K}_{ii} - 2\mathbf{K}_{ij} + \mathbf{K}_{jj}. \quad (31)$$

[1] Algorithm 3 is similar to an algorithm in [40], where the convergence and stability were studied in detail. Thus, the convergence and stability of Algorithm 3 are not presented here.

[2] The objective function of the 2-class SA-SVDD (i.e., Eq. (27)) is not convex, as analyzed in Section 3.1. Hence, we let the 2-class SA-SVDD algorithm run a longer time, expecting it to find a good solution.

**Table 1**
On the CBCL face dataset, the area under the curve of different classifiers.

| QP-SVDD | SA-SVDD | 2-class QP-SVDD | 2-class SA-SVDD | 2-class SVM |
|---|---|---|---|---|
| 0.8583 | 0.8574 | 0.8830 | 0.8893 | 0.8996 |

Therefore, we initialize the variable $R^2$ as

$$R^2 = \max_{i,j}(\mathbf{K}_{ii} - 2\mathbf{K}_{ij} + \mathbf{K}_{jj})/4, \quad (32)$$

where the indices $i$ and $j$ run over all the positive examples. The vector $\boldsymbol{\alpha}$ or $\boldsymbol{\alpha}_y$ is initialized as zero vector.

### 3.3. Computational complexity

Since the proposed SA-SVDD algorithm (Algorithm 3) contains $M$ iterations of Algorithm 1, we need to analyze the complexity of Algorithm 1. In step 3 of Algorithm 1, there is a search for the optimal step-size implemented by Algorithm 2, and this search needs to evaluate the objective function expressed by Eq. (20) or (27), where the most computationally expensive operation is to calculate $\boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha}$ or $\boldsymbol{\alpha}_y'\mathbf{K}\boldsymbol{\alpha}_y$. Since the kernel matrix $\mathbf{K}$ is of size $n \times n$ and the coefficient vector $\boldsymbol{\alpha}$ or $\boldsymbol{\alpha}_y$ is of size $n \times 1$, the operation $\boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha}$ or $\boldsymbol{\alpha}_y'\mathbf{K}\boldsymbol{\alpha}_y$ has computational complexity $O(n^2)$. Note that although Eq. (20) or (27) has the form of summation with index from 1 to $n$, the operation $\boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha}$ or $\boldsymbol{\alpha}_y'\mathbf{K}\boldsymbol{\alpha}_y$ only needs to be done once for an iteration. From our experience, Algorithm 2 often converges in less than 10 iterations, thus the complexity of step 3 of Algorithm 1 is of the order $O(n^2)$.

In step 4 of Algorithm 1, we need to calculate the gradient vector of the objective function according to Eqs. (21), (23) or Eqs. (28), (29). We see that again, the most expensive operation is $\boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha}$ or $\boldsymbol{\alpha}_y'\mathbf{K}\boldsymbol{\alpha}_y$, which has complexity $O(n^2)$.

In our experiments, the whole algorithm (Algorithm 3) will run at most 60 steps of conjugate gradient for SVDD with only positive examples, and the 2-class SA-SVDD will run at most 300 steps. Thus, the computational complexity of the SA-SVDD algorithms is about $O(n^2)$. In Section 4.2, we will numerically verify that the computational complexity of the proposed algorithm is of the order $O(n^2)$.

## 4. Experimental results

On a face dataset and the USPS handwritten digit dataset, we compared the performances of the proposed smoothly approximated SVDD (SA-SVDD) and the ordinary quadratic programming based SVDD (QP-SVDD). The SA-SVDD was developed using MATLAB, and we did not do any specific code optimization.[3] The QP-SVDD was implemented based on the MATLAB SVM toolbox of [12] with the quadratic programming solver from the C++ version of LIBSVM [8]. All the experiments were conducted on a personal computer with Pentium IV CPU 3.00 GHz and 3.25 GB memory, with WinXP operating system and MATLAB® R2012b as the platform.

In our experiments, we adopted the Gaussian kernel

$$K(\mathbf{u}, \mathbf{v}) = \exp\left\{-\frac{\|\mathbf{u}-\mathbf{v}\|^2}{2\sigma^2}\right\} \quad (33)$$

with $\sigma = 5$. We set the SVDD parameters $C=2$, $C_{+1}=2$, and $C_{-1}=2$ in the algorithms. The parameter setting in our

[3] In our implementation of SA-SVDD, the conjugate gradient algorithm is a nested loop, and the inner loop calls a function for Armijo step size, which is developed by ourselves. Changing the inner loop to MATLAB native code will significantly improve the efficiency of SA-SVDD.

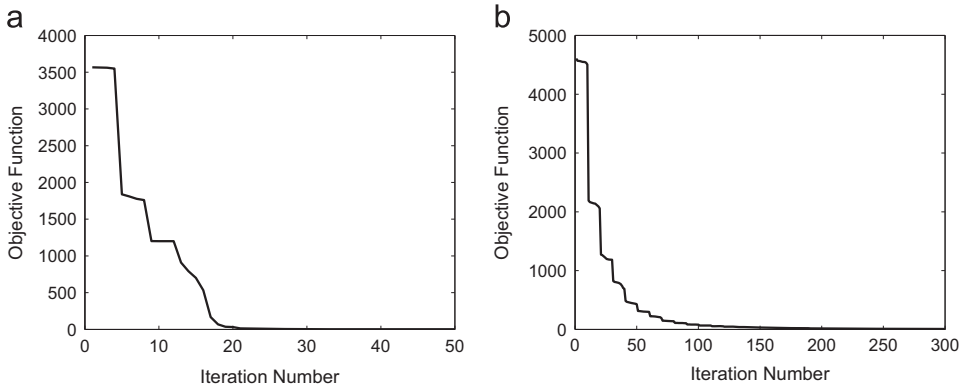


**Fig. 3.** The evolution curves of the objective functions for SA-SVDD algorithms on the CBCL face dataset: (a) SA-SVDD with only positive examples and (b) SA-SVDD with both positive and 1000 negative examples randomly selected.

experiments might not be the optimal to achieve the best testing error. Nevertheless, our purpose is not to achieve the optimal testing error, but to compare the performances of SA-SVDD and QP-SVDD; therefore, the comparison is fair as long as the parameter settings are the same for the two algorithms.

In the decision rules given by Eqs. (6) and (14), we change the value of the parameter $b$, and for each value of $b$, we calculate the true positive rate and false positive rate. We then plot the true positive rate vs. the false positive rate, resulting the receiver operating characteristic (ROC) curve [11], which will be used to illustrate the performance of the classifier. The classifier performs better if the corresponding ROC curve is higher. In order to numerically compare the ROC curves of different methods, we calculate the area under the curve (AUC) [11]. The bigger the AUC, the better the overall performance of the classifier.

### 4.1. Face detection

In the first experiment, we selected to use the face dataset provided by the Center for Biological & Computational Learning (CBCL) at MIT, which was downloaded from http://cbcl.mit.edu/software-datasets/FaceData2.html. The training set consists of 6977 images, with 2429 face images and 4548 non-face images; the testing set consists of 24 045 images, including 472 face images and 23 573 non-face images. Each image is of size $19 \times 19$ pixels, which is histogram equalized and normalized so that all pixel values are between 0 and 1. In the experiment, we did not extract any specific feature for face detection (e.g., the features used in [35]), instead, we used the pixel values as input to SA-SVDD or QP-SVDD directly.

We compare the performances of QP-SVDD and SA-SVDD by using all the 2429 face images as training set; to test the 2-class QP-SVDD and 2-class SA-SVDD, we randomly select 1000 non-face images combined with the 2429 face images as the training set. Fig. 2 shows the ROC curves of the QP-SVDD and SA-SVDD, for both versions; as a comparison to the 2-class SVDD models, the ROC curve of 2-class SVM is also given. The ROC curves clearly demonstrate that, for both versions, the performances of QP-SVDD and SA-SVDD are very close. The curves also show that with extra information from negative examples, the 2-class SVDD model performs better than the SVDD with only positive examples. We notice that the 2-class SVM has better performance than all the SVDD methods, which is not a surprise since SVM is designed to maximize the generalization ability (i.e., the margin of the classifier), while SVDD does not have such a mechanism.[4] Table 1 presents the AUC for different methods on the face detection problem, and the observation from Table 1 is consistent with our conclusion from Fig. 2.

**Table 2**
On the CBCL face dataset, the comparison between the QP-SVDD and SA-SVDD algorithms. The objective functions are calculated according to Eqs. (16) and (26) using the final solutions, the training times are in seconds.

| Method | QP-SVDD | SA-SVDD | 2-class QP-SVDD | 2-class SA-SVDD |
|---|---|---|---|---|
| Objective function | 0.6456 | 1.1888 | 0.6621 | 2.3047 |
| Training Time | 21,439 | 664.03 | 20,895 | 3389.7 |

Fig. 3 presents the evolution curves of the objective functions of SA-SVDD and the 2-class SA-SVDD, evaluated according to Eqs. (20) and (27), respectively. It is evident that the objective functions decrease monotonically as the algorithm proceeds, and the SA-SVDD algorithm converges within about 30 iterations, while the 2-class SA-SVDD converges in about 200 iterations. The minimum values of the objective functions are 1.2155 and 6.8739 for SA-SVDD and 2-class SA-SVDD, respectively.

Table 2 gives the objective function values[5] of QP-SVDD and SA-SVDD algorithms for both versions calculated using Eqs. (16) and (26), respectively, at the final solutions. We observe that at the optimal solution of SA-SVDD, the objective functions calculated by Eqs. (20) and (27) are higher than those calculated by Eqs. (16) and (26), which is expected because the smoothed objective functions in Eqs. (20) and (27) are upper bounds of the original objective functions in Eqs. (16) and (26). Table 2 shows that compared to QP-SVDD, the SA-SVDD solutions yield higher objective function values. We also notice that the objective function gap between the 2-class versions is higher than that of the 1-class versions, this is because the 2-class SA-SVDD objective function is not convex and the conjugate gradient method might have found a local minimum point. Nevertheless, since the final purpose is to perform classification, minimizing the objective function too much might overfit the classifier.

Table 2 also gives the training time for different models. For this problem, due to the large training set and expensive QP solver, the QP-SVDD algorithms spend very long time (about 6 h) in the training stage. The proposed SA-SVDD algorithm needs only 11 min to converge and the 2-class SA-SVDD needs less than 1 h to converge. Thus, we can conclude that compared to QP-SVDD, the proposed SA-SVDD algorithms converge much faster with similar classification performance on the CBCL face detection dataset.

[4] In the 2-class SVDD formulation, the radius $R$ of the hypersphere is not related to the generalization ability. Refer to Fig. 1.

[5] When plotting the ROC curves, we changed the values of $b$; when calculating the objective functions, we used the optimal solution.

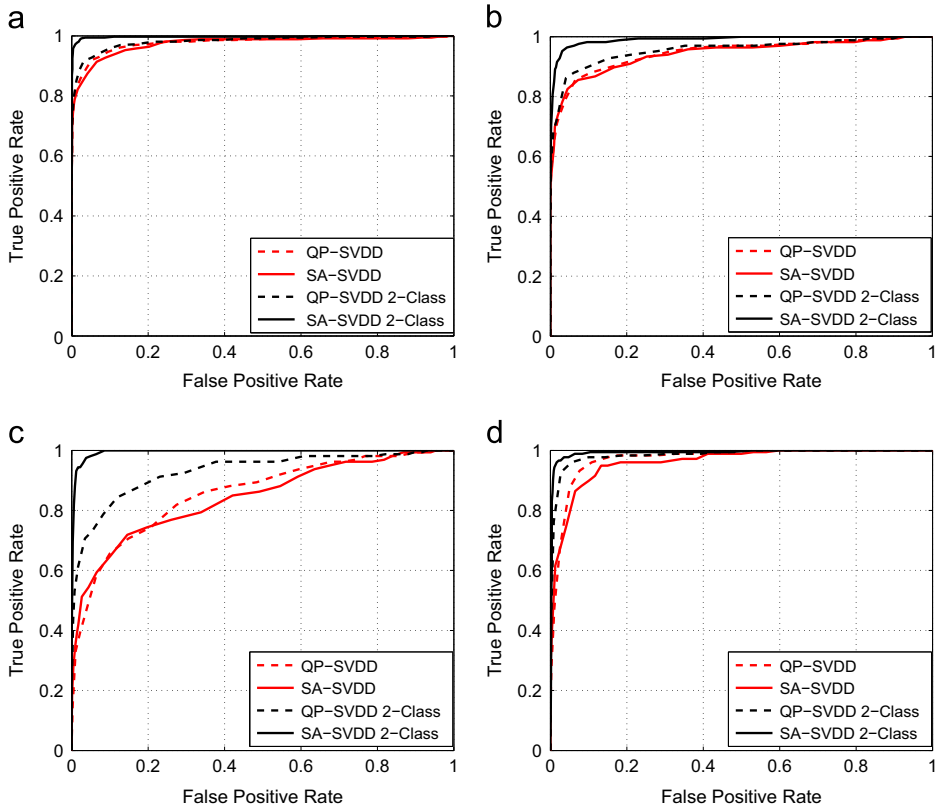


**Fig. 4.** The ROC curves of the considered SVDD algorithms on the USPS dataset with different digit as positive class: (a) digit "0" as positive; (b) digit "3" as positive; (c) digit "5" as positive; and (d) digit "9" as positive.

**Table 3**
The area under the curve of the considered classifiers on the USPS dataset.

| Digit | QP-SVDD | SA-SVDD | 2-class QP-SVDD | 2-class SA-SVDD |
|---|---|---|---|---|
| 0 | 0.9775 | 0.9734 | 0.9812 | 0.9965 |
| 3 | 0.9456 | 0.9426 | 0.9551 | 0.9901 |
| 5 | 0.8554 | 0.8423 | 0.9293 | 0.9949 |
| 9 | 0.9707 | 0.9603 | 0.9843 | 0.9922 |

### 4.2. Handwritten digit recognition

The task of recognizing handwritten digits captured the attention of pattern recognition community for a long time and has still been a benchmark problem. The dataset consists of normalized handwritten digits (0–9), automatically scanned from envelopes by the U.S. Postal Service. The originally scanned digits are binary with different sizes and orientations; then the images have been deslanted and size normalized, resulting in $16 \times 16$ gray-scale images [17]. Similar to the experiment in Section 4.1, we simply use these 256 pixel values as inputs to the algorithms. The dataset consists of 7291 training examples and 2007 testing examples, and is available at http://www-stat.stanford.edu/~tibs/ElemStatLearn.

To test the SVDD algorithms, we created four binary classification problems, with digits "0", "3", "5", and "9" as positive class, respectively. To test the performance of the 2-class SVDD algorithms, we randomly selected 400 negative examples for each problem. Refer to Table 5 for the training set size of each problem.

For each of the four binary classification problems, we present the ROC curves of QP-SVDD and SA-SVDD algorithms in Fig. 4. We observe that with only positive examples, the SA-SVDD performs slightly worse than the QP-SVDD, but in general, SA-SVDD benefits more from the negative examples than QP-SVDD does. As we have analyzed, 2-class QP-SVDD could find a global optimal solution because the objective function in Eq. (11) is convex, while the 2-class SA-SVDD can only get a local optimum point due to the non-convexity of the objective function in Eq. (27). However, in the 2-class SVDD situation, the global optimal solution does not necessarily bring good generalization ability, as illustrated by Fig. 1. This might be the reason why 2-class SA-SVDD performs better than 2-class QP-SVDD on this dataset. The 2-class SVM performs very similarly to the 2-class SA-SVDD, for visual effect, the ROC curve for SVM is not plotted. As a numerical measure, Table 3 shows the AUC of every classifier for each dataset, and the results are consistent with the observation from the curves in Fig. 4.

Figs. 5 and 6 show the evolution curves of the objective functions of SA-SVDD and 2-class SA-SVDD as the algorithms proceed, respectively. As we have observed on the CBCL dataset, the objective functions decrease monotonically, and SA-SVDD converges in about 30 iterations, while the 2-class SA-SVDD converges in about 200 iterations. In Figs. 5 and 6, we also list the final objective function values. Table 4 gives the objective function values of the QP-SVDD and SA-SVDD algorithms calculated using Eqs. (16) and (26), respectively, at the final solutions. Similar to the results on the CBCL dataset, we observe that the SA-SVDD algorithms yield higher objective function values than QP-SVDD algorithms.

Table 4 also lists the training times of different algorithms on the USPS dataset. To clearly illustrate the speed advantage of SA-SVDD, we calculate the ratio between the training times of QP-SVDD and SA-SVDD for different problems, and list the results in Table 5, which also gives the sizes of the considered problems. It is clearly seen that with only positive examples, the SA-SVDD algorithm is 20–30 times faster than the QP counterpart, and the 2-class SA-SVDD is 3–6 times faster than the 2-class QP-SVDD. We also notice that the speed advantage of the proposed algorithm is more significant for larger datasets. More importantly, we should

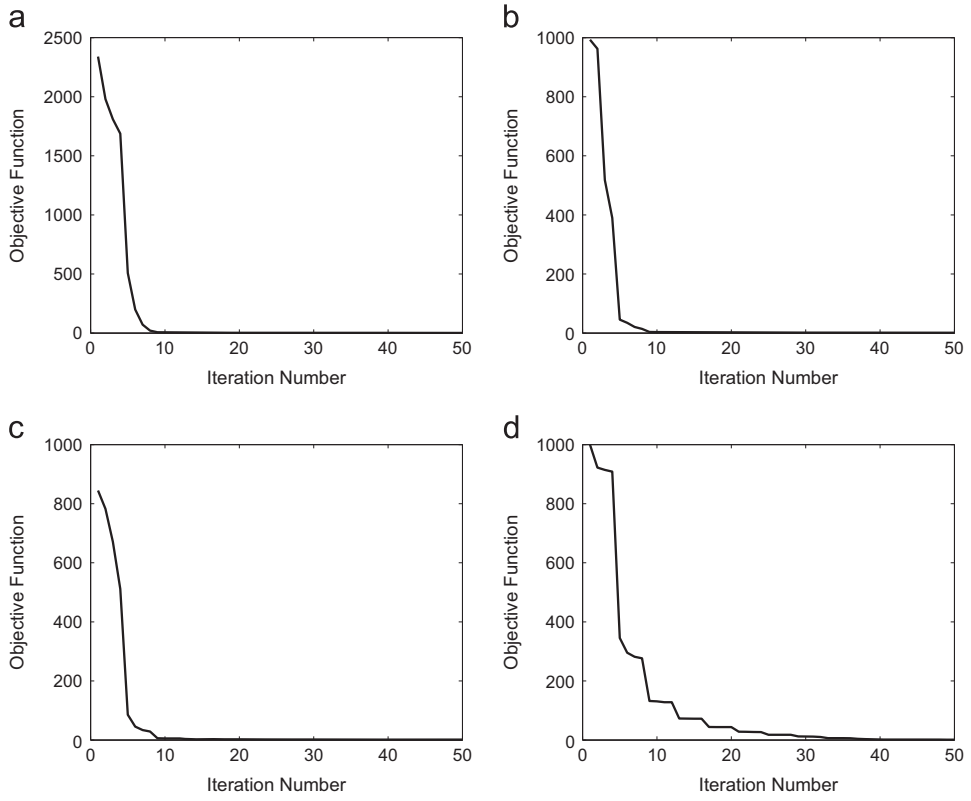


**Fig. 5.** The objective functions for SA-SVDD algorithm with only positive examples on the USPS dataset: (a) digit "0" as positive, with the minimum value 1.2737; (b) digit "3" as positive, with the minimum value 1.1995; (c) digit "5" as positive, with the minimum value 1.2337; and (d) digit "9" as positive, with the minimum value 1.1451.

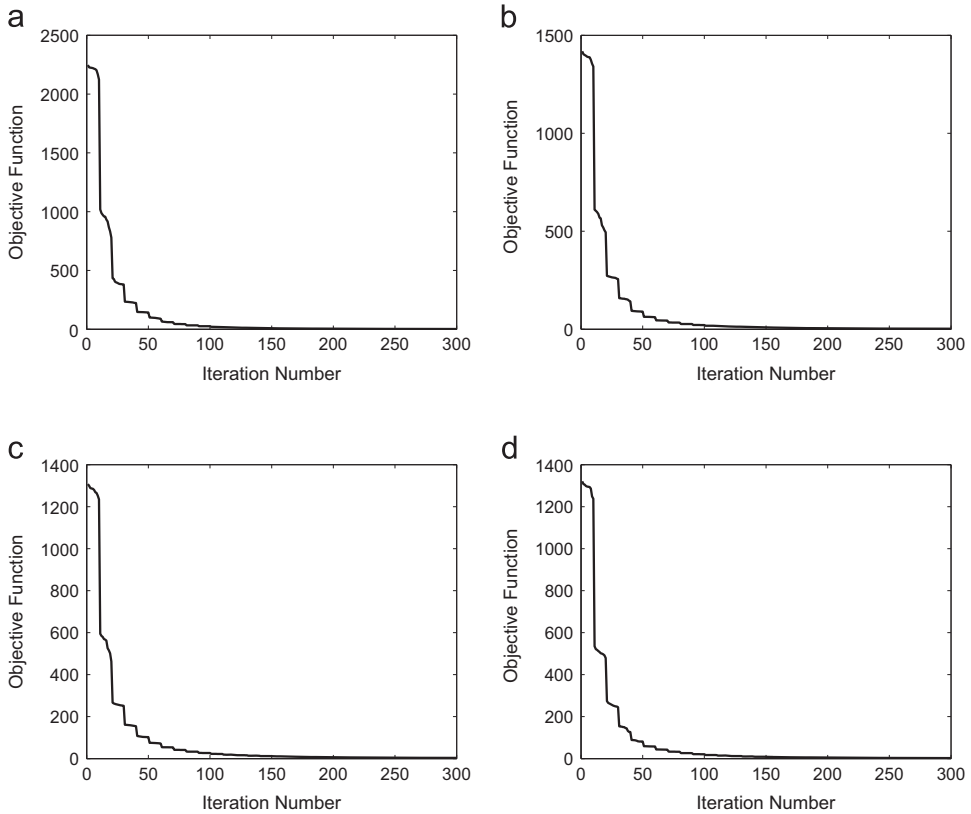


**Fig. 6.** The objective functions for SA-SVDD algorithm with positive examples and 400 randomly selected negative examples on the USPS dataset: (a) digit "0" as positive, with the minimum value 2.2640; (b) digit "3" as positive, with the minimum value 2.5702; (c) digit "5" as positive, with the minimum value 3.2738; and (d) digit "9" as positive, with the minimum value 2.7128.

**Table 4**
On the USPS dataset, the comparison between the QP-SVDD and SA-SVDD algorithms. The objective function are calculated according to Eqs. (16) and (26) using the final solutions, the training times are in seconds.

| Digit | Method | QP-SVDD | SA-SVDD | 2-class QP-SVDD | 2-class SA-SVDD |
|---|---|---|---|---|---|
| 0 | Objective function | 0.7588 | 1.2377 | 0.7600 | 1.5057 |
| | Training time | 2417.6 | 80.7813 | 1915.1 | 428.9688 |
| 3 | Objective function | 0.7310 | 1.1600 | 0.7340 | 1.6370 |
| | Training time | 414.8281 | 18.3594 | 559.0938 | 153.4375 |
| 5 | Objective function | 0.7566 | 1.1989 | 0.7634 | 2.0453 |
| | Training time | 252.6719 | 12.0313 | 408.5156 | 117.1406 |
| 9 | Objective function | 0.7029 | 1.0935 | 0.7051 | 1.8785 |
| | Training time | 392.5469 | 20.0156 | 554.1250 | 147.2500 |

**Table 5**
The ratio of training times of QP-SVDD and SA-SVDD on different problems. The problem sizes are also given.

| Problem | Digit 5 | Digit 9 | Digit 3 | Digit 0 | CBCL |
|---|---|---|---|---|---|
| Positive # | 556 | 644 | 658 | 1194 | 2429 |
| SVDD time ratio | 21.00 | 19.61 | 22.60 | 29.93 | 32.29 |
| Training # | 956 | 1044 | 1058 | 1594 | 3429 |
| 2-class SVDD time ratio | 3.49 | 3.76 | 3.64 | 4.46 | 6.16 |

**Table 6**
The time complexity of different algorithms on the USPS dataset.

| | | | |
|---|---|---|---|
| QP-SVDD | $(n_9/n_5)^3 = 1.5339$<br>$t_9/t_5 = 1.5536$ | $(n_3/n_5)^3 = 1.6575$<br>$t_3/t_5 = 1.6418$ | $(n_9/n_5)^3 = 9.9035$<br>$t_9/t_5 = 9.5681$ |
| SA-SVDD | $(n_9/n_5)^2 = 1.3416$<br>$t_9/t_5 = 1.6636$ | $(n_3/n_5)^2 = 1.4006$<br>$t_3/t_5 = 1.5260$ | $(n_9/n_5)^2 = 4.6117$<br>$t_9/t_5 = 6.7143$ |
| 2-class QP-SVDD | $(n_9/n_5)^3 = 1.3024$<br>$t_9/t_5 = 1.3564$ | $(n_3/n_5)^3 = 1.3554$<br>$t_3/t_5 = 1.3686$ | $(n_9/n_5)^3 = 4.6354$<br>$t_9/t_5 = 4.6879$ |
| 2-class SA-SVDD | $(n_9/n_5)^2 = 1.1926$<br>$t_9/t_5 = 1.2570$ | $(n_3/n_5)^2 = 1.2248$<br>$t_3/t_5 = 1.3099$ | $(n_9/n_5)^2 = 2.7801$<br>$t_9/t_5 = 3.6620$ |

mention that in our implementation, the core quadratic programming code for QP-SVDD was developed in C++ which is far more computationally efficient than our SA-SVDD implementation. Taking this factor into account, if the implementation of SA-SVDD is optimized, it would be much more time-efficient than QP-SVDD.

To gain further insight about the computational complexity and to verify our theoretical analysis in Section 3.3, we compared the training time on the digit results. Since the problem for digit "5" has the smallest size, we used it as the baseline. For each algorithm, we calculated the ratio of the training time on the digits "9", "3" and "0" to that on the digit "5", along with the square and cubic of the problem size ratio. Table 6 presents the results.

It is observed from Table 6 that the training time of QP-SVDD algorithms grows perfectly in the rate of $O(n^3)$. For the proposed SA-SVDD algorithms, the time complexity is close to $O(n^2)$. Because for different problems, the algorithm needs different number of iterations to converge, there is fluctuation in the training time ratio. This verifies the theoretical analysis in Section 3.3 that the time complexity of the proposed method is $O(n^2)$.

## 5. Conclusion and future works

Support vector domain description (SVDD) is a well-known tool for data description and pattern classification, with many applications. In literature, the SVDD model is often fitted by solving the dual of a constrained optimization problem, resulting in a quadratic programming problem, which is computationally expensive to solve, with time complexity about $O(n^3)$, where $n$ is the training set size. The high training cost is rather undesirable in model selection and some feature selection methods.

As an alternative, this paper attempts to fit the SVDD model by directly minimizing the primal form of the optimization problem. However, the primal objective function of SVDD is not differentiable, which makes the well-developed gradient based optimization methods inapplicable. These methods are desirable because they are easy to implement, and because they converge quickly to at least a local optimum. As such, we introduce a smooth approximation to the original primal objective function of the SVDD model, resulting in an unconstrained smooth optimization problem. Finally, the conjugate gradient method is applied to minimize the smoothed objective function. The proposed algorithm has computational complexity about $O(n^2)$.

Experiments were conducted on face detection and handwritten digit recognition problems, and we compared the performance of the proposed smoothly approximated SVDD (SA-SVDD) to that of the quadratic programming based SVDD (QP-SVDD), in terms of ROC analysis and training time. Our results show that the developed SA-SVDD has slightly worse classification performance than QP-SVDD if only positive examples are used for training; if the training set contains both positive and negative examples, the SA-SVDD has better classification performance. In our experiments, if there are only positive examples, SA-SVDD is 20–30 times faster than QP-SVDD; with both positive and negative examples, SA-SVDD is 3–6 times faster than QP-SVDD. We also observe that as the training set size gets larger, the speed advantage of SA-SVDD over QP-SVDD becomes greater.

As shown in [32], the SVDD model is equivalent to the model in [28], which separates the origin and the positive examples by a hyperplane. The model in [28] also uses the hinge loss function, yielding a quadratic programming problem. Thus, a possible extension to this work is to apply the idea of smooth approximation to the model in [28] and save computing time. Currently, we employ the conjugate gradient method for minimizing the smoothed objective function. It will be interesting to compare other gradient based optimization methods, for example, quasi-Newton method [4,36], coordinate gradient descent [7,15], or blockwise coordinate descent [22].

## Conflict of interest

None declared.

## Acknowledgment

The author would like to extend his sincere gratitude to the anonymous reviewers for their constructive suggestions and comments, which have greatly helped improve the quality of this paper. This work was supported by a Faculty Research Grant from Missouri State University. Part of this work was done while the author was on sabbatical leave.

## References


[1] L. Armijo, Minimization of functions having Lipschitz-continuous first partial derivatives, Pac. J. Math. 16 (1966) 1–3.
[2] A. Banerjee, C. Diehl, A support vector method for anomaly detection in hyperspectral imagery, IEEE Trans. Geosci. Remote Sens. 24 (2006) 2282–2291.
[3] A. Ben-Hur, D. Horn, H.T. Siegelmann, V. Vapnik, Support vector clustering, J. Mach. Learn. Res. 2 (2001) 125–137.
[4] D.P. Bertsekas, Nonlinear Programming, 2nd ed., Athena Scientific, Belmont, MA, 1999.
[5] A. Bordes, S. Ertekin, J. Weston, L. Bottou, Fast kernel classifiers with online and active learning, J. Mach. Learn. Res. 6 (2005) 1579–1619.
[6] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, New York, 2004.
[7] K.-W. Chang, C.-J. Hsieh, C.-J. Lin, Coordinate descent method for large-scale $L_2$-loss linear support vector machines, J. Mach. Learn. Res. 9 (2008) 1369–1398.
[8] C.C. Chang, C.J. Lin, LIBSVM: A library for support vector machines, ACM Trans. Intell. Syst. Technol. 2 (3) (2011) 27:1–27.
[9] O. Chapelle, Training a support vector machine in the primal, Neural Comput. 19 (2007) 1155–1178.
[10] C. Chen, O.L. Mangasarian, A class of smoothing functions for nonlinear and mixed complementarity problems, Comput. Optim. Appl. 5 (1996) 97–138.
[11] T. Fawcett, An introduction to ROC analysis, Pattern Recognit. Lett. 27 (8) (2006) 861–874.
[12] S.R. Gunn, Support Vector Machines for Classification and Regression, Technical Report, Image Speech and Intelligent Systems Research Group, University of Southampton, Available at: ⟨http://users.ecs.soton.ac.uk/srg/publications/pdf/SVM.pdf⟩, 1997.
[13] I. Guyon, J. Weston, S. Barnhill, V. Vapnik, Gene selection for cancer classification using support vector machines, Mach. Learn. 46 (1–3) (2002) 389–422.
[14] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning: Prediction, Inference and Data Mining, 2nd ed., Springer Verlag, New York, 2009.
[15] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S.S. Keerthi, S. Sundararajan, A dual coordinate descent method for large-scale linear SVM, in: Proceedings of the 25th International Conference on Machine Learning, 2008, pp. 408–415.
[16] J. Joachims, Making large-scale SVM learning practical, in: B. Schölkopf, C. Burges, A. Smola (Eds.), Advances in Kernel Methods – Support Vector Learning, MIT Press, Cambridge, Massachusetts, 1999.
[17] Y. Le Cun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, Handwritten digit recognition with a back-propagation network, in: Proceedings of Advances in Neural Information Processing Systems, 1990, pp. 396–404.
[18] D. Lee, J. Lee, Domain described support vector classifier for multi-classification problems, Pattern Recognit. 40 (1) (2007) 41–51.
[19] J. Lee, D. Lee, An improved cluster labeling method for support vector clustering, IEEE Trans. Pattern Anal. Mach. Intell. 27 (3) (2005) 461–464.
[20] J. Lee, D. Lee, Dynamic characterization of cluster structures for robust and inductive support vector clustering, IEEE Trans. Pattern Anal. Mach. Intell. 28 (11) (2006) 1869–1874.
[21] Y.-J. Lee, O.L. Mangasarian, SSVM: a smooth support vector machine for classification, Comput. Optim. Appl. 20 (October (1)) (2001) 5–22.
[22] H. Liu, M. Palatucci, J. Zhang, Blockwise coordinate descent procedures for the multi-task lasso, with applications to neural semantic basis discovery, in: Proceedings of the 26th International Conference on Machine Learning, 2009, pp. 649–656.
[23] J. Muñoz-Marí, L. Bruzzone, G. Camps-Valls, A support vector domain description approach to supervised classification of remote sensing images, IEEE Trans. Geosci. Remote Sens. 45 (8) (2007) 2683–2692.
[24] J. Muñoz-Marí, F. Bovolo, L. Gómez-Chova, L. Bruzzone, G. Camps-Valls, Semisupervised one-class support vector machines for classification of remote sensing data, IEEE Trans. Geosci. Remote Sens. 48 (8) (2010) 3188–3197.
[25] E. Osuna, R. Freund, F. Girosi, An improved training algorithm for support vector machines, in: Proceedings of IEEE Workshop Neural Networks for Signal Processing, 1997, pp. 276–285.
[26] E. Osuna, R. Freund, F. Girosi, Training support vector machines: an application to face detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1997.
[27] J. Platt, Fast training of support vector machines using sequential minimal optimization, in: B. Schölkopf, C. Burges, A. Smola (Eds.), Advances in Kernel Methods – Support Vector Learning, MIT-Press, Cambridge, Massachusetts, 1998.
[28] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, SV estimation of a distribution's support, in: Advances in Neural Information Processing Systems, 1999.
[29] R. Tang, J. Han, X. Zhang, Efficient iris segmentation method with support vector domain description, Opt. Appl. XXXIX (2) (2009) 365–374.
[30] D.M.J. Tax, R.P.W. Duin, Data domain description using support vectors, in: Proceedings of the European Symposium on Artificial Neural Networks, 1999, pp. 251–256.
[31] D.M.J. Tax, R.P.W. Duin, Support vector domain description, Pattern Recognit. Lett. 20 (11–13) (1999) 1191–1199.
[32] D.M.J. Tax, R.P.W. Duin, Support vector data description, Mach. Learn. 54 (1) (2004) 45–66.
[33] D.M.J. Tax, A. Ypma, R.P.W. Duin, Support vector data description applied to machine vibration analysis, in: M. Boassen, J. Kaandorp, J. Tonino, M.G. Vosselman (Eds.), Proceedings of the 5th Annual Conference of the ASCI, 1999, pp. 398–405.
[34] V. Vapnik, Statistical Learning Theory, John Wiley, New York, 1998.
[35] P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2001.
[36] G.R. Walsh, Methods of Optimization, John Wiley and Sons, New York, 1975.
[37] X. Wang, S. Lu, J. Zhai, Fast fuzzy multicategory SVM based on support vector domain description, Int. J. Pattern Recognit. Artif. Intell. (1) (2008) 109–120.
[38] A. Ypma, D.M.J. Tax, R.P.W. Duin, Robust machine fault detection with independent component analysis and support vector data description, in: Proceedings of IEEE Workshop on Neural Networks for Signal Processing, 1999.
[39] X. Yu, D. Dementhon, D. Doermann, Support vector data description for image categorization from Internet images, in: Proceedings of IEEE International Conference on Pattern Recognition, 2008.
[40] J. Zhang, R. Jin, Y. Yang, A.G. Hauptmann, Modified logistic regression: an approximation to SVM and its applications in large-scale text categorization, in: Proceedings of the 20th International Conference on Machine Learning, 2003, pp. 888–895.
[41] X. Zhang, X. Lu, Q. Shi, X. Xu, E. Hon-chiu, L. Harris, W. Wong, Recursive SVM feature selection and sample classification for mass-spectrometry and micro-array data, BMC Bioinform. 7 (1) (2006) 197.
[42] S. Zheng, Gradient descent algorithms for quantile regression with smooth approximation, Int. J. Mach. Learn. Cybern. 2 (3) (2011) 191–207.


**Songfeng Zheng** received his B.S. degree in Electrical Engineering and M.S. degree in Computer Science from Xi'an JiaoTong University, China, in 2000 and 2003, respectively. In 2008, he received Ph.D. degree in Statistics from the University of California, Los Angeles. From 2008 to 2014, he was an Assistant Professor with the Department of Mathematics, Missouri State University. He has been an Associate Professor with the same institute since 2014. His research interests include statistical learning, pattern recognition, statistical computation, and image analysis.