Neurocomputing 275 (2018) 2880–2891

Contents lists available at ScienceDirect

Neurocomputing

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

Sparse algorithm for robust LSSVM in primal space

Li Chen [a,b], Shuisheng Zhou [a,*]

[a] School of Mathematics and Statistics, Xidian University, 266 Xinglong Section, Xifeng Road, Xi'an, China
[b] Department of Basic Science, College of Information and Business, Zhongyuan Technology University, 41 Zhongyuan Middle Road, Zhengzhou, China

ARTICLE INFO

Article history:
Received 11 January 2017
Revised 28 August 2017
Accepted 25 October 2017
Available online 3 November 2017

Communicated by Prof. Zidong Wang

MSC:
68T10
65K10
90C06

Keywords:
Primal LSSVM
Sparse solution
Re-weighted LSSVM
Low-rank approximation
Outliers

ABSTRACT

As having the closed form solution, the least squares support vector machine (LSSVM) has been widely used for classification and regression problems owing to its competitive performance compared with other types of SVMs. However, the LSSVM has two drawbacks: it is sensitive to outliers and its solution lacks sparseness. The robust LSSVM (R-LSSVM) partially overcomes the first drawback via its nonconvex truncated loss function, but it is unable to address the second drawback because its current algorithms produce dense solutions that are inefficient for training large-scale problems. In this paper, we interpret the robustness of the R-LSSVM from a re-weighted viewpoint and develop a primal R-LSSVM using the representer theorem. The new model may have a sparse solution. Then, we design a convergent sparse R-LSSVM (SR-LSSVM) algorithm to achieve a sparse solution of the primal R-LSSVM after obtaining a low-rank approximation of the kernel matrix. The new algorithm not only overcomes the two drawbacks of LSSVM simultaneously, it also has lower complexity than the existing algorithms. Therefore, it is very efficient at training large-scale problems. Numerous experimental results demonstrate that the SR-LSSVM can achieve better or comparable performance to other related algorithms in less training time, especially when used to train large-scale problems.

1. Introduction

The least squares support vector machine (LSSVM) was introduced by Suykens and Vandewalle [1] and has been a powerful learning technique for classification and regression. It has been successfully used in many real-world pattern recognition problems, such as disease diagnosis [2], fault detection [3], image classification [4], partial differential equation solving [5], and visual tracking [6]. The LSSVM tries to minimize least squares errors on the training samples. The LSSVM is based on equality constraints rather than inequality constraints, unlike other SVMs; therefore, it produces closed-form solutions by solving a system of linear equations instead of adopting the conventional SVM method of iteratively solving a quadratic programming (QP) problem. Thus, the training of the LSSVM is simpler than that of other SVMs.

However, the LSSVM has two main disadvantages. One is that it is sensitive to outliers which always have large support values (the values of a Lagrange multiplier). It means that the influence of outliers is larger than other samples with respect to the construction of the decision function. The other disadvantage is that the solution of the LSSVM lacks sparseness (i.e., almost all the elements in the solution are nonzero), which limits the method when training large-scale problems.

In order to overcome the problem of outlier sensitivity, Suykens et al. [7] proposed a weighted LSSVM (W-LSSVM) model by distributing small weights to less important samples and outliers in order to reduce their influence on the model. Several other weight setting strategies were proposed as well; see [8,9]. Theoretical analyses and the experimental results indicate that such methods are highly effective at handling outliers. However, those methods must pre-solve the original LSSVM in order to set the weights, so they are all not suitable for training large-scale problems. Another technique utilizes non-convex loss functions to improve robustness. Non-convex loss gives a constant penalty for any outliers which have large penalty. For example, based on the non-convex truncated least squares loss function, Wang and Zhong [10] and Yang et al. [11] presented the robust LSSVM (R-LSSVM) model. Their experimental results show that the R-LSSVM significantly reduces the influence of outliers; however, the solutions to the R-LSSVM model achieved by the algorithms of Yang and Wang lack sparseness. In addition, they require the entire kernel matrix, $K \in \Re^{m \times m}$, to be computed beforehand, along with the inverse of $(\lambda I + K)$, where m is the number of training samples, $\lambda \in \Re$ is the regularization parameter, and $I \in \Re^{m \times m}$ is an identity matrix.

* Corresponding author.
E-mail addresses: lilichenhappy@163.com (L. Chen), sszhou@mail.xidian.edu.cn (S. Zhou).

https://doi.org/10.1016/j.neucom.2017.10.011
0925-2312/

Thus, they are both time-consuming for large-scale datasets. They are even unable to manage datasets containing more than 10,000 training samples on average computers.

In order to promote sparsity in the solution of the LSSVM, Suykens et al. [12,13] proposed a pruning algorithm that iteratively removes a small number of samples (5%) with the smallest support values to impose sparseness. In this pruning algorithm, a retraining of the LSSVM with the reduced training set is needed for each iteration, which leads to a large computation cost. The fixed-size least squares support vector machine (FS-LSSVM) [7] is another sparse algorithm. In this algorithm, some support vectors (SVs), referred to as prototype vectors, are fixed in advance, and then they are replaced iteratively by samples that are randomly selected from the training set based on the quadratic Rényi entropy criterion. However, in each iteration, this method only computes the entropy of the samples that are selected in the working set instead of the entire dataset. This may result in sub-optimized solutions. Jiao et al. [14] presented the fast sparse approximation for the LSSVM (FSA-LSSVM). This approach utilizes an approximated decision function that is constructed iteratively by adding the basis function from a kernel-based dictionary one by one until the ε criterion is satisfied. This algorithm obtains sparse classifiers at a rather low cost. However, by utilizing a very sparse setting, the experimental results in [15] show that FSA-LSSVM is not ideal when applied to several training datasets. Zhou [15] proposed the pivoting Cholesky of primal LSSVM (PCP-LSSVM), which is an iterative method based on the incomplete pivoting Cholesky factorization of the kernel matrix. Theoretical analyses and the experimental results indicate that the PCP-LSSVM can obtain acceptable test accuracy with an extremely sparse solution.

In this paper, we aim to obtain a sparse solution of the R-LSSVM model to overcome the two disadvantages of the LSSVM simultaneously. The new algorithm solves the R-LSSVM in primal space as Zhou [15] did for the LSSVM, and our main contributions are summarized as follows:

- By introducing an equivalent form of the truncated least squared loss function, we show that the R-LSSVM is equivalent to a re-weighted LSSVM model, which explains the robustness of the R-LSSVM.
- We illustrate that the representer theorem is also held for the non-convex loss function, and we propose the primal R-LSSVM model, which has a sparse solution if the kernel matrix is a low rank matrix.
- We propose the SR-LSSVM algorithm to obtain a sparse solution of the R-LSSVM by applying a low-rank approximation of the kernel matrix. The complexity of the new algorithm is lower than that of existing non-sparse R-LSSVM algorithms.
- A large number of experiments demonstrate that the proposed algorithm can process large-scale problems efficiently.

Recently, Zhang et al. proposed a least-squares-based, non-parametric method known as ProCRC (Probabilistic Collaborative Representation based Classifier) [16], which is an improved CRC (Collaborative Representation based classification) method [17]. The ProCRC uses the training samples directly to predict the class labels of unknown data. It has been successfully used in multi-class classification problems, particularly in face recognition [16–18]. In our experiments, we compared the performance of the ProCRC and our SR-LSSVM method in terms of accuracy and time.

The rest of this paper is organized as follows. Brief descriptions of the R-LSSVM and its existing algorithms are given in Section 2. In Section 3, the robustness of the R-LSSVM is interpreted from a re-weighted viewpoint. In Section 4, the primal R-LSSVM and its smoothed version are discussed, and the novel sparse algorithm is proposed. Subsequently, the convergence and complexity of the new algorithm are analyzed. Section 5 includes several experiments to demonstrate the efficiency of the proposed algorithm. Section 6 concludes this paper.

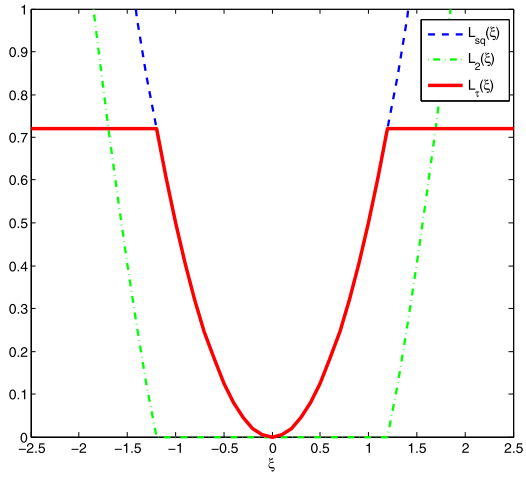

Fig. 1. Plots of the least squares loss $L_{sq}(\xi)$ (dashed), the truncated least squares loss $L_\tau(\xi)$ (solid) and their difference $L_2(\xi) = L_{sq}(\xi) - L_\tau(\xi)$ (dotted-dashed), where $\tau = 1.2$.

2. Robust LSSVM model and the existing algorithms

In this section, we briefly summarize the R-LSSVM and existing algorithms.

2.1. Robust LSSVM

Consider a training set with m pairs of samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$, where $\mathbf{x}_i \in \Re^l$ are the input data and $y_i \in \{-1, +1\}$ or $y_i \in \Re$ are the output targets corresponding to the inputs for classification or regression problems. The classical LSSVM model is described as follows:

$$\min_{\mathbf{w}\in\Re^m, b\in\Re} \frac{\lambda}{2}\mathbf{w}^\top\mathbf{w} + \frac{1}{m}\sum_{i=1}^{m} L_{sq}\left(y_i - \mathbf{w}^\top\varphi(\mathbf{x}_i) - b\right), \tag{1}$$

where $\lambda > 0$ is the regularization parameter, $\varphi(\mathbf{x})$ is a map that maps the input $\mathbf{x}$ into a high-dimensional feature space, especially for managing nonlinear learning problems, $L_{sq}(\xi) = \frac{1}{2}\xi^2$ is the least squares loss with $\xi = y - \mathbf{w}^\top\varphi(\mathbf{x}) - b$ being the predict error, $\mathbf{w}$ is the normal of the hyperplane, and b is the bias.

By replacing $L_{sq}(\xi)$ in (1) with the truncated least squares loss $L_\tau(\xi)$:

$$L_\tau(\xi) = \frac{1}{2}\min\left(\tau^2, \xi^2\right) = \begin{cases} \frac{1}{2}\xi^2, & \text{if } |\xi| \le \tau, \\ \frac{1}{2}\tau^2, & \text{if } |\xi| > \tau, \end{cases} \tag{2}$$

Wang and Zhong [10] and Yang et al. [11] introduced the R-LSSVM:

$$\min_{\mathbf{w}\in\Re^m, b\in\Re} \frac{\lambda}{2}\mathbf{w}^\top\mathbf{w} + \frac{1}{m}\sum_{i=1}^{m} L_{\tau}\left(y_i - \mathbf{w}^\top\varphi(\mathbf{x}_i) - b\right), \tag{3}$$

where $\tau > 0$ is the truncated parameter that controls the errors of the outliers. Fig. 1 plots the $L_\tau(\xi)$ in (2) with $\tau = 1.2$, the least square loss $L_{sq}(\xi)$ and the difference between them $L_2(\xi)$. It is clear that the losses of the outliers (samples with larger errors) are bounded by $L_\tau(\xi)$, hence it reduces the effects of the outliers in the R-LSSVM. We will investigate the robustness of the R-LSSVM from a re-weighted viewpoint in Section 3.2.

2.2. Existing algorithms for R-LSSVM

The truncated least squares loss, $L_\tau(\xi)$, is non-convex and non-smooth, and can be easily observed in Fig. 1, but $L_\tau(\xi)$ can be expressed as the difference between two convex functions $L_{sq}(\xi)$ and $L_2(\xi)$ [10], where

$$L_2(\xi) = \begin{cases} 0, & \text{if } |\xi| \le \tau, \\ \frac{1}{2}(\xi^2 - \tau^2), & \text{if } |\xi| > \tau. \end{cases} \tag{4}$$

Then, the R-LSSVM can be transformed to a difference of convex (DC) programming:

$$\min_{\mathbf{w}\in\Re^m, b\in\Re} \frac{\lambda}{2}\mathbf{w}^\top\mathbf{w} + \frac{1}{m}\sum_{i=1}^{m} L_{sq}\big(y_i - \mathbf{w}^\top\varphi(\mathbf{x}_i) - b\big) - \frac{1}{m}\sum_{i=1}^{m} L_2\big(y_i - \mathbf{w}^\top\varphi(\mathbf{x}_i) - b\big). \tag{5}$$

Wang and Zhong [10] and Yang et al. [11] solve the DC programming (5) using the CCCP (Concave–Convex Procedure). Then, through different methods, they both focus on solving the following linear equations (6) iteratively.

$$\begin{bmatrix} I_m + \frac{1}{m\lambda}K & e \\ e^\top & 0 \end{bmatrix}\begin{bmatrix} \beta \\ b \end{bmatrix} = \begin{bmatrix} \mathbf{y} - \gamma^{(t)} \\ 0 \end{bmatrix}, \tag{6}$$

where K is the positive semi-definite kernel matrix satisfying $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i)^\top\varphi(\mathbf{x}_j)$, $(i, j \in M = \{1, \dots, m\})$, $I_m \in \Re^{m\times m}$ is an identity matrix, $\mathbf{y} = (y_1, \dots, y_m)^\top$, $e = (1, \dots, 1)^\top \in \Re^m$, and $\gamma^{(t)} = (\gamma_1^{(t)}, \dots, \gamma_m^{(t)})^\top$ is the value of γ at the tth iteration satisfying

$$\gamma_i^{(t)} \in \partial L_2(\xi_i^{(t)}), \ i = 1, \dots, m, \tag{7}$$

where $\xi_i^{(t)} = y_i - K_{iM}\beta^{(t)} - b^{(t)}$, $K_{iM} = [k(\mathbf{x}_i, \mathbf{x}_1), \dots, k(\mathbf{x}_i, \mathbf{x}_m)]$ is the ith row of the kernel matrix K.

Through iteratively solving (6) with respect to β and b until convergence, the output deterministic function is $f(\mathbf{x}) = \frac{1}{m\lambda}\sum_{i=1}^{m}\beta_i k(\mathbf{x}_i, \mathbf{x}) + b$.

In order to compute (7), Wang and Zhong [10] neglect the non-differentiability points in $L_2(\xi)$ and adopt the following formula:

$$\gamma_i^{(t)} = \begin{cases} 0, & \text{if } |\xi_i^{(t)}| \le \tau, \\ \xi_i^{(t)}, & \text{if } |\xi_i^{(t)}| > \tau, \end{cases} \tag{8}$$

and Yang et al. compute (7) after smoothing the function $L_2(\xi)$ by a piecewise quadratic function [11].

One limitation of these two algorithms is that the solution lacks sparseness. That is because the coefficient matrix of (6) is a non-singular symmetric dense matrix, and the vector on the right side of the equations is also dense. Hence, the training speeds of these two algorithms are slow and they cannot train large-scale problems efficiently.

3. Sparse R-LSSVM algorithm

In this section, we present the primal R-LSSVM and propose a sparse algorithm to obtain a sparse solution of the R-LSSVM. In addition, we illustrate that the R-LSSVM has robustness from a re-weighted viewpoint.

3.1. Primal R-LSSVM

If the loss function is convex, such as in the LSSVM model (1), by duality theory, the optimal solution $\mathbf{w}$ can be represented as

$$\mathbf{w} = \sum_{i=1}^{m}\alpha_i\varphi(\mathbf{x}_i), \tag{9}$$

where $\alpha_i \in \Re$. If the loss function is nonconvex, strong duality does not hold, hence we cannot obtain (9) by duality. However, by the following representer theorem [19,20], it is easy to prove that (9) also holds for model (3) and (5), which is described in Theorem 2.

Theorem 1 (*Representer Theorem*). *[19,20] Suppose we are given a nonempty set χ, a mapping φ from χ to a Hilbert space, a training sample $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m) \in \chi \times \Re$, a monotonically nondecreasing real-valued function $g: \Re_+ \to \Re$ and an arbitrary cost function f: $\Re^m \to \Re$; then, minimizing the regularized risk functional*

$$f(\langle\mathbf{w}, \varphi(\mathbf{x}_1)\rangle, \dots, \langle\mathbf{w}, \varphi(\mathbf{x}_m)\rangle) + g(\|\mathbf{w}\|)$$

admits a representation of the form (9).

Theorem 2. *Assume that φ is a mapping from $\Re^l$ to a Hilbert space. Then, there exists a vector $\alpha \in \Re^m$ such that $\mathbf{w} = \sum_{i=1}^{m}\alpha_i\varphi(\mathbf{x}_i)$ is an optimal solution of (3) and (5).*

Substituting (9) into (3), the R-LSSVM can be translated into the following model in primal space without the implicit feature map $\varphi(\mathbf{x})$:

$$\min_{\alpha\in\Re^m, b\in\Re} \frac{\lambda}{2}\alpha^\top K\alpha + \frac{1}{m}\sum_{i=1}^{m} L_\tau(y_i - K_{iM}\alpha - b), \tag{10}$$

The model (10) is called the primal R-LSSVM.

3.2. Robustness of R-LSSVM from a re-weighted viewpoint

Wang and Zhong [10] illustrated the robustness of the R-LSSVM through experimentation only. Yang et al. [11] explained it using the relationship between the solutions of the R-LSSVM and the W-LSSVM [13]. In this section, we will show that the R-LSSVM has robustness from a re-weighted viewpoint.

In order to explain the robustness of the preceding model (10) more clearly, we propose an equivalent form of L_τ in Lemma 1 from the idea in [21,22].

Lemma 1. *$L_\tau(\xi) = \frac{1}{2}\min\{\xi^2, \tau^2\}$ can be expressed as*

$$L_\tau(\xi) = \min_{\omega\in\Re_+} \frac{1}{2}\omega\xi^2 + \phi(\omega) \tag{11}$$

where

$$\phi(\omega) = \frac{\tau^2}{2}(1-\omega)_+. \tag{12}$$

Proof.

$$\min_{\omega\in\Re_+} \frac{1}{2}\omega\xi^2 + \phi(\omega) = \min_{\omega\in\Re_+} \begin{cases} \frac{1}{2}(\xi^2 - \tau^2)\omega + \frac{1}{2}\tau^2, & \text{if } 0 \le \omega \le 1, \\ \frac{1}{2}\omega\xi^2, & \text{if } \omega > 1, \end{cases}$$
$$= \begin{cases} \frac{1}{2}\xi^2, & \text{if } |\xi| \le \tau, \\ \frac{1}{2}\tau^2, & \text{if } |\xi| > \tau, \end{cases}$$
$$= L_\tau(\xi).$$

Moreover,

$$\omega^* := \arg\min_{\omega\in\Re_+}\left\{\frac{1}{2}\omega\xi^2 + \phi(\omega)\right\} = \begin{cases} 1, & \text{if } |\xi| \le \tau, \\ 0, & \text{if } |\xi| > \tau. \end{cases} \tag{13}$$

□

By Lemma 1 and the research of re-weighted LSSVM in [23], we have

Proposition 1. *Any stationary point of the R-LSSVM (10) can be obtained by solving an iteratively re-weighted LSSVM as follows:*

$$\min_{\alpha\in\Re^m, b\in\Re} \frac{\lambda}{2}\alpha^\top K\alpha + \frac{1}{2m}\sum_{i=1}^{m}\omega_i^{(t)}(y_i - K_{iM}\alpha - b)^2, \tag{14}$$

where $\omega_i^{(t)}$ is the value of t-th iteration of the weight ω_i.

Proof. Substituting (11) into (10), we have

$$\min_{\omega\in\Re_+^m,\alpha\in\Re^m,b\in\Re} J(\alpha,b,\omega) := \frac{\lambda}{2}\alpha^\top K\alpha + \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}\omega_i\xi_i^2 + \frac{1}{m}\sum_{i=1}^{m}\phi(\omega_i), \tag{15}$$

where $\xi_i = y_i - K_{iM}\alpha - b$. Since $J(\alpha, b, \omega)$ is nonconvex. There may be more than one local optimum for (15), but we only consider one of its stationary points. Let (α^*, b^*) be one of the stationary points of (10). By Lemma 1, there exists $\omega^* \in \arg\min_{\omega\in\Re_+^m} J(\alpha^*, b^*, \omega)$ such that $(\alpha^*, b^*, \omega^*)$ be the solution of (15). On the other hand, if $(\alpha^*, b^*, \omega^*)$ is any stationary point of (15), then $(\alpha^*, b^*) = \arg\min_{\alpha\in\Re^m, b\in\Re} J(\alpha, b, \omega^*)$ also solves (10). Hence, we can iteratively solve (15) by alternating direction method (ADM) [24] as follows:

$$(\alpha^{(t)}, b^{(t)}) = \arg\min_{\alpha\in\Re^m, b\in\Re} J(\alpha, b, \omega^{(t-1)}) \tag{16}$$

$$\omega^{(t)} \in \arg\min_{\omega\in\Re_+^m} J(\alpha^{(t)}, b^{(t)}, \omega) \tag{17}$$

Obviously, the optimization problem in (17) has the closed form solution (13). The optimization problem in (16) is just the re-weighted LSSVM (14). □

Because ξ_i denotes the predicted error, similar to the robustness analysis in article [25], the larger $|\xi_i|$ is, the more likely it is that the instance pair $(\mathbf{x}_i, y_i)$ will be an outlier. From (13) and (14), it is observed that when the $|\xi_i|$ is sufficiently large for the outlier instance $(\mathbf{x}_i, y_i)$, the corresponding weight ω_i in (14) will be zero. That is, the truncated least squares loss function L_τ can reduce the influence of samples that are far away from their true targets. This explains the robustness of the R-LSSVM from the re-weighted viewpoint.

3.3. Sparse solution for primal R-LSSVM

Substituting (9) into (5), we obtain a DC programming with regard to α and b as follows:

$$\min_{\alpha\in\Re^m, b\in\Re} H(\alpha, b) = H_1(\alpha, b) - H_2(\alpha, b) \tag{18}$$

with convex functions $H_1(\alpha, b) = \frac{\lambda}{2}\alpha^\top K\alpha + \frac{1}{m}\sum_{i=1}^{m} L_{sq}(y_i - K_{iM}\alpha - b)$ and $H_2(\alpha, b) = \frac{1}{m}\sum_{i=1}^{m} L_2(y_i - K_{iM}\alpha - b)$. We call the model (18) or its equivalent form (10) as the primal R-LSSVM for convenience.

Using the CCCP method in [10,11,26], the solution to the problem (18) can be obtained by iteratively solving the following convex QP until it converges:

$$\begin{aligned}\left(\alpha^{(t+1)}, b^{(t+1)}\right) &= \arg\min_{\alpha\in\Re^m, b\in\Re}\left\{H_1(\alpha, b) - \langle[\alpha^\top, b]^\top, \partial H_2(\alpha^{(t)}, b^{(t)})\rangle\right\} \\ &= \arg\min_{\alpha\in\Re^m, b\in\Re}\left\{H_1(\alpha, b) + \frac{1}{m}\sum_{i=1}^{m}\gamma_i^{(t)}(K_{iM}\alpha + b)\right\},\end{aligned} \tag{19}$$

where $\gamma_i^{(t)}$ is the same as (7) with $\xi_i^{(t)} = y_i - K_{iM}\alpha^{(t)} - b^{(t)}$.

However, the computation of $\gamma_i^{(t)}$ is not simple, because $L_2(\xi)$ is non-differentiable at some points. Inspired by the idea in [27], we smooth $L_2(\xi)$ by the entropy penalty function. Let

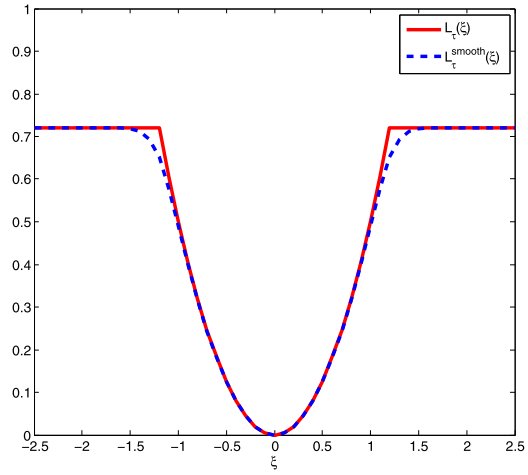

Fig. 2. Plots of the truncated least squares loss $L_\tau(\xi)$ (solid) and the smoothed truncated least squares loss $L_\tau^{smooth}(\xi)$ (dashed) with $p = 5$.

$$\tilde{L}_2(\xi) = \frac{1}{2}\max\left\{0, \xi^2 - \tau^2\right\} + \frac{1}{2p}\log\left(1 + \exp\left(-p|\xi^2 - \tau^2|\right)\right). \tag{20}$$

Then, we have $\tilde{L}_2(\xi) \to L_2(\xi)$ whenever $p \to +\infty$. $\tilde{L}_2(\xi)$ is the smooth approximation of $L_2(\xi)$, and the upper bound of the difference between $\tilde{L}_2(\xi)$ and $L_2(\xi)$ is $\frac{\log 2}{p}$. If we set p sufficiently large such as $p = 10^4$, the difference between them can be neglected. Fig. 2 shows the comparison between $L_\tau(\xi)$ and the smoothed truncated least squares loss function $L_\tau^{smooth}(\xi) = L_{sq}(\xi) - \tilde{L}_2(\xi)$ with $p = 5$.

The derivative of $\tilde{L}_2(\xi)$ is:

$$\tilde{L}_2'(\xi) = \xi \cdot \frac{\min\left\{1, \exp\left[p\left(\xi^2 - \tau^2\right)\right]\right\}}{1 + \exp\left(-p|\xi^2 - \tau^2|\right)}. \tag{21}$$

Replacing $L_2(\xi_i)$ with $\tilde{L}_2(\xi_i)$ in (7), the $\gamma_i^{(t)}$ in (19) is calculated as follows:

$$\gamma_i^{(t)} = \frac{\xi_i^{(t)}\min\left\{1, \exp\left[p\left(\xi_i^{(t)^2} - \tau^2\right)\right]\right\}}{1 + \exp\left(-p\left|\xi_i^{(t)^2} - \tau^2\right|\right)} \tag{22}$$

Yang et al. [11] also adopted a smoothing procedure, but their method required that the smoothing parameter be tuned in order to be most effective. That makes the parameter adjustment procedure complex. In comparison, our smoothing strategy (based on the entropy penalty function) does not require the tuning of any parameter. What we need to do is set a large value for p in (22).

3.4. Sparse solution for primal R-LSSVM

After obtaining $\gamma_i^{(t)}$ by (22), $(\alpha^{(t+1)}, b^{(t+1)})$ in (19) are the solutions of the following system of linear equations:

$$\begin{bmatrix} m\lambda K + KK^\top & Ke \\ e^\top K^\top & m \end{bmatrix}\begin{bmatrix}\alpha \\ b\end{bmatrix} = \begin{bmatrix} K \\ e^\top\end{bmatrix}\left(\mathbf{y} - \gamma^{(t)}\right). \tag{23}$$

It appears that (23) is more complicated than (6) upon first impression. However, the coefficient matrix of (6) is a nonsingular symmetric dense matrix, which leads to a non-sparse solution of (6). In comparison, the coefficient matrix of (23) may be low rank if the related kernel matrix, K, has low rank or can be approximated by a low-rank matrix. In this situation, (23) may have a sparse solution, which partially overcomes the limitation of the previous methods.

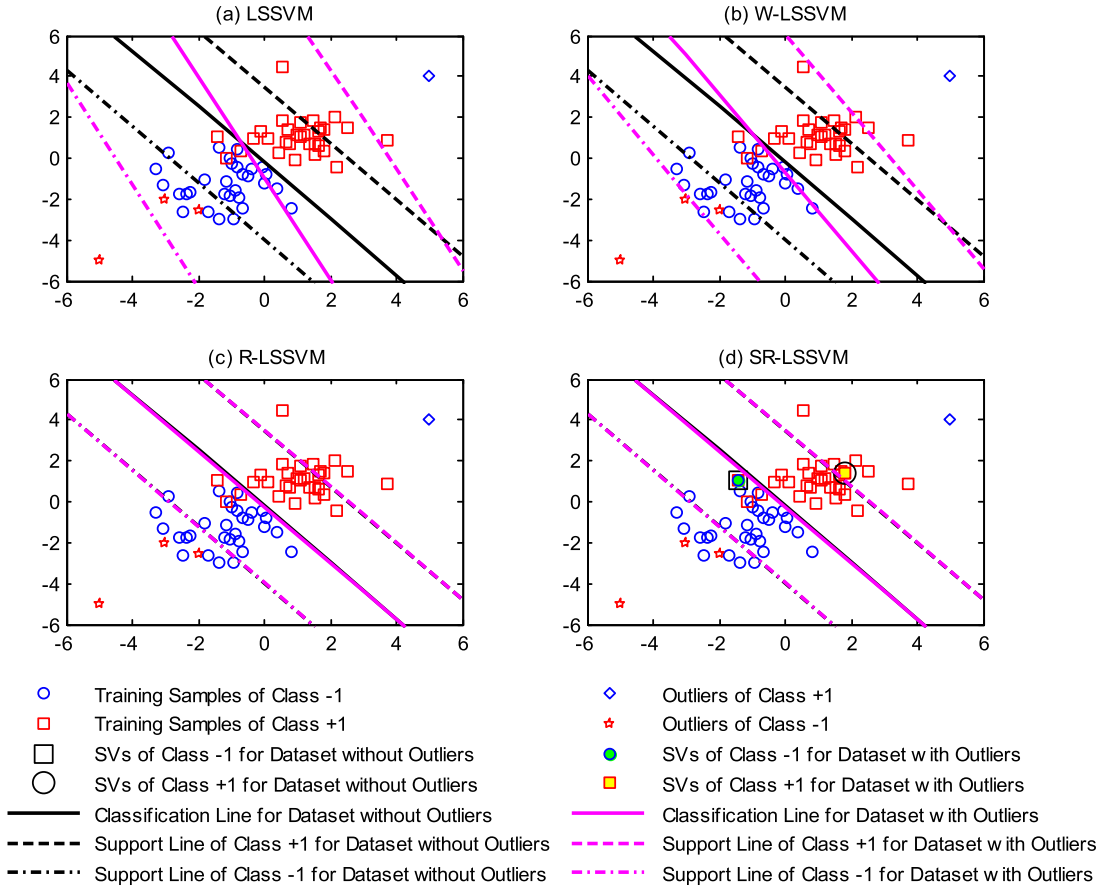

Fig. 3. Comparison of the proposed approach SR-LSSVM with LSSVM, W-LSSVM and R-LSSVM for linearly inseparable classification dataset with and without outliers. The numbers of the support vectors (SVs) are both 2 for datasets with and without outliers for SR-LSSVM in Fig.3(d). We do not mark SVs in the subgraphs (a)–(c), because almost all of training samples are SVs for LSSVM, W-LSSVM and R-LSSVM. For dataset without outliers, the test accuracy are all 91.50% for these four algorithms, and for dataset with outliers, the test accuracy are 89.50%, 90.00%, 91.50% and 91.50% for LSSVM, W-LSSVM, R-LSSVM and SR-LSSVM respectively.

Now, we discuss the sparse optimization solution of (23) as soon as the kernel matrix can be approximated by a low-rank matrix.

Using the second equation of (23), we obtain $b = \frac{1}{m}\left[e^\top(\mathbf{y} - \gamma^{(t)}) - e^\top K\alpha\right]$. After eliminating b, (23) is simplified to the following linear equation:

$$(m\lambda K + KK - \tfrac{1}{m}Kee^\top K)\alpha = K\left[(\mathbf{y} - \gamma^{(t)}) - \tfrac{e^\top(\mathbf{y}-\gamma^{(t)})}{m}e\right]. \quad (24)$$

Nyström Approximation is the most popular method available for obtaining a low-rank approximation of kernel matrix K (see [15,28–32]). Here, we employ Zhou's incomplete pivoting Cholesky factorization method in [15]. It costs $O(mr^2)$ to obtain the best rank-r Nyström type approximation of K under the trace norm. Using this method, the set $B \subset M$ (the number of elements in B is r, i.e. $|B| = r$) and the full column rank matrix $P \in \Re^{m\times r}$ satisfying $PP^\top = K_{MB}K_{BB}^{-1}K_{MB}^\top$ are obtained, where $K_{MB} \in \Re^{m\times r}$ is a sub-matrix of K whose elements are K_{ij} for $i \in M$ and $j \in B$, and $K_{BB} \in \Re^{r\times r}$ is also a sub-matrix of K whose elements are K_{ij} for $i \in B$ and $j \in B$. Moreover, only the well-chosen r columns of the kernel matrix and its diagonal elements are needed in the algorithm. If K_{MB} is obtained by some other low-rank approximation methods [29–32], let $P = K_{MB}K_{BB}^{-\frac{1}{2}}$ and the following analysis is the same.

Substituting $PP^\top$ into (24) instead of K, (24) is simplified as:

$$(m\lambda I_r + P^\top P - \tfrac{1}{m}P^\top ee^\top P)P^\top\alpha = P^\top\left[(\mathbf{y} - \gamma^{(t)}) - \tfrac{e^\top(\mathbf{y}-\gamma^{(t)})}{m}e\right], \quad (25)$$

where $I_r \in \Re^{r\times r}$ is an identity matrix. By permuting rows of matrix P, we obtain $[P_B^\top, P_N^\top]^\top$, where $P_B \in \Re^{r\times r}$ is a full rank and lower triangular matrix if P is obtained as [15], and P_N is comprised by the rest $m - r$ rows of P. Correspondingly, let $\alpha = [\alpha_B^\top, \alpha_N^\top]^\top$, then we have

$$\begin{cases} \alpha_B = \left(P_B^\top\right)^{-1}J^{-1}P^\top\left(\mathbf{y} - \gamma^{(t)} - \frac{e^\top(\mathbf{y}-\gamma^{(t)})}{m}e\right), \\ \alpha_N = 0 \end{cases} \quad (26)$$

is the sparse solution of (25), where

$$J = \left(m\lambda I_r + P^\top P\right) - \frac{1}{m}\left(e^\top P\right)^\top e^\top P. \quad (27)$$

Therefore, the sparse R-LSSVM (SR-LSSVM) algorithm is obtained by iteratively updating $(\alpha^{(t+1)}, b^{(t+1)})$ as follows:

$$\begin{cases} \alpha_B^{(t+1)} = \left(P_B^\top\right)^{-1}J^{-1}P^\top\left(\mathbf{y} - \gamma^{(t)} - \frac{e^\top(\mathbf{y}-\gamma^{(t)})}{m}e\right), & (28) \\ \alpha_N^{(t+1)} = 0, \\ b^{(t+1)} = \frac{1}{m}\left[e^\top\left(\mathbf{y} - \gamma^{(t)}\right) - e^\top PP_B^\top\alpha_B^{(t+1)}\right]. & (29) \end{cases}$$

3.5. Sparse R-LSSVM algorithm

From the above analysis, our SR-LSSVM algorithm is listed as Algorithm 1.

After obtaining the optimal α_B and b by Algorithm 1, the decision function for regression is:

Algorithm 1 *SR-LSSVM* —Sparse **R-LSSVM**.

Input: Training set $\mathbf{T}=\{(\mathbf{x}_i,y_i)\}_{i=1}^m$, kernel function $k(x,z)$, the regularization parameter $\lambda>0$, the truncated parameter τ, the stop criterion $\varepsilon>0$, $r=|B|$.

Output: α_B and b.

1: Find P and B such that $K\approx PP^\top$,$\gamma^{(0)}=0\in\Re^m$;
2: Compute J as (27). Set $t=0$;
3: Update $\alpha_B^{(t+1)}$ and $b^{(t+1)}$ by (28) and (29);
4: Set $\xi^{(t+1)}=\mathbf{y}-PP_B^\top\alpha_B^{(t+1)}-b^{(t+1)}$, and compute $\gamma_i^{(t+1)}$ by (22);
5: **if** $\left\|\gamma^{(t+1)}-\gamma^{(t)}\right\|<\varepsilon$ **then**
6: stop with $\alpha_B=\alpha_B^{(t+1)}$ and $b=b^{(t+1)}$;
7: **else**
8: let $t=t+1$, go to step 3.
9: **end if**

$$f(\mathbf{x})=\sum_{i\in B}\alpha_i k(\mathbf{x}_i,\mathbf{x})+b. \tag{30}$$

For classification, the decision function is $\mathrm{sgn}f(\mathbf{x})$. We provide some comments about Algorithm 1.

For Algorithm 1, the computation cost of step 1 and step 2 are both $O(mr^2)$ $(r\ll m)$ [15]. The complexity of iteratively calculating step 3 is $O(T_s mr)$, where T_s is the total iterative number of the SR-LSSVM. Thus, the overall complexity of this algorithm is $O\left(mr^2+T_s mr\right)$. In contrast, the computational complexities of the Wang and Yang R-LSSVM algorithms in [10] and [11] are both $O(T_d m^3)$, where T_d is the iterations of their algorithms. It is obvious that our method has simpler computational complexity than existing approaches.

Comment 1. When $t=0$, the first cycle of Algorithm 1 is equivalent to the optimization of the P-LSSVM [15]. That is because if $\gamma^{(t)}=0$, (24) is the same as the equation system derived by primal LSSVM (P-LSSVM) in [15].

Comment 2. To promote computational efficiency, (28) can be rewritten as:

$$\alpha_B^{(t+1)}=\alpha_B^{(0)}-G\left(P_{S_t}^\top\gamma_{S_t}^{(t)}-\frac{e^\top\gamma^{(t)}}{m}\hat{p}\right), \tag{31}$$

where $G=\left(P_B^\top\right)^{-1}J^{-1}\in\Re^{r\times r}$, $\hat{p}=P^\top e\in\Re^r$, $\alpha_B^{(0)}=G(P^\top\mathbf{y}-\frac{e^\top\mathbf{y}}{m}\hat{p})\in\Re^r$ is the sparse solution of the primal LSSVM, $S_t\subset M$ is the index set of nonzero elements of $\gamma^{(t)}$, $\gamma_{S_t}^{(t)}$ is a vector comprised of nonzero elements of $\gamma^{(t)}$, P_{S_t} is comprised by several rows of P, and the indexes of these rows in P correspond to the elements in S_t.

Then, step 2 and 3 in Algorithm 1 can be replaced with the following:

Step 2': Compute J, G, $\hat{p}$ and α_{LS}. Set $t=0$;

Step 3': Update $\alpha_B^{(t+1)}$ and $b^{(t+1)}$ by (31) and (29) respectively.

Hence, when G, $\hat{p}$ and $\alpha_B^{(0)}$ are pre-calculated, the computational complexity of Step 3 in Algorithm 1 is reduced from $O(mr)$ to $O(|S_t|r)$ (the most cost of (31) is from $P_{S_t}^\top\gamma_{S_t}^{(t)}$). Therefore, the method described in this comment can improve the efficiency of Algorithm 1.

Comment 3. In Algorithm 1, the parameter τ limits the upper bound of the loss function. τ should not be set to a too large or too small value. An improper τ value results in poor generalization performance. To overcome the sensitivity of the loss function to τ, we can tone τ as follows. Firstly, we set a slightly larger τ, such as $\tau=\delta*\max\{|\xi_1|,\ldots,|\xi_m|\}$, where $0<\delta<1$. Then add the following step between the step 3 and step 4 in Algorithm 1: reduce τ if $\left\|\gamma^{(t+1)}-\gamma^{(t)}\right\|$ is small until $\tau\le\tau_{\min}$, where $\tau_{\min}$ is the minimum of τ we set.

Comment 4. *Parallel computing potential.* In Algorithm 1, several calculations are easy to perform, thus serial computing is appropriate for them. However, for several costly calculations, we can utilize parallel computing to further improve computing efficiency. The main computational cost of Algorithm 1 is from computing $P^\top P$, which can be implemented in parallel. For example, P can be partitioned into k chunks according to the row satisfying $P^\top=[P_1^\top,\ldots,P_k^\top]$, thus $P^\top P=\sum_{i=1}^k P_i^\top P_i$ which can be efficiently calculated by the parallel algorithm of matrix multiplication, where P_i is the i-th block of the matrix P.

Table 1
Comparison of different algorithms on medium-scale benchmark classification datasets with outliers (10%). The standard deviations are given in brackets. 'nSVs' refers to the average number of support vectors. m and n are the numbers of training and testing samples respectively, l is the dimension of data. The best values are highlighted in bold.

Data	Algorithms	Training times(s)	nSVs	Accuracy(%)
Splice	ProCRC	**0.03**(0.00)	–	**85.65**(0.01)
$m=1,000$	C-SVC	0.12(0.01)	820.8(8.3)	76.38(0.08)
$n=2,175$	LSSVM	0.19(0.01)	1000(0)	75.99(0.10)
$l=60$	W-LSSVM	0.20(0.01)	1000(0)	76.04(0.10)
	FS-LSSVM	0.42(0.02)	**100**(0)	76.66(0.07)
	R-LSSVM	0.22(0.01)	947.6(19.5)	80.50(0.04)
	SR-LSSVM	0.19(0.01)	**100**(0)	81.27(0.03)
Pendigits	ProCRC	**0.002**(0.00)	–	99.81(0.002)
$m=1,466$	C-SVC	0.36(0.02)	433.5(7.7)	99.95(0.001)
$n=733$	LSSVM	0.12(0.01)	1466(0)	99.26(0.005)
$l=16$	W-LSSVM	0.18(0.01)	1464.6(1.7)	99.92(0.002)
	FS-LSSVM	0.19(0.01)	**73**(0)	99.90(0.001)
	R-LSSVM	0.19(0.01)	1436.8(8.0)	99.09(0.005)
	SR-LSSVM	0.03(0.00)	**73**(0)	**99.96**(0.001)
Satimage	ProCRC	**0.02**(0.00)	–	98.05(0.002)
$m=2,110$	C-SVC	0.25(0.01)	693.5(13.6)	99.86(0.001)
$n=931$	LSSVM	0.76(0.01)	2109.6(0.7)	99.23(0.002)
$l=36$	W-LSSVM	0.90(0.02)	1897.5(2.1)	99.91(0.001)
	FS-LSSVM	0.50(0.02)	**105**(0)	97.93(0.008)
	R-LSSVM	0.87(0.03)	1916.5(15.1)	99.88(0.001)
	SR-LSSVM	0.27(0.00)	**105**(0)	**99.97**(0.001)
USPS	ProCRC	**0.79**(0.02)	–	99.36(0.000)
$m=2,199$	C-SVC	0.92(0.02)	646.9(14.4)	99.34(0.001)
$n=623$	LSSVM	2.53(0.01)	2198.7(0.5)	99.34(0.002)
$l=256$	W-LSSVM	2.67(0.01)	1973.8(3.1)	99.49(0.001)
	FS-LSSVM	1.21(0.01)	109(0)	98.28(0.006)
	R-LSSVM	2.60(0.02)	2002.4(14.9)	**99.52**(0.000)
	SR-LSSVM	1.91(0.01)	**108.7**(0.3)	**99.52**(0.000)
Mushrooms	ProCRC	**0.13**(0.01)	–	99.87(0.002)
$m=5,614$	C-SVC	2.33(0.04)	2244.8(27.1)	99.99(0.000)
$n=2,708$	LSSVM	5.75(0.08)	5415.8(0.4)	98.67(0.002)
$l=112$	W-LSSVM	7.41(0.19)	4837.8(12.2)	99.90(0.001)
	FS-LSSVM	2.69(0.02)	**268.9**(0.7)	99.66(0.003)
	R-LSSVM	7.48(0.18)	4928(16.9)	99.71(0.001)
	SR-LSSVM	1.28(0.01)	270(0)	**100**(0)
Protein	ProCRC	**2.58**(0.24)	–	66.26(0.000)
$m=8,186$	C-SVC	55.64(0.47)	5486.8(27.1)	77.98(0.004)
$n=3,509$	LSSVM	22.67(0.30)	8185.9(0.32)	78.22(0.002)
$l=357$	W-LSSVM	27.77(0.30)	8185.7(0.67)	**78.24**(0.003)
	FS-LSSVM	27.55(0.61)	**408.1**(1.20)	77.00(0.003)
	R-LSSVM	25.17(0.81)	7876.7(46.6)	78.23(0.004)
	SR-LSSVM	11.65(0.03)	409(0)	78.04(0.002)

3.6. Convergence analysis

The CCCP is globally or locally convergent; see [26,33,34]. Similar to the convergence proof of the DCA (DC Algorithm) for general DC programming in article [35], we have the following Lemma.

Lemma 2. *If the optimal value of the problem (18) is finite, and the infinite sequences $(\alpha^{(t)},b^{(t)})$ and $\partial H_2(\alpha^{(t)},b^{(t)})$ are bounded, then every limit point $(\widetilde{\alpha},\widetilde{b})$ of the sequence $(\alpha^{(t)},b^{(t)})$ is a generalized KKT (Karush–Kuhn–Tucker) point of $H_1(\alpha,b)-H_2(\alpha,b)$.*

Obviously, the objective function of (18) and (10) is bounded below. Assume the prediction error variable $\xi_i^{(t)}$ is bounded, which is reasonable in a real application, then $\gamma_i^{(t)}$ is bounded by (22).

Table 2
Comparison of different algorithms on large-scale benchmark classification data sets with outliers (10%). The standard deviations are given in brackets. 'nSVs' refers to average number of support vectors. m and n are the numbers of training and testing samples respectively, l is the dimension of data.

Datasets	Algorithms	Training time(s)	nSVs	Accuracy(%)
IJCNN1	ProCRC	**0.22**(0.0)	–	90.50(0.000)
$m = 49,990$	C-SVC	84.3(1.6)	17103(75.6)	92.16(0.000)
$n = 91,701$	FS-LSSVM	34.9(0.8)	398.8(1.0)	94.17(0.014)
$l = 22$	CSI	63.0(1.2)	400(0)	95.10(0.002)
	PCP-LSSVM	2.8(0.1)	215.6(1.6)	95.39(0.002)
	SR-LSSVM	2.8(0.1)	215(0)	**95.47**(0.002)
Cod-RNA	ProCRC	**0.14**(0.0)	–	92.29(0.001)
$m = 59,535$	C-SVC	86.3(0.8)	22095(42.6)	94.91(0.000)
$n = 271,617$	FS-LSSVM	36.0(0.8)	399.7(0.5)	94.84(0.001)
$l = 8$	CSI	78.6(0.4)	400(0)	94.99(0.001)
	PCP-LSSVM	6.0(0.0)	400(0)	94.78(0.001)
	SR-LSSVM	6.4(0.1)	400(0)	**95.04**(0.001)
SensIT Vehicle	ProCRC	**1.04**(0.0)	–	71.46(0.012)
(acoustic)	C-SVC	278.3(5.6)	28302(107.0)	74.87(0.001)
$m = 60,562$	FS-LSSVM	75.1(0.5)	399.2(1.0)	**78.17**(0.002)
$n = 15,125$	CSI	74.3(0.2)	400(0)	77.62(0.003)
$l = 50$	PCP-LSSVM	40.1(0.1)	400(0)	77.52(0.001)
	SR-LSSVM	41.4(0.1)	400(0)	77.73(0.001)
Skin-nonskin	ProCRC	**0.11**(0.0)	–	94.60(0.003)
$m = 163,371$	C-SVC	1329.4(10.6)	59910(146.7)	99.30(0.001)
$n = 81,686$	FS-LSSVM	42.6(0.7)	199.2[1](0.8)	99.82(0.000)
$l = 3$	CSI	42.9(0.7)	100[2](0)	99.84(0.000)
	PCP-LSSVM	32.7(0.6)	400(0)	**99.86**(0.000)
	SR-LSSVM	34.8(2.5)	400(0)	**99.86**(0.000)
Checkerboard(1M)	ProCRC	**0.88**(0.4)	–	60.01(0.002)
$m = 1,000,000$	C-SVC[3]	–	–	–
$n = 600,000$	FS-LSSVM	139.9(5.9)	299(0.9)	99.36(0.001)
$l = 2$	CSI	154.4(16.2)	56.6(2.1)	96.51(0.003)
	PCP-LSSVM	80.8(0.4)	300(0)	**99.37**(0.000)
	SR-LSSVM	129.3(0.5)	300(0)	98.12(0.000)

[1] The working set size was set as 200 for FS-LSSVM, because if we set the working set size at 400, the software cannot operate normally.
[2] The maximal rank of CSI algorithm was 100 for Skin-nonskin dataset. If we set this number bigger, the procedure may make an error.
[3] Training time is too long.

Therefore $(\alpha^{(t)}, b^{(t)})$ and $\partial H_2(\alpha^{(t)}, b^{(t)}) = [{\gamma^{(t)}}^\top K, {\gamma^{(t)}}^\top e]^\top$ are also bounded because of the boundedness of $(P_B^\top)^{-1}$, J^{-1} and $P^\top$ in (28) and (29). By Lemma 2, we get the following theorem.

Theorem 3. *Assume the predict error* $\xi = y - \mathbf{w}^\top\varphi(\mathbf{x}) - b$ *is bounded for all given samples* $(\mathbf{x}, y)$ *with selected parameters* $\mathbf{w}$ *and* b, *then limit point of the sequence* $(\alpha^{(t)}, b^{(t)})$ *is the generalized KKT point of the problem (18), that is, Algorithm 1 is convergent.*

4. Numerical experiments and discussions

In order to verify the effectiveness of the proposed algorithm, we compare the following algorithms with our SR-LSSVM:

- ProCRC: the Probabilistic Collaborative Representation based Classifier method in [16]. The training time for ProCRC in our tables refers to the total running time because it is a non-parametric classifier. We adopted the Sherman-Morrison-Woodbury (SMW) formula [36,37] to raise the operating efficiency in the experiment.
- C-SVC and ϵ-SVR: the Support Vector Machine for Classification and Regression respectively in [38] are implemented in LIBSVM software and coded in the C programming language[1].
- LSSVM: the Least Square Support Vector Machine in [1]. The solution is deduced by KKT conditions. Our code utilizes Cholesky factorization in solving the solution to LSSVM.
- W-LSSVM: the Weighted Least Square Support Vector Machine [13]. Our code for W-LSSVM is written according to the W-LSSVM algorithm in [13].
- FS-LSSVM: the Fixed Size Least Square Support Vector Machine [7]. In our experiments, the FS-LSSVM is operated in the LS-SVMlab v1.8 software [39][2].
- R-LSSVM: Yang's algorithm for the Robust Least Square Support Vector Machine [11]. In our experiment, the code for R-LSSVM is written according to the pseudo-code in [11]. Smoothing parameter in this algorithm is denoted by h.

For several large-scale problems, some algorithms mentioned above must calculate the whole kernel matrix which requires too much computer memory. These algorithms include the LSSVM, W-LSSVM and the R-LSSSVM. Therefore, we only compare the SR-LSSVM with the ProCRC, C-SVC/ϵ-SVR, FS-LSSVM, and the following sparse algorithms:

- PCP-LSSVM: the Pivoted Choleskian of Primal Least Square Support Vector Machine [15][3]. It obtains sparse solutions for the LSSVM in primal space.
- CSI: Cholesky with Side Information [40]. Codes are available in http://www.di.ens.fr/~fbach/csi/index.html.

All computations were implemented in Windows 8 with Matlab R2014a. All experiments were run on a PC with an Intel Core i5-

[1] Codes are available in https://www.csie.ntu.edu.tw/~cjlin/libsvm/.

[2] Codes are available in http://www.esat.kuleuven.be/sista/lssvmlab/.

[3] Codes and article can be downloaded from http://web.xidian.edu.cn/sszhou/paper.html.

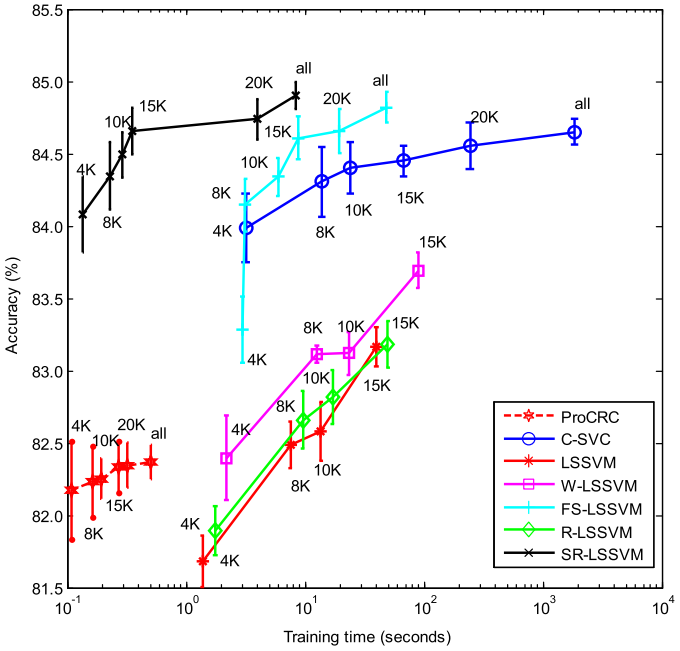

Fig. 4. The training time, accuracy and their standard deviations of different algorithms on six subsets of the Adult data set with outliers (about 10%). The markers 4K to 20K denote 4000 to 20,000 samples in the training sets. 'all' means using all 32,561 samples. LSSVM, W-LSSVM and R-LSSVM were only implemented on the data sets containing less than 15,000 samples due to memory limitation of computer. The basic subset size for SR-LSSVM and working set size for FS-LSSVM were both set as 400. In figure, the more left and upper, the better.

Table 3
Comparison the results of Cod-RNA dataset with different rates of outliers (0%, 5% and 10%). The standard deviations are given in brackets. '$R_{outlier}$' refers to the rate of outliers.

	Testing accuracy(%)		
Algorithms	$R_{outlier}=0\%$	$R_{outlier}=5\%$	$R_{outlier}=10\%$
ProCRC	95.11(0.000)	94.54(0.000)	92.29(0.001)
FS-LSSVM	96.18(0.001)	95.91(0.001)	95.03(0.001)
CSI	96.15(0.000)	96.02(0.001)	94.89(0.002)
PCP-LSSVM	96.31(0.000)	96.13(0.000)	94.92(0.001)
SR-LSSVM	**96.32**(0.000)	**96.29**(0.000)	**95.10**(0.001)

4210U CPU and a maximum of 8 G bytes of memory available for all processes.

We fixed the values of the smoothing parameter, $p=10^4$, in the SR-LSSVM and the stop criterion, $\varepsilon=10^{-2}$, respectively. For all the datasets, we used a cross-validation procedure and grid search to search the best values of the parameter λ,μ,σ,τ and h, where λ and μ are regularization parameters, σ is the parameter in the Gaussian kernel function $k(\mathbf{x}_i,\mathbf{x}_j)=\exp(-\sigma\|\mathbf{x}_i-\mathbf{x}_j\|^2)$, τ is the truncated parameter in R-LSSVMs, and h is the smoothing parameter in the R-LSSVM algorithm.

In ProCRC, we used the SMW formula to improve computing efficiency. Thus, the time listed in the tables is much lower than the time spent by directly running the code on authors' website[4]. For the R-LSSVM, the training times in our article are much lower than those in papers [10,11] for the same datasets, and the total complexity is reduced from $O(T_d m^3)$ [10,11] to $O(m^3+T_d m^2)$. This is because the coefficient matrix of (6) was decomposed by Cholesky factorization once, and such decomposition is unchanged per loop.

[4] Codes are available in http://www4.comp.polyu.edu.hk/~cslzhang/papers.htm.

Table 4
Comparison of different algorithms on medium-scale benchmark regression data sets with outliers (10%). The standard deviations are given in brackets. 'nSVs' refers to average number of support vectors. m and n are the numbers of training and testing samples respectively, l is the dimension of data.

Data	Algorithms	Training time(s)	nSVs	RMSE
Mg	ϵ-SVR	0.030(0.006)	863.3(46.8)	0.129(0.005)
$m=923$	LSSVM	0.108(0.006)	923(0)	0.132(0.006)
$n=462$	W-LSSVM	0.125(0.003)	897.8(8.4)	0.132(0.005)
$l=6$	FS-LSSVM	0.041(0.004)	**46**(0.9)	0.126(0.005)
	R-LSSVM	0.112(0.004)	923(0)	0.132(0.006)
	SR-LSSVM	**0.023**(0.004)	**46**(0)	**0.125**(0.004)
Winequality	ϵ-SVR	0.074(0.004)	911.9(12.4)	0.652(0.017)
$m=1,066$	LSSVM	0.054(0.003)	1066(0)	0.724(0.034)
$n=533$	W-LSSVM	0.080(0.006)	977.6(5.4)	**0.646**(0.022)
$l=11$	FS-LSSVM	0.171(0.006)	**42.4**(1.2)	0.748(0.033)
	R-LSSVM	0.091(0.008)	944.8(24.4)	**0.646**(0.028)
	SR-LSSVM	**0.013**(0.001)	**42.4**(1.2)	**0.646**(0.023)
Abalone	ϵ-SVR	0.957(0.095)	2772.2(2.2)	2.211(0.071)
$m=2,784$	LSSVM	1.215(0.094)	2784(0)	3.271(0.199)
$n=1,393$	W-LSSVM	1.468(0.049)	2784(0)	2.250(0.217)
$l=8$	FS-LSSVM	0.225(0.004)	**48.2**(2.2)	2.188(0.072)
	R-LSSVM	1.494(0.039)	2784(0)	2.251(0.119)
	SR-LSSVM	**0.060**(0.004)	48.5(1.7)	**2.183**(0.091)
Tic	ϵ-SVR	2.654(0.083)	1536.8(38.4)	0.235(0.005)
$m=6,548$	LSSVM	4.294(0.068)	6540.7(2.5)	0.229(0.005)
$n=2,374$	W-LSSVM	7.196(0.102)	534.4(24.6)	0.243(0.006)
$l=85$	FS-LSSVM	4.391(0.089)	**395.5**(2.1)	0.235(0.005)
	R-LSSVM	4.416(0.043)	6540.7(2.5)	**0.229**(0.005)
	SR-LSSVM	**1.751**(0.022)	399.9(0.3)	**0.229**(0.002)

4.1. Classification experiments

In this section, we test one synthetic classification dataset and several benchmark classification datasets to illustrate the effectiveness of the SR-LSSVM. For the benchmark datasets, each attribute of the samples was normalized into $[-1,\ 1]$ for all the algorithms except the ProCRC. In the ProCRC, every sample was normalized to have unit l_2-norm according to article [16]. To compare the performances of the algorithms mentioned at the beginning of this section, we separate the datasets into two groups: the medium-scale dataset group and the large-scale dataset group. The results of the experiments on Adult dataset in Section 4.1.3 show the reason why we separate the datasets into two groups. The benchmark datasets can be downloaded from [41]. Finally, we test the robustness of our proposed algorithm on large-scale datasets with different rates of outliers. Outliers were generated by the following procedure. We chose 30% of samples that were far away from the decision hyperplane, then randomly sampled 1/3 of them and flipped their labels to simulate outliers.

4.1.1. Synthetic classification dataset experiment

To compare the robustness and sparseness of the four algorithms (LSSVM, W-LSSVM, R-LSSVM, and SR-LSSVM), we conducted an experiment on a linear binary classification dataset that included 60 training samples and 100 test samples. Fig. 3 shows the experimental results. To simulate outliers, we added four training samples labeled with incorrect classes. They are marked as '$\diamond$' and '$*$' for positive and negative classes, respectively. Through the grid search, we obtained the best parameter values for this dataset: $m\lambda=10^{-2},\ \tau=1.5,\ h=0.01$.

Fig. 3 illustrates that the decision lines of the LSSVM and W-LSSVM algorithms change greatly after adding outliers, and that these two methods have lower accuracy than SR-LSSVM and R-LSSVM. In contrast, the decision boundaries of the SR-LSSVM and R-LSSVM are almost unchanged, and their accuracy remains stable before and after adding outliers. Thus, the SR-LSSVM is insensitive to outliers. Moreover, almost all of training samples are SVs

for the LSSVM, W-LSSVM, and R-LSSVM, while, for the SR-LSSVM, the support vector sizes are both only two for datasets with and without outliers. Therefore, the proposed algorithm is sparseness, which can accelerate the training speed of our approach in processing large-scale problems.

4.1.2. Medium-scale benchmark classification datasets experiments

The proposed SR-LSSVM is then tested on several medium-scale benchmark classification datasets. The selected parameters of the algorithms for every dataset are listed in Table A1. The information of datasets are listed in Table 1, and the detailed information of several multi-class datasets are listed below.

- *Pendigits*: It is a pen-based recognition of handwritten digits data set to classify the digits 0 to 9. We only classify the digit 3 versus 4 here.
- *Satimage*: It is comprised by six classes. Here the task of classifying class 1 versus 6 is trained.
- *USPS*: It is a muti-class dataset with 10 classes. Here, a binary classification problem is trained to separate class 1 from class 2.
- *Protein*: Protein is a multi-class dataset with three classes. Here, a binary classification problem is trained to separate class 1 from class 2.

Table 1 reports experimental results for the medium-scale classification datasets with outliers. The best results are highlighted in bold. In Table 1, we set $r = |B| = 0.05\ \ m$ for SR-LSSVM and FS-LSSVM for all datasets except Splice ($r = 0.1\,m$). All the algorithms independently operated 10 times to get the unbiased results.

Table 1 illustrates that our proposed method, the SR-LSSVM, has higher accuracy than any of the other approaches on most datasets. With regard to training time, the ProCRC is the fastest algorithm, but its accuracy is not comparable to the SR-LSSVM. The SR-LSSVM is much faster than the other LSSVM-based approaches, including the LSSVM, W-LSSVM and R-LSSVM. The C-SVC performs well on some medium-scale datasets in training speed, but on some larger-scale datasets, such as Protein and Mushroom, the running speeds of the C-SVC are slower than that of the SR-LSSVM. In addition, the accuracy of the C-SVC is lower than that of the SR-LSSVM.

In terms of sparseness, the SR-LSSVM and FS-LSSVM need much fewer support vectors than other approaches. However, the accuracy of the FS-LSSVM is lower than that of the SR-LSSVM, and the FS-LSSVM takes more time than the SR-LSSVM on all datasets. The C-SVC also displays sparsity, but its support vector size is much larger than that of the SR-LSSVM and FS-LSSVM, partly because outliers often have larger Lagrange multipliers.

4.1.3. Adult dataset experiments

To investigate the performance of each algorithm on datasets of different sizes, we randomly chose 4,000, 8,000, 10,000, 15,000, 20,000 and all of the 32,561 training samples from the adult data training set [41]. The test set size keeps 16,281.

Fig. 4 shows the experimental results of the approaches utilized on the datasets with outliers (10%). Regarding the accuracy on these datasets, in general, the SR-LSSVM, FS-LSSVM, and C-SVC perform better than other methods, and our SR-LSSVM method performs the best. In terms of the training speed, the ProCRC is the fastest algorithm, but its classification accuracy is low. In addition, from Fig. 4, we can draw the conclusion that for the medium-scale training datasets, especially those with less than 8,000 samples, every comparison algorithm runs fast. However, if the training set size exceeds 20,000, the LSSVM, W-LSSVM and R-LSSVM cannot operate on our common computer due to lack of memory. Therefore, for the large-scale benchmark datasets, we do not compare our SR-LSSVM method with the LSSVM, W-LSSVM and R-LSSVM. Moreover, Fig. 4 also shows that the training time of C-SVC increases rapidly as the sizes of training samples grow larger.

4.1.4. Large-scale benchmark classification datasets experiments

We compare our SR-LSSVM method with the ProCRC and several sparse algorithms, including the C-SVC, FS-LSSVM, PCP-LSSVM and CSI, on the large-scale datasets. The optimal parameters for every dataset are listed in Table A2, and information on some of the datasets is listed as follows, for others see Table 2.

- *SensIT Vehicle (acoustic)*: This dataset has three classifications labeled 1, 2, and 3. We only chose the second and the third classifications to operate.
- *Checkerboard(1M)*: It is first given in [42] and widely used to show the effectiveness of nonlinear kernel methods [15,27,43]. The dataset was generated by the following method: randomly sampled 1,600,000 points from the regions $[0,\ 1]\times[0,\ 1]$ and labeled two classes by 4×4 XOR problem. Then, we randomly chose 1,000,000 points as training samples and the remaining 600,000 points as test samples.

Table 2 reports the experimental results for the large-scale datasets with outliers (10%). All the algorithms are independently operated five times to obtain the unbiased results for every dataset. The best results are highlighted in bold.

From Table 2, it is obvious that the SR-LSSVM achieves good performance. It achieves higher prediction accuracy than the other algorithms on all datasets except the SensIT Vehicle (acoustic). For the SensIT Vehicle (acoustic), the accuracy of the SR-LSSVM is slightly lower than that of the FS-LSSVM method, but the run time of the FS-LSSVM is longer than that of the SR-LSSVM. Furthermore, it is obvious that the C-SVC is the slowest algorithm among these algorithms, and the size of its support vectors is much larger than other algorithms. ProCRC is the fastest algorithm among the compared approaches, but its accuracy is lower than the SR-LSSVM. Based on the above analyses, we conclude that our proposed algorithm (SR-LSSVM) is more suitable for large-scale classification problems, as it has higher test accuracy, requires less time, and produces sparser solutions than the other methods.

4.1.5. Robustness comparisons for large-scale dataset

In order to compare the robustness of the ProCRC, PCP-LSSVM, FS-LSSVM, CSI and SR-LSSVM algorithms for large-scale datasets, we set the rates of the outliers at 0%, 5% and 10%, respectively, on the Cod-RNA dataset. Table 3 illustrates the results. From Table 3, it is easy to observe that the SR-LSSVM has higher accuracy than other approaches, and the accuracy of SR-LSSVM declines slower than other algorithms with the increasing of the rates of outliers. Therefore, the SR-LSSVM is more robust to outliers than other algorithms.

4.2. Regression experiments

In the regression experiments, we conducted experiments on medium-scale and large-scale benchmark regression datasets. We adopt the popular regression estimation criterion RMSE (root mean square error) to measure the efficiency of algorithms. The Mg, Abalone, and Cadata datasets were downloaded from the LIBSVM [41], and the Winequality, Tic, Relation N D, Slice and 3D-spatial datasets were downloaded from the UCI database [44]. Each attribute of the samples was normalized into $[-1,\ 1]$. For every dataset, we randomly selected 2/3 of the samples as the training set and the rest of the samples as the test set. To test the insensitiveness of our proposed algorithm to outliers, we randomly selected 1/10 training samples, and added Gaussian noise $v_i \sim N(0, d^2)$ on their targets to simulate outliers. We set d as half

Table 5
Comparison of different algorithms on large-scale benchmark regression datasets with outliers (10%). The standard deviations are given in brackets. 'nSVs' refers to average number of support vectors. m and n are the numbers of training and testing samples respectively, l is the dimension of data.

Datasets	Algorithms	Training time(s)	nSVs	RMSE
Cadata	ϵ-SVR	7.44(0.12)	8462.8(58.4)	0.2779(0.007)
$m = 13,760$	FS-LSSVM	1.61(0.03)	300(0.9)	0.2383(0.004)
$n = 6,880$	CSI	6.03(1.59)	**159**(36.9)	0.2463(0.006)
$l = 8$	PCP-LSSVM	**1.02**(0.01)	302(0)	0.2383(0.005)
	SR-LSSVM	1.05(0.01)	302(0)	**0.2382**(0.002)
Relation N D	ϵ-SVR	112.6(2.1)	24,556(82.4)	1.31(0.076)
$m = 35,608$	FS-LSSVM	2.66(0.03)	173.8(1.9)	1.55(0.060)
$n = 17,805$	CSI	9.44(1.35)	**107.4**(21.9)	1.78(0.086)
$l = 24$	PCP-LSSVM	**1.58**(0.01)	178(0)	1.41(0.058)
	SR-LSSVM	1.76(0.09)	178(0)	**1.27**(0.028)
Slice	ϵ-SVR	5088.77(23.2)	35,627.4(3.7)	17.44(0.100)
$m = 35,666$	FS-LSSVM	65.78(0.23)	594.8(0.84)	23.80(0.771)
$n = 17,834$	CSI	161.39(56.5)	**52**(22.6)	12.92(0.364)
$l = 386$	PCP-LSSVM	**42.96**(0.25)	600(0)	12.81(0.369)
	SR-LSSVM	43.08(0.27)	600(0)	**8.84(0.054)**
3D-spatial	ϵ-SVR	6576.82(46.0)	276,625(234.8)	0.49(0.006)
$m = 289,916$	FS-LSSVM	62.51(0.45)	399.4(0.5)	**0.45**(0.001)
$n = 144,958$	CSI	132.3(13.00)	**173.3**(41.7)	0.46(0.001)
$l = 3$	PCP-LSSVM	**36.41**(0.13)	400(0)	**0.45**(0.001)
	SR-LSSVM	37.09(0.35)	400(0)	**0.45**(0.001)
Sinc	ϵ-SVR[1]	–	–	–
$m = 1,000,000$	FS-LSSVM	17.82(0.72)	31(0)	**0.748**(0.00)
$n = 600,000$	CSI	305.10(26.4)	**10**(0)	0.752(0.00)
$l = 1$	PCP-LSSVM	**4.87**(0.48)	26.4(0.5)	**0.748**(0.00)
	SR-LSSVM	5.89(0.43)	26(0)	**0.748**(0.00)

[1] The training time is too long.

of the mean of the targets for each dataset. We set the smoothing parameter $p = 10^4$ in the SR-LSSVM and the stop criterion $\varepsilon = 10^{-2}$, and in ϵ-SVR, $\epsilon = 0.01$.

4.2.1. Medium-scale benchmark regression datasets experiments

Table 4 reports the experimental results for the medium-scale datasets. The optimal parameters are listed in Table A3. We set $r = 0.05\,m$ for the SR-LSSVM and FS-LSSVM algorithms for every dataset except Tic ($r = 400$). All the algorithms independently operated 10 times to obtain unbiased results.

From Table 4, it is clear that the proposed approach (SR-LSSVM) has lower prediction error than any other approaches for all datasets. Moreover, we can also observe that the SR-LSSVM is the fastest method, and that the standard deviations of running time are the smallest in most cases. The sizes of the support vectors of the SR-LSSVM and FS-LSSVM are much smaller than other methods, and their standard deviations are quite small as well. This means that these two methods are sparseness and the sizes of support vectors are stable. However, the FS-LSSVM has worse predicted performance (RMSE) than our method and is more time-consuming. Overall, the predicted error of our SR-LSSVM algorithm are smaller than that of other algorithms, and our method dramatically reduces training time because of its sparseness. Therefore, it is a good choice for regression problems.

4.2.2. Large-scale benchmark regression datasets experiments

We compare our SR-LSSVM with several sparse algorithms, including the ϵ-SVR, FS-LSSVM, PCP-LSSVM and CSI on the large-scale regression datasets.

The Sinc dataset is an artificial dataset that contains 1,600,000 samples. They were generated by the Sinc function with Gaussian noise $y = \frac{sin(3x)}{3x} + \varsigma$, where $x \in [-9, 9]$, $\varsigma \sim N(0,\ 0.25)$, and $N(0,\ 0.25)$ represents normal distribution with zero means and variance 0.25. In order to simulate outliers, we randomly chose 1/5 samples and added larger Gaussian noise $N(0,\ 1)$ to their targets.

Table 5 reports experimental results for the large-scale regression datasets. The optimal parameters are listed in Table A4. All the algorithms operated 5 times independently to obtain unbiased results for each dataset.

From Table 5, It is clear that the prediction errors of our approach are smaller than those of the other compared algorithms for all datasets. With respect to the running time, the SR-LSSVM method is faster than the CSI, FS-LSSVM and ϵ-SVR on all of datasets, and ϵ-SVR is the slowest. For example, for the 3D-spatial dataset, ϵ-SVR spends about 6577 seconds for training, whereas our method SR-LSSVM only needs approximately 34 s and obtains smaller value of RMSE. The PCP-LSSVM is the fastest algorithm, but its prediction error is larger than the SR-LSSVM on some datasets such as Slice and Relation N D. As regard to the number of support vectors, ϵ-SVR has the most, whereas the support vector size of the SR-LSSVM is quite small. Although support vector size of the CSI is the smallest, the prediction error of it is larger than that of the SR-LSSVM, and CSI is slower than our method. From Table 5, it can be seen that our proposed approach is more suitable for training large-scale regression datasets, as it is quite fast and the prediction performance is good.

5. Conclusion

The R-LSSVM model is robust for classification and regression problems, which is interpreted from a re-weighted viewpoint in this paper. However, the main disadvantage of the model is that the existing algorithms for the R-LSSVM lose sparseness, therefore they cannot be used for large-scale problems. To overcome this disadvantage, we utilize the representer theory and the incomplete pivoting Cholesky factorization technique to obtain a sparse solution of R-LSSVM, and propose an effective algorithm called the SR-LSSVM. The experimental results indicate that our algorithm not only has sparseness and robustness, but also has better or comparable prediction performance than the other algorithms. Furthermore, the training speed of the proposed algorithm is faster than the R-LSSVM, LSSVM, W-LSSVM, FS-LSSVM, CSI and SVMs (C-SVC for classification and ϵ-SVR for regression). Therefore, the SR-

LSSVM is a suitable option for dealing with large-scale classification and regression problems.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (NNSFC) under the grant number 61772020 and 71301067, and the Fundamental Research Funds for the Central Universities under the grant number JB150718.

Appendix A. Parameter settings

Tables A1 and A2 list the parameter settings of all the compared algorithms used on the classification datasets in Tables 1 and 2 respectively.

Tables A3 and A4 give the parameter settings for all the compared algorithms used on the regression datasets in Tables 4 and 5 respectively.

In Tables A1 and A3, LSSVMs include LSSVM, W-LSSVM, FS-LSSVM, R-LSSVM and SR-LSSVM algorithms. R-LSSVMs include R-LSSVM and SR-LSSVM algorithms. In Tables A2 and A4, LSSVMs include FS-LSSVM, CSI, PCP-LSSVM and SR-LSSVM algorithms.

Table A1
Parameter settings of algorithms for medium-scale classification datasets.

	ProCRC		C-SVC		LSSVMs		R-LSSVMs	
Data	λ	μ	C	σ	$m\lambda$	σ	τ	h
Splice	10^{-2}	$10^{-0.5}$	10^{0}	2^{-9}	10^{-2}	2^{-12}	0.9	0.5
Pendigits	10^{-2}	10^{-4}	10^{2}	2^{-4}	10^{-3}	2^{-4}	1.5	0.25
Satimage	10^{-2}	10^{-7}	10^{0}	2^{-1}	10^{0}	2^{-1}	0.5	0.3
USPS	$10^{-1.5}$	10^{-6}	10^{0}	2^{-7}	10^{0}	2^{-7}	1.1	0.15
Mushrooms	10^{-2}	10^{-3}	10^{0}	2^{-3}	10^{-1}	2^{-3}	0.6	0.3
Protein	10^{-2}	10^{-7}	10^{0}	2^{-5}	10^{0}	2^{-5}	0.8	0.7

Table A2
Parameter settings of algorithms for large-scale classification datasets.

	ProCRC		C-SVC		LSSVMs		SR-LSSVM
Data	λ	μ	C	σ	$m\lambda$	σ	τ
IJCNN1	10^{-2}	10^{-3}	10^{0}	2^{-4}	10^{-3}	2^{-8}	2.5
Cod-RNA	$10^{-1.3}$	10^{-9}	10^{-2}	2^{-6}	10^{-4}	2^{-1}	1.8
SensIT Vehicle (acoustic)	10^{-2}	10^{-6}	10^{-1}	2^{-4}	10^{-6}	2^{-4}	1.7
Skin-nonskin	10^{-2}	10^{-9}	10^{0}	2^{-2}	10^{-3}	2^{-10}	1.5
Checkerboard(1M)	10^{-2}	10^{-2}	–	–	10^{-3}	2^{6}	0.08

Table A3
Parameter settings of algorithms for medium-scale regression datasets.

	ϵ-SVR		LSSVMs		R-LSSVMs	
Data	C	σ	$m\lambda$	σ	τ	h
Mg	10^{0}	2^{-3}	10^{-2}	2^{1}	0.9	0.025
Winequality	10^{0}	2^{-4}	10^{-1}	2^{-6}	1	0.175
Abalone	10^{2}	2^{-5}	10^{-4}	2^{-4}	0.01	0.15
Tic	10^{0}	2^{-4}	10^{0}	2^{-6}	1	0.025

Table A4
Parameter settings of algorithms for large-scale regression datasets.

	ϵ-SVR		LSSVMs		SR-LSSVM
Data	C	σ	$m\lambda$	σ	τ
Cadata	10^{0}	2^{-1}	10^{-3}	2^{-1}	1.5
Relation N D	10^{1}	2^{-2}	10^{-2}	2^{-4}	13.7
Slice	10^{2}	2^{-6}	10^{0}	2^{-11}	40
3D-Spatial	10^{0}	2^{-1}	10^{-3}	2^{-3}	1.6
Sinc	–	–	10^{-2}	2^{-2}	0.6

Appendix B. Computational complexity reductions for the ProCRC

ProCRC (Probabilistic Collaborative Representation based classification) [16] is an improved method of CRC (Collaborative Representation based classification) [17]. They are both non-parametric methods based on least squares. The models of CRC and ProCRC are simple and they can easily solve multi-class classification problems. Therefore, they have been successfully used in many areas, such as face recognition and other visual recognition tasks [18,45–47]. However, the computational complexity of the code on authors' website[5] for the CRC and ProCRC is high. In this section, we show that the computational complexity of the ProCRC can be reduced by Sherman–Morrison–Woodbury (SMW) formula [36,37]. We only discuss the two-class classification problem here, and the method can be generalized to multi-class case easily.

Let $X=[X_1;X_2]\in\Re^{m\times l}$, where $l<m$, X_k is the dataset of the kth class, and each row of X_k is a sample of class k. The test set is denoted by $\widetilde{X}=\{\widetilde{\mathbf{x}}_i\}_{i=1}^{n}$ with $\widetilde{\mathbf{x}}_i\in\Re^{l}$ and $l<n$. We denote I as an identity matrix with appropriate size.

For the CRC, denote $\hat{\alpha}_i=[(XX^\top+\lambda\cdot I)^{-1}X]\widetilde{\mathbf{x}}_i^\top$. Then the test sample $\widetilde{\mathbf{x}}_i$ $(i=1,2,\ldots,n)$ is predicted as

$$\widetilde{y}_i=\arg\min_k \ \|\widetilde{\mathbf{x}}_i^\top-X_k^\top\cdot\hat{\alpha}_i^k\|_2^2/\|\hat{\alpha}_i^k\|_2^2. \tag{B.1}$$

We can use the SMW formula to reduce the computational complexity of $\hat{\alpha}_i$.

Recently in [16], the ProCRC is proposed as an improvement of the CRC. The key step of the ProCRC is to compute

$$\hat{\alpha}_i=T\widetilde{\mathbf{x}}_i^\top, \tag{B.2}$$

where T is the projection matrix:

$$T=\left(XX^\top+\begin{bmatrix}\lambda I+\mu X_1X_1^\top & 0\\ 0 & \lambda I+\mu X_2X_2^\top\end{bmatrix}\right)^{-1}X, \tag{B.3}$$

λ and μ are two regularization parameters. Then the class label of the test sample $\widetilde{\mathbf{x}}_i$ $(i=1,2,\ldots,n)$ is predicted as

$$\widetilde{y}_i=\arg\min_k \ \|X^\top\hat{\alpha}_i-X_k^\top\hat{\alpha}_i^k\|_2^2. \tag{B.4}$$

According to the Matlab code on authors' website[5], the computational complexity of calculating T is $O(m^3)$ (mainly for calculating the inverse of a $m\times m$ matrix), and the computational complexity of predicting targets of n test samples is $O(mnl)$ (mainly for calculating $\hat{\alpha}_i^k$ for $i=1,2,\ldots,n$). When the number of samples, m and n, are large, the computational complexity is high.

Actually, if denote $C=\begin{bmatrix}C_1 & 0\\ 0 & C_2\end{bmatrix}$ with $C_k=\lambda I+\mu X_kX_k^\top$, then (B.3) can be calculated as

$$T=[C^{-1}-C^{-1}X(I+X^\top C^{-1}X)^{-1}X^\top C^{-1}]X$$

by SMW formula, where $C^{-1}=\begin{bmatrix}C_1^{-1} & 0\\ 0 & C_2^{-1}\end{bmatrix}$ with $C_k^{-1}=\frac{1}{\lambda}(I-X_k(\frac{\lambda}{\mu}I+X_k^\top X_k)^{-1}X_k^\top)$. Let $P=C^{-1}X$, then

$$T=P-P(I+X^\top P)^{-1}X^\top P. \tag{B.5}$$

By (B.5), the computational complexity of calculating T is reduced to $O(l^2m)$.

[5] Codes are available in http://www4.comp.polyu.edu.hk/~cslzhang/papers.htm.

In order to reduce the computational complexity in predicting stage, putting (B.2) into (B.4), we obtain

$$\widetilde{y}_i = \arg\min_k \ \|(X^\top T - X_k^\top T_k)\widetilde{\mathbf{x}}_i^\top\|_2^2, \tag{B.6}$$

where T_k is a sub-matrix of T with rows associated with X_k and $T = [T_1 \quad T_2]$. The computational complexity for predicting samples in the whole test set by (B.6) is $O((m+n)l^2)$, where $O(ml^2)$ is for calculating $Q_k = (X^\top T - X_k^\top T_k) \in \Re^{l\times l}$, and $O(nl^2)$ is for calculating $Q_k\widetilde{\mathbf{x}}_i^\top$ for $i = 1, 2, \ldots, n$.

However, the computational complexity of prediction rule (B.1) for the CRC cannot be reduced by this approach, because we must calculate $\hat{\alpha}_i$ explicitly in (B.1).

References

[1] J.A.K. Suykens, J. Vandewalle, Least squares support vector machine classifiers, Neural Process. Lett. 9 (3) (1999) 293–300.
[2] D. Calisir, E. Dogantekin, A new intelligent hepatitis diagnosis system: PCA-LSSVM, Expert Syst. Appl. 38 (8) (2011) 10705–10708.
[3] B. Long, W. Xian, M. Li, H. Wang, Improved diagnostics for the incipient faults in analog circuits using LSSVM based on PSO algorithm with Mahalanobis distance, Neurocomputing 133 (10) (2014) 237–248.
[4] L. Yang, S. Yang, S. Li, R. Zhang, F. Liu, L. Jiao, Coupled compressed sensing inspired sparse spatial-spectral LSSVM for hyperspectral image classification, Knowl. Based Syst. 79 (2015) 80–89.
[5] S. Mehrkanoon, J.A. Suykens, Learning solutions to partial differential equations using LS-SVM, Neurocomputing 159 (2) (2015) 105–116.
[6] Y. Gao, X. Shan, Z. Hu, D. Wang, Y. Li, X. Tian, Extended compressed tracking via random projection based on MSERs and online LS-SVM learning, Pattern Recognit. 59 (2016) 245–254.
[7] J.A.K. Suykens, T.V. Gestel, J.D. Brabanter, B.D. Moor, J. Vandewalle, Least Squares Support Vector Machines, World Scientific, Singapore, 2002.
[8] J. Valyon, G. Horvth, A weighted generalized LS-SVM, Period. Polytech. Ser. Electr. Eng. 47 (3–4) (2003) 229–251.
[9] L. You, L. Jizhen, Q. Yaxin, A new robust least squares support vector machine for regression with outliers, Adv. Control Eng. Inf. Sci. 15 (2011) 1355–1360.
[10] K. Wang, P. Zhong, Robust non-convex least squares loss function for regression with outliers, Knowl. Based Syst. 71 (2014) 290–302.
[11] X. Yang, L. Tan, L. He, A robust least squares support vector machines for regression and classification with noise, Neurocomputing 140 (2014) 41–52.
[12] J.A.K. Suykens, L. Lukas, J. Vandewalle, Sparse approximation using least squares support vector machines, in: Proceedings of the IEEE International Symposium on Circuits and Systems, Geneva, Switzerland, 2000, pp. 757–760.
[13] J. Suykens, J.D. Brabanter, L. Lukas, J. Vandewalle, Weighted least squares support vector machines: robustness and sparse approximation, Neurocomputing 48 (2002) 85–105.
[14] L. Jiao, L. Bo, L. Wang, Fast sparse approximation for least squares support vector machines, IEEE Trans. Neural Netw. 18 (3) (2007) 685–697.
[15] S. Zhou, Sparse LSSVM in primal using Cholesky factorization for large scale problems, IEEE Trans. Netw. Learn. Syst. 27 (4) (2016) 783–795.
[16] S. Cai, L. Zhang, W. Zuo, X. Feng, A probabilistic collaborative representation based approach for pattern classification, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2950–2959.
[17] L. Zhang, M. Yang, X. Feng, Sparse representation or collaborative representation: which helps face recognition? in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2011, pp. 471–478.
[18] Y. Chi, F. Porikli, Connecting the dots in multi-class classification: from nearest subspace to collaborative representation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012, pp. 3602–3609.
[19] B. Schölkopf, R. Herbrich, A.J. Smola, A generalized representer theorem, in: Proceedings of the Annual Conference on Computational Learning Theory, Springer, Amsterdam, Netherlands, 2001, pp. 416–426.
[20] S. Shalev-Shwartz, S. Ben-David, Understanding Machine Learning-from Theory to Algorithms, Cambridge, 2014.
[21] D. Geman, C. Yang, Nonlinear image recovery with half-quadratic regularization, IEEE Trans. Image Process. 4 (7) (1995) 932–946.
[22] M. Nikolova, M.K. Ng, Analysis of half-quadratic minimization methods for signal and image recovery, SIAM J. Sci. Comput. 27 (3) (2005) 937–966.
[23] K.D. Brabanter, K. Pelckmans, J.D. Brabanter, M. Debruyne, J. Suykens, M. Hubert, B. DeMoor, Robustness of kernel based regression: a comparison of iterative weighting schemes, in: Proceeding of the Nineteenth International Conference on Artificial Neural Networks (ICANN), Limassol, Cyprus, 2009, pp. 100–110.
[24] B. He, X. Yuan, On the O(1/n) convergence rate of the Douglas–Rachford alternating direction method, SIAM J. Numer. Anal. 50 (2) (2012) 700–709.
[25] Y. Feng, Y. Yang, X. Huang, S. Mehrkanoon, J.A.K. Suykens, Robust support vector machines for classification with nonconvex and smooth losses, Neural Comput. 28 (2016) 1217–1247.
[26] A. Yuille, A. Rangarajia, The concave–convex procedure, Neural Comput. 15 (4) (2003) 915–936.
[27] S. Zhou, J. Cui, F. Ye, H. Liu, Q. Zhu, New smoothing SVM algorithm with tight error bound and efficient reduced techniques, Comput. Optim. Appl. 56 (3) (2013) 599–617.
[28] S. Zhou, Y. Zheng, X. Mu, K2DPCA methods for face recognition based on Cholesky decomposition, Syst. Eng. Theory Pract. 36 (2) (2016) 528–535.
[29] C. Williams, M. Seeger, Using the Nyström method to speed up kernel machines, in: Advances in Neural Information Processing Systems, 13, MIT Press, 2001, pp. 682–688.
[30] P. Drineas, M.W. Mahoney, On the Nyström method for approximating a Gram matrix for improved kernel-based learning, J. Mach. Learn. Res. 6 (Dec) (2005) 2153–2175.
[31] K. Zhang, J.T. Kwok, Clustered Nyström method for large scale manifold learning and dimension reduction, IEEE Trans. Neural Netw. 21 (10) (2010) 1576–1587.
[32] S. Si, C.J. Hsieh, I.S. Dhillon, Computationally efficient Nyström approximation using fast transforms, in: Proceedings of the International Conference on Machine Learning (ICML), 2016, pp. 2655–2663.
[33] T.P. Dinh, H.M. Le, H.A.L. Thi, F. Lauer, A difference of convex functions algorithm for switched linear regression, IEEE Trans. Autom. Control 59 (8) (2014) 2277–2282.
[34] B.K. Sriperumbudur, G.R.G. Lanckriet, A proof of convergence of the Concave–Convex Procedure using Zangwills theory, Neural Comput. 24 (6) (2012) 1391–1407.
[35] P.D. Tao, L.T.H. An, Convex analysis approach to D. C. programming: theory, algorithms and applications, Acta Math. Vietnamica 22 (1) (1997) 289–355.
[36] J. Sherman, W.J. Morrison, Adjustment of an inverse matrix corresponding to a change in one element of a given matrix, Annals Math. Stat. 21 (1) (1950) 124–127.
[37] M.A. Woodbury, Inverting modified matrices, Memo. Rep. 42 (106) (1950) 336.
[38] N. Deng, Y. Tian, Support Vector Machine: Theory, Algorithm and Prolongation, Science Press, Beijing, 2009.
[39] K.D. Brabanter, P. Karsmakers, F. Ojeda, C. Alzate, J.D. Brabanter, K. Pelckmans, B.D. Moor, J. Vandewalle, J. Suykens, 2011, http://www.esat.kuleuven.be/sista/lssvmlab/.
[40] F.R. Bach, M.I. Jordan, Predictive low-rank decomposition for kernel methods, in: Proceedings of the Twenty-Second International Conference on Machine Learning, Bonn, Germany, 2005, pp. 33–40.
[41] C.-C. Chang, C.-J. Lin, 2011, LIBSVM: a library for support vector machines. http://www.csie.ntu.edu.tw/~cjlin/libsvm.
[42] T.K. Ho, E.M. Kleinberg, Building projectable classifiers of arbitrary complexity, in: Proceedings of the Thirteenth International Conference on Pattern Recognition, IEEE, vol. 2, 1996, pp. 880–885.
[43] O.L. Mangasarian, D.R. Musicant, Successive overrelaxation for support vector machines, IEEE Trans. Neural Netw. 10 (5) (1999) 1032–1037.
[44] M. Lichman, 2013, UCI machine learning repository http://archive.ics.uci.edu/ml.
[45] Y. Chi, F. Porikli, Classification and boosting with multiple collaborative representations, IEEE Trans. Pattern Anal. Mach. Intell. 36 (8) (2014) 1519–1531.
[46] H. Zhang, J. Yang, Y. Zhang, N.M. Nasrabadi, T.S. Huang, Close the loop: joint blind image restoration and recognition with sparse representation prior, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2011, pp. 770–777.
[47] X. Jiang, J. Lai, Sparse and dense hybrid representation via dictionary decomposition for face recognition, IEEE Trans. Pattern Anal. Mach. Intell. 37 (5) (2015) 1067–1079.

Li Chen was born in Henan, China, in 1982. She received the M.S. degree in mathematics from China Agricultural University, Beijing, China, in 2009. She is currently working towards the Ph.D. degree in School of Mathematics and Statistics, Xidian University. Her current research interests include optimization algorithm and its application, machine learning, pattern recognition, and support vector machines.

Shuisheng Zhou was born in Shaanxi, China, in 1972. He received the M.S. degree in applied mathematics and the Ph.D. degree in computer science from Xidian University, Xi'an, China, in 1998 and 2005, respectively. He is currently a Professor in the School of Mathematics and Statistics, Xidian University. His current research interests include optimization algorithm and its application, machine learning, pattern recognition, kernel-based learning, and support vector machines.