Noniterative Sparse LS-SVM Based on Globally Representative Point Selection

Yuefeng Ma, Xun Liang, *Senior Member, IEEE*, Gang Sheng, James T. Kwok, *Fellow, IEEE*, Maoli Wang, and Guangshun Li

***Abstract*— A least squares support vector machine (LS-SVM) offers performance comparable to that of SVMs for classification and regression. The main limitation of LS-SVM is that it lacks sparsity compared with SVMs, making LS-SVM unsuitable for handling large-scale data due to computation and memory costs. To obtain sparse LS-SVM, several pruning methods based on an iterative strategy were recently proposed but did not consider the quantity constraint on the number of reserved support vectors, as widely used in real-life applications. In this article, a noniterative algorithm is proposed based on the selection of globally representative points (global-representation-based sparse least squares support vector machine, GRS-LSSVM) to improve the performance of sparse LS-SVM. For the first time, we present a model of sparse LS-SVM with a quantity constraint. In solving the optimal solution of the model, we find that using globally representative points to construct the reserved support vector set produces a better solution than other methods. We design an indicator based on point density and point dispersion to evaluate the global representation of points in feature space. Using the indicator, the top globally representative points are selected in one step from all points to construct the reserved support vector set of sparse LS-SVM. After obtaining the set, the decision hyperplane of sparse LS-SVM is directly computed using an algebraic formula. This algorithm only consumes *O(N2)* in computational complexity and *O(N)* in memory cost which makes it suitable for large-scale data sets. The experimental results show that the proposed algorithm has higher sparsity, greater stability, and lower computational complexity than the traditional iterative algorithms.**

***Index Terms*— Globally representative point, noniterative algorithm, pruning method, quantity constraint, sparse least squares support vector machine (LS-SVM).**

I. Introduction

CLASSIFICATION is an important domain in machine learning with the objective of obtaining an efficient classifier [1]. To achieve this objective, numerous methods are applied that are based on different approaches such as Bayes method, decision trees, neural networks, and so on. Among these methods, support vector machines (SVMs) have attracted much attention due to high efficiency [2]. In SVM, a hyperplane is constructed as a classifier with support vectors in Hilbert space and is obtained using a mapping function. The SVM is highly advantageous because it is sparse, efficient, and highly accurate in classification [1]–[3].

As an important variant, the least squares support vector machine (LS-SVM) can be obtained by replacing the loss function in SVM with a square function [4]. Thus, LS-SVM can be treated as a linear system, which means that the process of solving LS-SVM is simple and efficient. In addition, the generalization ability of LS-SVM was demonstrated as equal to that of SVM in experiments and applications [5]. However, the limitation of LS-SVM is also apparent compared with SVM. Because LS-SVM includes almost all of the training data as support vectors, LS-SVM loses the property of sparseness that is held by SVMs, which means that the computational and memory cost rapidly increases with the increasing number of support vectors when LS-SVM is used to identify an unlabeled datum. Therefore, pruning of the number of support vectors of LS-SVM is a significant challenge [6]–[8].

In the research on sparse LS-SVM, the main approach is based on an iterative pruning method. Suykens *et al.* [6] proposed an iterative algorithm in which several support vectors are pruned in each iteration. Considering the general ability and optimization of LS-SVM, a comprehensive method was proposed to obtain sparse LS-SVM [7]. Using the criterion of selecting the support vector with the least deviation, the accuracy of sparse LS-SVM was improved [8]. A light variant of sparse LS-SVM was proposed to improve the performance of the pruning algorithm [9]. In each iteration, the point with the minimum influence on the dual optimization problem was selected for pruning [10]. In a pruning algorithm that included two steps, an indicator of fragmentization was introduced to evaluate the influence of points in LS-SVM [11]. A modified active subset selection method based on quadratic Rényi entropy and fast cross-validation for fixed-size LS-SVMs was proposed for classification and regression in an optimized tuning process [12]. With the iterative build-up of a conjugate set of vectors of increasing cardinality, sparse conjugate directions pursuit was proposed to obtain sparse LS-SVM by solving a small linear subsystem in each iteration [13]. Based on an iterative approximation to the l_0-norm, LS-SVM was adapted to a classical SVM classifier [14]. By importing an l_0-norm regularization term

Manuscript received March 20, 2019; revised October 12, 2019 and January 26, 2020; accepted March 4, 2020. Date of publication April 7, 2020; date of current version February 4, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 71531012 and Grant 71601013 and in part by the Natural Science Foundation of Beijing under Grant 4172032. *(Corresponding author: Xun Liang.)*

Yuefeng Ma, Xun Liang, Maoli Wang, and Guangshun Li are with the School of Computer, Qufu Normal University, Jining 276826, China (e-mail: rzmyf1976@ruc.edu.cn; xliang@ruc.edu.cn; wangml@qfnu.edu.cn; guangshunli@qfnu.edu.cn).

Gang Sheng is with the School of Information Engineering, Yancheng Teachers University, Yancheng 224007, China (e-mail: shenggang@neusoft.edu.cn).

James T. Kwok is with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong (e-mail: jamesk@cse.ust.hk).

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

Digital Object Identifier 10.1109/TNNLS.2020.2979466

2162-237X © 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Qufu Normal University. Downloaded on February 06,2021 at 08:09:25 UTC from IEEE Xplore. Restrictions apply.

of parameters into the primal optimization problem, sparse support vector classification (SSVC) was proposed and is iteratively trained on the training set until it converges to a highly sparse solution [15]. Combining prior knowledge and an adaptive learning process, a weighted l_q-norm adaptive LS-SVM model classifier was introduced that chooses q according to the structure of the data set [16], [17]. A fast sparse approximation scheme for LS-SVM (FSALS-SVM) was presented that iteratively builds the decision function by adding one basis function from a kernel-based dictionary at one iteration until the insensitive criterion is satisfied [18]. Combining a reduced technique with an iterative strategy, Zhao and Sun [19] proposed a Recursive Reduced Least Squares Support Sector Regression (RR-LSSVR) in which the support vector should be selected by considering its contribution to the target function, and all constraints are generated based on the entire training set. Using increasing and decreasing learning procedures, an adaptive pruning algorithm based on a bottom-to-top strategy was presented in which a small support vector set covering most of the information in the training set can be formed adaptively [20]. An improved RR-LSSVR was developed using a pruning criterion in which the point leading to the largest reduction in the target function is selected for pruning [21]. Compressive sampling was applied to find the sparse support vector set of LS-SVM to obtain the sparse LS-SVM [22], [23]. A least-squares regression estimator was developed using bias-variance analysis in compressed spaces [24]. Liang *et al.* [25] proposed a single iterative method to remove superfluous support vectors (SVs) based on kernel row vector space. This method can be applied to obtain sparse LS-SVM with a simple modification. The correct number of initial prototype vectors as the final support vectors is introduced using the sparsity-error tradeoff method to obtain sparse LS-SVM [26]. Shi *et al.* [27] presented an iterative method for pruning redundant support vectors in which a confidence interval is introduced to select support vectors for pruning in LS-SVM in each iteration. Combining an iterative pruning method and primal fixed-size LSSVM (PFS-LSSVM) model, a pruning algorithm for highly sparse LS-SVM was proposed for large-scale data [28]. Based on partial pivoting Cholesky factorization of the kernel matrix, full pivoting Cholesky factorization of primal LSSVM (PCP-LSSVM) was extended with an iterative strategy [29]. By solving the least absolute shrinkage and selection operator problem, a pruning method of least angle regression was used to obtain sparse LS-SVM in which the most important points should be iteratively selected as support vectors [30]. Using the density clustering method, reconstructed support vectors were selected to obtain sparse LS-SVM without repeatedly training the LS-SVM [31]. A sparse LS-SVM was proposed in reduced empirical feature space, and a wrapper method known as block addition was used to decrease the size of the kernel matrix in LS-SVM [32]. Based on the training mean square error and sensitivity measure, a localized generalization error model was introduced to prune the support vectors in LS-SVM [33]. With the two objectives of maximizing the accuracy of classifiers and minimizing the number of reserved support vectors, Silva and Neto [34] presented a pruning method to obtain sparse LS-SVM with a multiobjective genetic algorithm in which outliers and nonrelevant points were iteratively reduced. By approximating the kernel matrix using a low-rank matrix and smoothing the loss function using the entropy penalty function, a convergent, sparse, robust LS-SVM was proposed in primal space [35]. Obviously, most of the pruning methods are iterative, leading to highly expensive computation and memory cost. Otherwise, the criteria used to select the support vector to be pruned were based mainly on the relation between the support vector and the decision hyperplane. This approach implies that the decision hyperplane must be obtained beforehand with high computation and memory cost. Finally, in most real applications, the number of reserved support vectors in sparse LS-SVM is limited, which has not been considered in most pruning methods. Therefore, the urgent need exists to develop a noniterative pruning method to obtain sparse LS-SVM with quantity constraint on the number of reserved support vectors. To address these situations, we propose a noniterative pruning method designed to obtain sparse LS-SVM using the most representative points as the reserved support vectors.

Our contributions are described as follows. First, we demonstrate an optimization model of sparse LS-SVM with a quantity constraint on the number of reserved support vectors. Second, we propose an indicator of the global representation of points by considering point density and point dispersion in feature space. Finally, based on the model and the indicator, we present a noniterative pruning method designed to obtain sparse LS-SVM.

In this article, we describe the fundamental process of solving LS-SVM in Section II. Next, the global-representation-based sparse least squares support vector machine (GRS-LSSVM) is proposed in Section III. In Section IV, we demonstrate the effectiveness of GRS-LSSVM in experiments on several data sets. In Section V, we present the conclusions.

II. Process of Solving LS-SVM

LS-SVM can be obtained from SVM by replacing its loss function and constraints. The task of LS-SVM is to obtain a decision hyperplane described as follows:

$$f(x) = w^{\mathrm{T}}\varphi(x) + b \tag{1}$$

where $\varphi(\cdot)$ is a map function from data space R^{d} to feature space H, w is a vector in feature space, $b \in R$, and w^{T} is the transpose of w. Let (x_i, y_i), $i = 1, \ldots, N$, be a pattern, where $x_i \in R^{\mathrm{d}}$ is a point, with $y_i \in \{-1, +1\}$ for classification and $y_i \in R$ for regression. Using $X = \{x_i\}$ and $Y = \{y_i\}$, (1) can be obtained by solving an optimization problem in the LS-SVM formulation as follows:

$$\begin{aligned} &\min_{w,b,e_i} \; J = \frac{1}{2}\|w\|^2 + \frac{C}{2}\sum_{i=1}^{N} e_i^2 \\ &\text{s.t. } w^{\mathrm{T}}\varphi(x_i) + b = y_i - e_i, \quad i = 1, \ldots, N \end{aligned} \tag{2}$$

where $C \in R^+$ is a regularization parameter.

In practice, by constructing the Lagrangian function of (2), we can obtain the following formulas based on the

Authorized licensed use limited to: Qufu Normal University. Downloaded on February 06,2021 at 08:09:25 UTC from IEEE Xplore. Restrictions apply.

Karush–Kuhn–Tucker (KKT) condition:

$$w = \sum_{i=1}^{N} \alpha_i \varphi(x_i) \tag{3}$$

$$\begin{bmatrix} K(X,X) + I/C & \mathbf{1} \\ \mathbf{1}^{\mathrm{T}} & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ b \end{bmatrix} = \begin{bmatrix} Y \\ 0 \end{bmatrix} \tag{4}$$

where $K(\Omega, \Lambda) = [k(x_i, x_j)]_{|\Omega|\times|\Lambda|}$, $x_i \in \Omega$, $x_j \in \Lambda$, $k(\cdot,\cdot)$ is a kernel function, $\mathbf{1} = (1, \ldots, 1)^{N\times 1}$, and I is a unit matrix.

From (4), we observe that the coefficients of (1) can be obtained by solving a linear system. Obviously, the process of solving LS-SVM is simpler than that of SVM. Because almost all of the coefficients are nonzero, the decision hyperplane of LS-SVM includes all the data in the training data set, which means that LS-SVM loses sparseness. The pruning method for sparse LS-SVM is intended to obtain a sparse decision hyperplane that only includes a few points.

III. GRS-LSSVM

A. Model of Sparse LS-SVM With Quantity Constraint

The goal of sparse LS-SVM is to maintain the performance of $f(x)$ using the following decision hyperplane:

$$g(x) = \upsilon^{\mathrm{T}} \varphi(x) + b \tag{5}$$

where $\upsilon = \sum_{i=1}^{L} \beta_i \varphi(s_i)$, $s_i \in X$, $L < N$, and L is the given number of reserved support vectors of sparse LS-SVM.

Let $S = \{s_1, \ldots, s_L\}$ be the set of reserved support vectors in sparse LS-SVM. Then, the optimization problem of sparse LS-SVM with the quantity constraint on the number of reserved support vectors can be written as follows:

$$\begin{aligned} &\min_S J_g = \frac{1}{2}\|\upsilon\|^2 + \frac{C}{2}\sum_{i=1}^{N}(g(x_i) - y_i)^2 \\ &\text{s.t. } |S| \le L \\ &\quad S \subset X \end{aligned} \tag{6}$$

where J_g is the value of J using $g(x)$ as the decision hyperplane. Because $f(x)$ is the optimal solution of (2), the smaller the difference between J_f and J_g, the higher is the similarity in the performance of $f(x)$ and $g(x)$.

Let $\mu = (\beta, b)^{(L+1)\times 1}$, $g(x) = \upsilon^T\varphi(x) + b = \mu^T(K(\{x\}, S), 1)$. When S is fixed, (6) can be transformed into the following optimization problem:

$$\min_{\mu} J_g = \frac{1}{2}\|\upsilon\|^2 + \frac{C}{2}\sum_{i=1}^{N}(g(x_i) - y_i)^2. \tag{7}$$

Then, the error item in (7) can be transformed into the following formula:

$$\begin{aligned} (g(x_i) - y_i)^2 &= \left(\mu^T(K(\{x_i\}, S), 1) - y_i\right)^2 \\ &= \mu^T(K(S, \{x_i\}), 1)(K(\{x_i\}, S), 1)\mu \\ &\quad - 2y_i(K(\{x_i\}, S), 1)\mu + y_i^2 \\ &= \mu^T \begin{bmatrix} K(S,\{x_i\})K(\{x_i\},S) & K(S,\{x_i\}) \\ K(\{x_i\},S) & 1 \end{bmatrix} \mu \\ &\quad - \begin{bmatrix} 2y_iK(\{x_i\},S) & 2y_i \end{bmatrix}\mu + y_i^2. \end{aligned} \tag{8}$$

Because y_i is constant, we transfer J_g' into the following formula with omitting the constant term:

$$\begin{aligned} J_g &= \frac{1}{2}\left(\beta^T K(S,S)\beta\right) \\ &\quad + \frac{C}{2}\sum_{i=1}^{N}\left(\mu^T \begin{bmatrix} K(S,\{x_i\})K(\{x_i\},S) & K(S,\{x_i\}) \\ K(\{x_i\},S) & 1 \end{bmatrix}\mu \right. \\ &\qquad \left. - \begin{bmatrix} 2y_iK(\{x_i\},S) & 2y_i \end{bmatrix}\mu\right) \\ &= \mu^T \begin{bmatrix} \frac{1}{2}K(S,S) & 0 \\ 0 & 0 \end{bmatrix}\mu \\ &\quad + \frac{C}{2}\mu^T\left(\sum_{i=1}^{N}\begin{bmatrix} K(S,\{x_i\})K(\{x_i\},S) & K(S,\{x_i\}) \\ K(\{x_i\},S) & 1 \end{bmatrix}\right)\mu \\ &\quad - \frac{C}{2}\sum_{i=1}^{N}\begin{bmatrix} 2y_iK(\{x_i\},S) & 2y_i \end{bmatrix}\mu \\ &= \mu^T \begin{bmatrix} \frac{1}{2}K(S,S) & 0 \\ 0 & 0 \end{bmatrix}\mu \\ &\quad + \mu^T \begin{bmatrix} \frac{C}{2}K(S,S)\sum_{i=1}^{N}k(x_i,x_i) & \frac{C}{2}\sum_{i=1}^{N}K(S,\{x_i\}) \\ \frac{C}{2}\sum_{i=1}^{N}K(\{x_i\},S) & \frac{CN}{2} \end{bmatrix}\mu \\ &\quad - \mu^T \begin{bmatrix} C\sum_{i=1}^{N}y_iK(S,\{x_i\}) \\ \sum_{i=1}^{N}Cy_i \end{bmatrix} = \frac{1}{2}\mu^{\mathrm{T}}A\mu - B\mu \end{aligned} \tag{9}$$

where

$$A = \begin{bmatrix} K(S,S)(1 + C\sum_{i=1}^{N}k(x_i,x_j)) & C\sum_{i=1}^{N}K(S,\{x_i\}) \\ C\sum_{i=1}^{N}K(\{x_i\},S) & CN \end{bmatrix} \tag{10}$$

$$B = \begin{bmatrix} C\sum_{i=1}^{N}y_iK(S,\{x_i\}) \\ \sum_{i=1}^{N}Cy_i \end{bmatrix}. \tag{11}$$

Once S is determined, (7) can reach the minimum value at the following condition:

$$\mu = A^{-1}B. \tag{12}$$

It is clear that we can obtain μ to reach the minimum value of (7) when (10) is nonsingular. Thus, the essential problem of sparse LS-SVM is to select a suitable S that can make (10) reach the full rank. To efficiently select support vectors for sparse LS-SVM, we select the support vectors directly from X without solving (4). The main approach involves the selection

Authorized licensed use limited to: Qufu Normal University. Downloaded on February 06,2021 at 08:09:25 UTC from IEEE Xplore. Restrictions apply.

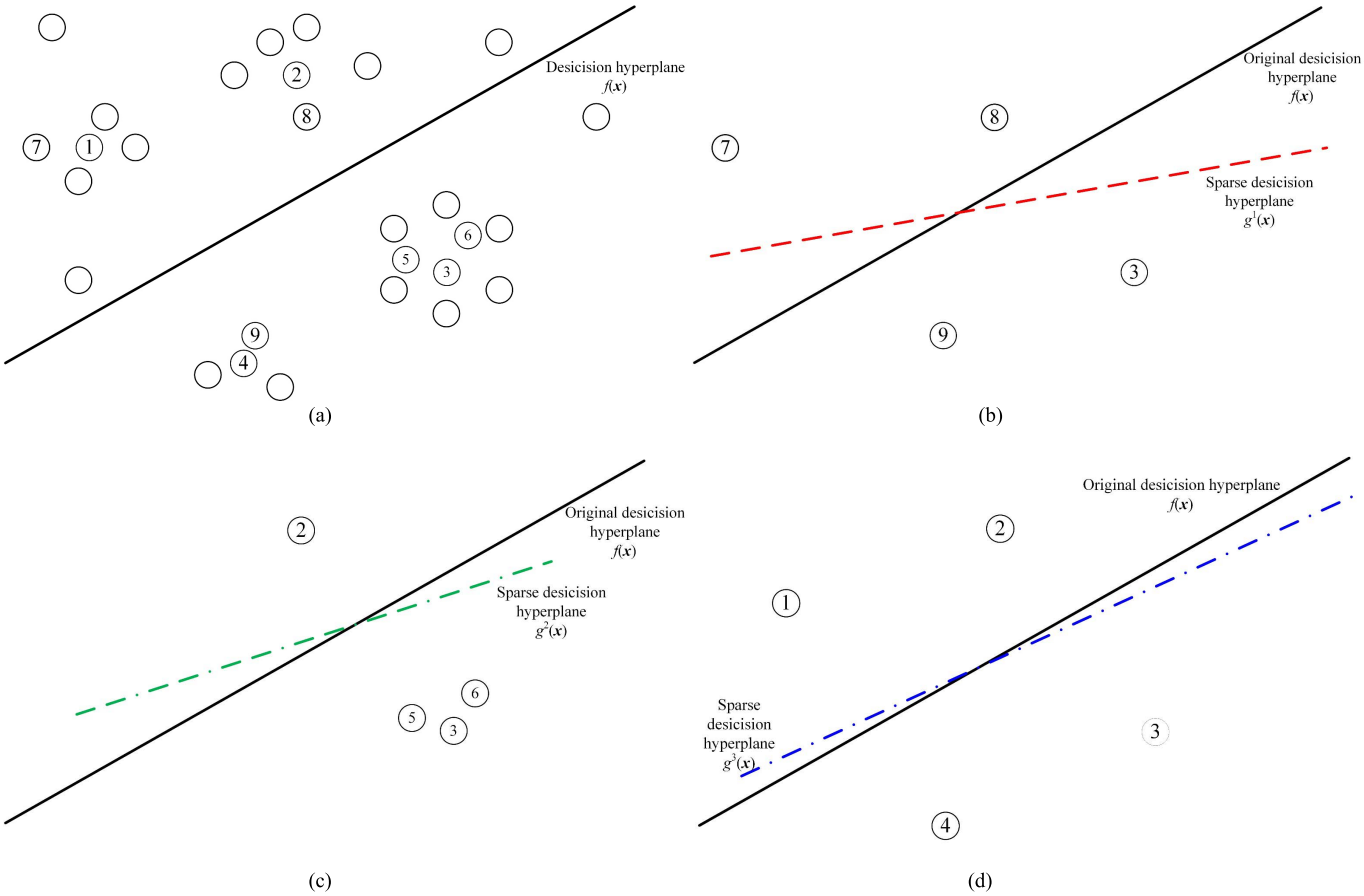

Fig. 1. Description of sparse LS-SVM constructed with different points in different areas in feature space with the number of reserved support vectors equal to four. (a) General LS-SVM in feature space with all data. (b) Sparse LS-SVM in feature space using points in a low-density area. (c) Sparse LS-SVM using points with consideration of density in feature space. (d) Sparse LS-SVM using points with consideration of density and dispersion in feature space.

of support vectors based on the characteristics of points in feature space. In Section III-B, we discuss the process of selecting the support vectors noniteratively.

B. Globally Representative Point Selection

Because the problem of selecting support vectors to construct S from X is an NP-hard problem, it is difficult to obtain the globally optimal solution. Therefore, we focus on developing a noniterative selection method based on the characteristics of points in feature space. We present the basic concept for this method in Fig. 1.

We display a general state of LS-SVM in feature space in Fig. 1(a). Each point is located on one side of the decision hyperplane (the black solid line) in feature space, and every point is a support vector. Because $|S| \ll |X|$, the process of approaching sparse LS-SVM can be viewed as a process of selecting points suitable to represent other points. The question of how to evaluate the representation of points in LS-SVM is discussed in the following.

Although all of the points can be included in S, the effect of a point in a different area is different if it is included in S. In Fig. 1(b), three points in low-density areas (points 7, 8, and 9) are selected into S. The decision hyperplane of sparse LS-SVM [red broken line, $g^1(x)$] displays a large deviation from the original decision hyperplane. We consider only the density of the points in feature space. If we select the points in a high-density area into S, the decision hyperplane of sparse LS-SVM is more similar to the original decision hyperplane than the decision hyperplane based on points in a low-density area. This situation is demonstrated in Fig. 1(c), where the decision hyperplane of sparse LS-SVM [green dot-dashed line, $g^2(x)$] is constructed with four points in the highest density area (points 2, 3, 5, and 6). Obviously, the points in the low-density area are less effective than the points in the high-density area. In other words, a point in a high-density area is more representative than a point in a low-density area in sparse LS-SVM. Therefore, it is a sensible choice to select the point in a high-density area into S rather than the point in the low-density area. However, in practice, if we only consider the density in feature space, this leads to an awkward state in which S is full of points from few high-density areas, as displayed in Fig. 1(c). Thus, we consider using both density and dispersion to select points into S. In Fig. 1(d), we use density and dispersion to evaluate the representation of a point and select the four top representative points (points 1, 2, 3, and 4) to construct S, which leads to the best decision hyperplane of sparse LS-SVM [blue dot-dashed

Authorized licensed use limited to: Qufu Normal University. Downloaded on February 06,2021 at 08:09:25 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1 GRPS

Input: dataset X, kernel function $k(\cdot,\cdot)$, threshold values θ.
Output: an ordered sequence τ
01) Initializing $\rho = 0^{N\times 1}, \zeta = 0^{N\times 1}$, $mdis = 0^{N\times 1}$ respectively
02) For $i = 1$ to N do:
03) Compute $D_{iX} = \{d_{ij} | x_j \in X\}$
04) Compute ρ_i based on (14) and (15)
05) Compute $mdis_i = \max(D_{iX})$
06) EndFor
07) For $i = 1$ to N do:
08) Construct $X_{\text{alternative}} = \{x_j | \rho_i < \rho_j\}$
09) If $X_{\text{alternative}}$ is empty:
10) Then
11) $\zeta_i = mdis_i$
12) Else
13) Compute $D_{iX\ \text{alternative}} = \{d_{ij} |\ x_j \in X_{\text{alternative}}\}$
14) $\zeta_i = \min(D_{iX\ \text{alternative}})$
15) EndIf
16) EndFor
17) Normalizing ρ and ζ based on (17) and (18) respectively
18) Compute τ based on (19)
19) Soring τ
20) Return τ

line, $g^3(x)$] to replace the original decision hyperplane in all of three decision hyperplanes.

For the convenience of discussion, we give two definitions for density and dispersion of a point in feature space based on the formulas in [26].

In feature space, the distance between x_i and x_j can be calculated using the following formula:

$$d_{ij} = \sqrt{k(x_i, x_i) + k(x_j, x_j) - 2k(x_i, x_j)}. \tag{13}$$

Based on the distance between two points, we give the density of point x_i in feature space as follows.

Definition 1 *(Point Density):* Let x_i be a point in data space and $k(\cdot,\cdot)$ be a kernel function. The number of points that are located in the θ neighborhood of x_i is referred to as the point density of x_i in feature space. The point density of x_i in feature space is denoted as ρ_i, which can be calculated as follows:

$$\rho_i = \sum_{j=1}^{N} \delta(d_{ij}) \tag{14}$$

where

$$\delta(z) = \begin{cases} 1, & z \le \theta \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

Obviously, ρ can be used to evaluate the representation of the points in the local area in feature space.

Definition 2 *(Point Dispersion):* Let x_i be a point in data space. The point dispersion in feature space denoted as ζ_i is defined as follows:

$$\zeta_i = \min_{\rho_i < \rho_j, j=1,\ldots,N} d_{ij} \tag{16}$$

where ζ_i is the least distance among the distances between x_i to other points whose ρ are larger than x_i [36], and the corresponding point X_j is called as the nearest neighbor (NN) of point x_i. For the point x with maximal ρ, we define its ζ as the maximal distance in all distances between x and the other points. Thus, ζ_i can be viewed as an indicator to evaluate the relative location of x_i in the entire data set in feature space.

For ρ and ζ as 2-D items, we can construct a space known as $\rho - \zeta$ space. In the $\rho - \zeta$ space, we discover that all of the points can be separated into three classes. The first class contains points with high point density and high point dispersion. The second contains the points with high point density and low point dispersion. The last class contains points with low point density and high point dispersion. In the $\rho - \zeta$ space, the three classes are located in three distinct areas. An illustration is shown in Fig. 2. The locations of data in data space and in ρ–ζ space are displayed in Fig. 2(a) and (b), respectively. Because $\rho > 0$ and $\zeta > 0$, all data are located in quadrant 1. The three classes are located in three separate areas in quadrant 1. Area I is located far from the ρ-axis and ζ-axis, which includes points of the first class such as points 1 and 6. Area II includes points of the second class close to the ρ-axis and far from the ζ-axis such as points 2 and 3. Area III is located far from the ρ-axis and close to the ζ-axis, which includes points of the last class such as point 7. Obviously, the points in different areas have different characteristics. If a point is located in area I, both the point density and the point dispersion of the point are larger than those of points in the two other sections. This observation means that the points lying in area I have highly global representation in all data. For example, point 1 can be viewed as a representative point for points 2 and 3. Points 4, 5, and 6 have the same ability to represent points located close to them. Therefore, the main problem of constructing S is to reasonably select points in area I.

Because the units of ρ and ζ are different, we cannot directly compare the values of ρ and ζ of different points. Therefore, we normalize ρ and ζ as follows:

$$\rho_i' = \frac{\rho_i - \rho_{\min}}{\rho_{\max} - \rho_{\min}} \tag{17}$$

$$\zeta_i' = \frac{\zeta_i - \zeta_{\min}}{\zeta_{\max} - \zeta_{\min}} \tag{18}$$

where $\rho_{\min}$ and $\rho_{\max}$ are the minimum and the maximum values of all ρ, and $\zeta_{\min}$ and $\zeta_{\max}$ are the minimum and the maximum values of all ζ, respectively. For convenience, we continue to use ρ_i and ζ_i to denote the values after normalization.

Then we introduce an indicator known as the global representation of a point, which is denoted as τ and calculated as follows:

$$\tau_i = \min(\rho_i, \zeta_i) \tag{19}$$

where ρ and ζ are normalized. It is clear that the larger the value of τ, the more representative a point will be. So we can select the top points to represent all points in the training set.

Obviously, to obtain all of τ_i for each point, we need to compute all d_{ij}. If we compute all d_{ij} in one step, the memory

Authorized licensed use limited to: Qufu Normal University. Downloaded on February 06,2021 at 08:09:25 UTC from IEEE Xplore. Restrictions apply.

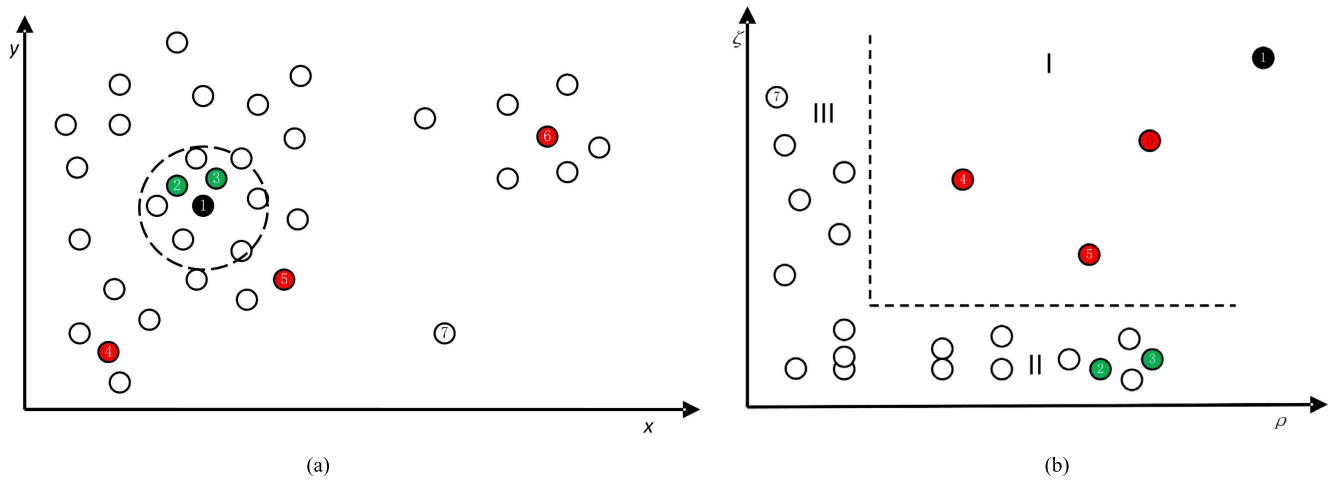

Fig. 2. Description of globally representative data. (a) Location of data in x–y space. (b) Location of data in ρ–ζ space.

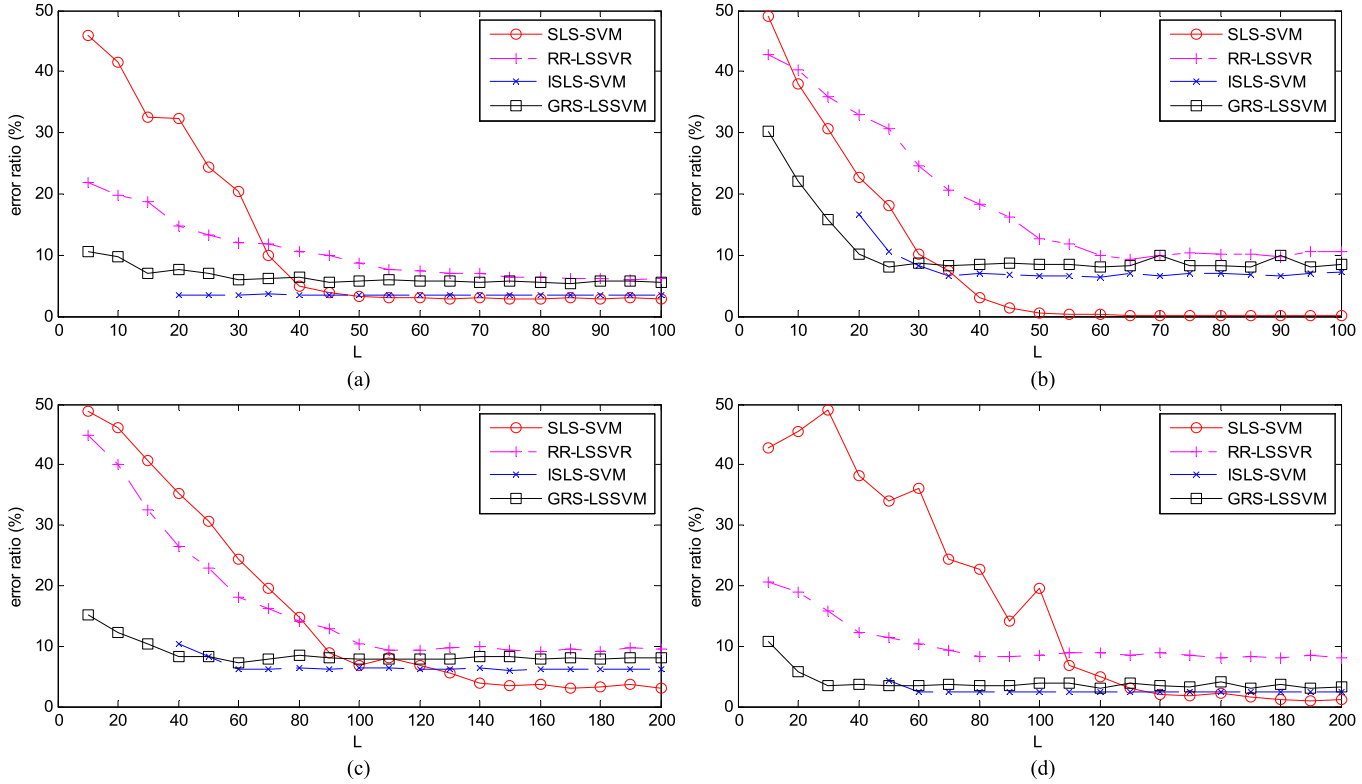

Fig. 3. Comparison of error ratio in different data sets. (a) BCW, (b) BA, (c) MK, and (d) LR, where L is the given number of reserved support vectors in sparse LS-SVM. Results from data sets.

cost should be $O(N^2)$. It is too large for large-scale data sets. Therefore, we used an alternative way to compute all of τ_i as algorithm Globally Representative Point Selection (GRPS). The flowchart of GRPS is displayed in Algorithm 1.

In GRPS, we use three variables to store information for each point. The three variables are ρ, ζ, and mdis. Each length is N. For x_i, ρ_i is used to store the value of point density, ζ_i is used to store the value of point dispersion, and mdis_i is used to store the maximal distance from x_i to other points.

The algorithm mainly includes three steps. The first step is to compute all ρ_i. For each x_i, we calculate $D_{iX} = \{d_{ij} | x_j \in X\}$ based on (13) and obtain ρ_i based on (14). The second step is to compute ζ_i for each point. For point x_i, we find all points whose ρ is bigger than ρ_i to construct set $X_{\text{alternative}} = \{x_j | \rho_i < \rho_j\}$. Then, we can calculate $D_{iX\,\text{alternative}} = \{d_{ij} | x_j \in X_{\text{alternative}}\}$ and obtain $\zeta_i = \min(d_{i*})$. The last step includes normalizing ρ and ζ, respectively, calculating τ, and sorting τ.

Then, we discuss the computational complexity and memory complexity of GRPS. For the first step, the computational complexity is $O(N^2)$ because we need to calculate all d_{ij}. The memory complexity is $O(N)$ because we need to store D_{iX}, ρ, and mdis with length N. For the second step, the computational complexity is $O(N^2)$ because we need to

Authorized licensed use limited to: Qufu Normal University. Downloaded on February 06,2021 at 08:09:25 UTC from IEEE Xplore. Restrictions apply.

compute all d_{ij} in the worst situation. The memory complexity is $O(N)$ because we need only to memorize D_{iX} and ζ with length N. For the last step, the computational complexity is $O(N \log N)$ if we sort τ by using a quick sorting method. The memory complexity is $O(N)$ because we need to put τ into the memory. Therefore, the computational complexity is $O(N^2)$ and memory complexity is $O(N)$, respectively.

In GRPS, θ plays an important role. Obviously, the value of θ determines how many neighbors are calculated for a point. In other words, the value determines the value of ρ for each point. However, the influence is global. It means that a small value of θ will lead to decrease in all the values of ρ of points. On the contrary, a large value of θ will lead to increase all values of ρ of points. In an acceptable interval, the value of θ has little influence on the ordered sequence of ρ. Then, the ordered sequence of ζ is stable, which leads to the ordered sequence of τ is stable. Therefore, GRPS is robustness for θ. The property has been discussed in [36].

C. GRS-LSSVM

To avoid a situation that S only includes point from one class, we employ a strategy that GRPS should be used on each class, respectively. We allocate the number of points in S to different classes based on the ratio of the number of points in different classes before selecting the globally representative points from the entire training set. The numbers of globally representative points selected from the positive class and negative class can be obtained via the following formulas:

$$L^+ = \max\left(1, \text{round}\left(L \times \frac{N^+}{N}\right)\right) \tag{20}$$

$$L^- = L - L^+ \tag{21}$$

where N^+ is the number of points in the positive class in the training set, N is the number of points in the training set, and round$(\cdot)$ is a rounding function, L^+ is the number of globally representative points selected from the positive class, L^- is the number of globally representative points selected from the negative class.

With L^+ and L^-, we can select the globally representative points for each class in the training data set to construct S. Based on S and (10), matrix A can be obtained. It is a very low possibility that A is ill-conditioned or even singular. To solve the problem, we can replace points that lead A to be singular in S with new globally representative points. However, it is expensive for computational complexity. To save computational time, we add a constant on diagonal with small entries. The pseudo-code for the GRS-LSSVM is given in Algorithm 2.

Obviously, the cost of GRS-LSSVM consists of two components in binary classification. One component contains two repetitions of calling the GRPS, and the other component calculates the coefficients of sparse LS-SVM. The cost of the computation of GRPS is $O(N^2)$. The computational complexity of the second component is $O(L^{\log 7}) \approx O(L^{1.95})$ because we need to compute A^{-1} [37], [38]. So, the computational complexity of the whole algorithm is $O(N^2 + L^{1.95})$. Because $L \ll N$, the total computational cost of GRS-LSSVM is $O(N^2)$. In memory cost, GRS-LSSVM mainly includes two sequences for two classes in two times of calling GRPS and matrix A in the second component. The number of points of the two sequences is N and the size of A is L^2. It is excited that the memory used by the two sequences can be released in the second component. Therefore, the memory complexity of GRS-LSSVM is $O(\max(N, L^2))$ in fact. Of course, in most conditions, the memory complexity of GRS-LSSVM is $O(N)$ for $L \ll N$. Therefore, GRS-LSSVM is a fast algorithm suitable for handling large-scale data. The code of GRS-LSSVM has been uploaded on Github with https://github.com/rzmyf1976/Spare-Least-Square-Support-Vcetor-Machine/.

Algorithm 2 GRS-LSSVM

Input: dataset X, label set Y, kernel function $k(\cdot,\cdot)$, threshold values θ^+ and θ^-, the given number of reserved support vectors L

Output: the set of reserved support vectors S, coefficients β and b

1) Compute L^+ and L^- using (20) and (21) respectively
2) τ^+ =GRPS(X^+, k, θ^+)
3) τ^- =GRPS(X^-, k, θ^-)
4) Obtain S^+ with selecting the top L^+ points from τ^+
5) Obtain S^- with selecting the top L^- points from τ^-
6) $S = S^+ \Upsilon S^-$
7) Compute β and b based on (12)
8) Return S, β, and b

TABLE I
DESCRIPTION OF DATA SETS

Datasets	The number of samples	Dimension	Ratio of the two classes
Breast Cancer Wisconsin (BCW)	684	9	445:239
Banknote Authentication (BA)	1372	4	610:762
Musk (MK)	7074	166	1224:5850
Letter Recognition (LR)	20000	16	789:19211
Skin Segmentation (SK)	245057	3	60859: 184198
HEPMASS (HM)	1000000	28	499665: 500335

IV. EXPERIMENTAL RESULTS AND DISCUSSION

All of the experiments are performed on a PC with Intel Xeon E5-1620, 16-G memory, running Windows 10. The algorithms are implemented in MATLAB.

A. Data Set Description

In this article, we use six real data sets from the UCI (UC Irvine Machine Learning Repository). A simple description of the six sets is presented in Table I. In data set LR, we let letter B represent a class and other letters represent another class.

Authorized licensed use limited to: Qufu Normal University. Downloaded on February 06,2021 at 08:09:25 UTC from IEEE Xplore. Restrictions apply.

In data set HM, we use the "all" data set and randomly select 1 000 000 samples to construct data set in experiments.

To evaluate the performance of the proposed algorithm, we use two indicators, namely, the computational time and error ratio. In our experiments, we used tenfold cross validation to evaluate the performances of the different methods. Ten experiments were conducted for each data set. The mean values of different indicators are displayed in the results.

In our experiments, we use Sparse LS-SVM (SLS-SVM) [6], RR-LSSVR [19], and Iterative Sparse LS-SVM (ISLS-SVM) [14] as comparison methods. Because the stopping criteria are different in the different algorithms, we use a different stopping criterion in each algorithm. The alternative stopping criterion is that the number of reserved support vector is no more than the given number L. SVM and LS-SVM are used as the base algorithm of sparse SVM and nonsparse SVM, respectively. The code of SVM is Libsvm which is downloaded from https://www.csie.ntu.edu.tw/∼cjlin/libsvm/ supported by Chih-Chung Chang and Chih-Jen Lin. The code of LSSVM is obtained from https://www.esat.kuleuven.be/ sista/lssvmlab/.

For kernel function selection, because RBF kernel function, $k(x, y) = \exp(-\Pi x - y\Pi^2/(2\sigma^2))$, has been used in many applications, we still use this kernel function in our experiments. The parameter setting of SVM has a significant influence on its application performance. If parameter values are set in appropriate, the performance of SVM may be very poor. For SVM, the error penalty parameter C and the kernel function parameter such as the kernel parameter σ^2 for the Gaussian kernel function are vital. In practice, we can use cross-validation method or other methods to select parameters [39]–[42]. In our experiments, we do not use these methods to select the optimal parameters for convenience. For the parameter σ^2, we choose 10 from 1, 10, 100, and 200 with simple comparison because the primary object of this article is to verify the performances in the generalization ability and complexity of these algorithms under the same condition. For parameter θ, we use 0.4 selected from 0.3, 0.35, and 0.4 in all experiments. The constant added on diagonal A is 10^{-6} in experiments.

B. Experimental Results

The error ratios and the computational times of different algorithms in different data sets are shown in Figs. 3 and 4, respectively. In Fig. 4, we use lg(time) as the vertical coordinate to replace the traditional vertical coordinate. The results of SVM and LS-SVM are listed in Table II, where N_s is the number of support vectors included in the decision hyperplane obtained by the algorithms.

From Table II, we observe that the generalization ability between SVM and LS-SVM is similar because the error ratios are approximate on different data sets. However, the sparseness is notably different between SVM and LS-SVM. In LS-SVM, all of the points in the training set are included in the decision hyperplane. On the contrary, with LS-SVM, SVM only includes approximately 20%–40% points in its decision hyperplane.

First, we find that SLS-SVM, RR-LSSVR, and GRS-LSSVM can all accomplish the task of sparseness when the number of reserved support vectors is given in most conditions. However, ISLS-SVM cannot satisfy the arbitrary number of reserved support vectors. In other words, when the given number of reserved support vectors is too small, ISLS-SVM cannot perform as expected. When the sparseness ratios reach steady levels, we observe that the order of the sparseness ratios obtained from different algorithms is GRS-LSV < ISLS-SVM < SLS-SVM < RR-LSSVR, which means that GRS-LSSVM can reach a high sparseness when the given number is small.

Second, for the indicator of error ratio, these algorithms demonstrate several common features. In the first algorithm, the error ratios of all algorithms can reach low levels approximate to those of LS-SVM when the reserved support vectors reach a threshold value. In the second, when the threshold value is reached, the increasing number of reserved support vectors does not lead to a prominent improvement in the error ratio. Finally, the standard deviation of the error ratio maintains a stable low level after the level of error ratio reaches a steady level. The differences in error ratios among the four algorithms are apparent. When the error ratio achieves a steady level, the order of the error ratios of the four algorithms is SLS-SVM < ISLS-SVM < GRS-LSSVM < RR-LSSVR. Although the error ratio of GRS-LSSVM is not the smallest in all the algorithms, GRS-LSSVM is exciting based on the performance of sparseness when the requirement is not as severe.

With respect to computational complexity, the performance of the four algorithms is quite different. SLS-SVM displays a trend of a slow decline in the computational time as the number of support vectors increases. The reason for this observation is that the iterative times of SLS-SVM should decrease as the number of reserved support vectors increases. For RR-LSSVR, the computational time rapidly increases with an increasing number of reserved support vectors because it is an incremental algorithm used to obtain the sparse LS-SVM with only one support vector included in the final support vector set. The proposed GRS-LSSVM displays three properties of computational time. First, for the same training data set, the computational time remains stable while the number of reserved support vectors changes. Second, for different training data sets, the computational time does not vary greatly, while the numbers of the training data sets change significantly.

From the above results, GRS-LSSVM shows stable performance in different experiments for the same training data set. Therefore, GRS-LSSVM displays comparable performance with respect to the other algorithms.

For a large-scale data set, we focus on the computational complexity and memory complexity of different algorithms. The results have been listed in Table III. The four algorithms, LS-SVM, SLS-SVM, ISLS-SVM, and RR-LSSVR, cannot be executed in the two data sets. The reason lies in which they are all $O(N^2)$ on memory complexity which means that the memory requirement will reach 40 GB when the training set is as large as 200 000 points. That is a scary cost. As far as the two large scale data sets, SK and HM,

Authorized licensed use limited to: Qufu Normal University. Downloaded on February 06,2021 at 08:09:25 UTC from IEEE Xplore. Restrictions apply.

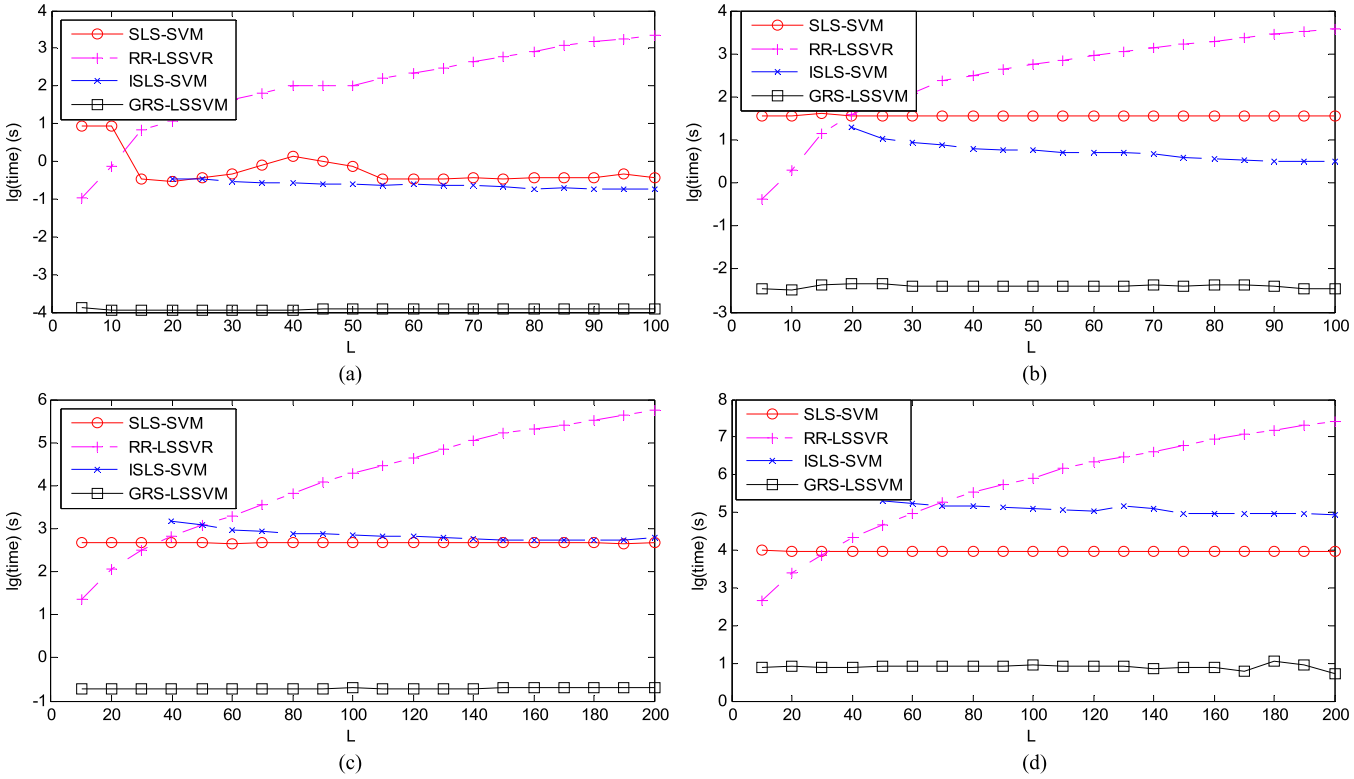

Fig. 4. Comparison of computation time in different data sets. (a) BCW. (b) BA. (c) MK. (d) LR.

TABLE II
RESULTS OF SVM AND LS-SVM IN DATA SETS BCW, BA, MK, AND LR

Dataset	SVM			LS-SVM		
	Error ratio (%)	Time (s)	N_S	Error ratio (%)	Time (s)	N_S
BCW	3(±0.01)	0.02(±0.005)	93.2(±0.85)	3(±0.01)	0.02(±0.001)	500(±0)
BA	2.4(±0.0 1)	0.09(±0.005)	418.8(±1.96)	1(±0.01)	0.072(±0.007)	1000(±0)
MK	5.7(±0.04)	0.3(±0.01)	642.2(±6.4)	5.1(±0.1)	0.38(±0.02)	2000(±0)
LR	1(±0.04)	1.32(±0.05)	1706(±79)	1(±0.035)	1.78(±0.05)	4000(±0)

TABLE III
RESULTS ON DATA SETS SK AND HM

L	SK		HM	
	Time(s)	Error(%)	Time(s)	Error(%)
10	7408.32+0.13	18.09	155597.18+0.26	29.27
20	7413.42+0.19	16.21	153356.42+0.86	25.46
100	7426.51+0.53	13.79	156783.44+1.02	18.77
500	7436.93+2.07	11.85	157592.67+3.35	14.25
1000	7488.56+3.84	11.21	157908.17+5.78	13.39
SVM	5354.32+103.4	4.38	138136.85+2926.62	8.07

1. The mean values of the number of support vectors are 29068(±1482) and 333371(±1482) on SK and HM of SVM respectively.
2. The time is consisted of two parts. One part is training time (value before plus) and another is testing time (value after plus).

are concerned, the algorithms SVM and GRS-LSSVM can be executed normally. On computational time, SVM costs about 5000 s on SK and 130 000 s on HM in the training stage, respectively. Corresponding with SVM, GRS-LSSVM uses about 7000 s on SK and 150 000 s on HM in the training stage, respectively. With the increasing number of training sets, the growth of GRS-LSSVM is lower than SVM on computational time. On the testing time, SVM is greatly larger than GRS-LSSVM because the number of support vector in SVM is larger than the number in GRS-LSSVM. Based on the results on the two data sets, GRS-LSSVM is more suitable for large-scale data set than SVM.

V. CONCLUSION

In this article, we proposed a novel algorithm for the sparseness of LS-SVM based on a globally representative candidate set. In this work, we demonstrated an optimization model on the sparseness of LS-SVM considering the constraint on the number of support vectors. Based on the density and dispersion of points in feature space, we designed an indicator to evaluate the global representation of points. By selecting support vectors in a noniterative strategy using global representation, a fast sparse LS-SVM algorithm was proposed with good performance in real applications. Unlike traditional algorithms, GRS-LSSVM can select all of the reserved support

Authorized licensed use limited to: Qufu Normal University. Downloaded on February 06,2021 at 08:09:25 UTC from IEEE Xplore. Restrictions apply.

vectors in one step, which means that GRS-LSSVM is suitable for working with a large-scale data set. Using a noniterative strategy, we reduced the computational complexity to $O(N^2)$ and memory complexity to $O(N)$. With this strategy, we can avoid repeated computation of the decision hyperplane, which wastes a large amount of time. In addition, we proposed a novel viewpoint in which the selection of the final support vectors is based on the overall distribution of the data set, which leads to a better solution than the traditional methods. Since GRS-LSSVM is based on the global representation in feature space, it is suitable for pruning other models based on support vectors with a simple modification.

REFERENCES

[1] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*. San Francisco, CA, USA: Morgan Kaufmann, 2011.

[2] V. Vapnik, *Statistical Learning Theory*. Hoboken, NJ, USA: Wiley, 1998.

[3] S. Theodoridis, K. Koutroumbas, *Pattern Recognition*. New York, NY, USA: Academic, 2008.

[4] J. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Process. Lett.*, vol. 9, no. 3, pp. 293–300, 1999.

[5] T. van Gestel *et al.*, "Benchmarking least squares support vector machine classifiers," *Mach. Learn.*, vol. 54, no. 1, pp. 5–32, Jan. 2004.

[6] J. A. K. Suykens, L. Lukas, and J. Vandewalle, "Sparse approximation using least squares support vector machines," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2000, pp. 757–760.

[7] J. A. K. Suykens, J. De Brabanter, L. Lukas, and J. Vandewalle, "Weighted least squares support vector machines: Robustness and sparse approximation," *Neurocomputing*, vol. 48, nos. 1–4, pp. 85–105, Oct. 2002.

[8] B. J. de Kruif and T. J. A. de Vries, "Pruning error minimization in least squares support vector machines," *IEEE Trans. Neural Netw.*, vol. 14, no. 3, pp. 696–702, May 2003.

[9] L. Hoegaerts, J. A. K. Suykens, J. Vandewalle, and B. De Moor, "A comparison of pruning algorithms for sparse least squares support vector machines," in *Proc. 11th Int. Conf. Neural Inf. Process.*, 2004, pp. 1247–1253.

[10] X. Zeng and X. Chen, "SMO-based pruning methods for sparse least squares support vector machines," *IEEE Trans. Neural Netw.*, vol. 16, no. 6, pp. 1541–1546, Nov. 2005.

[11] B. P. R. Carvalho and A. P. Braga, "IP-LSSVM: A two-step sparse classifier," *Pattern Recognit. Lett.*, vol. 30, no. 16, pp. 1507–1515, Dec. 2009.

[12] K. De Brabanter, J. De Brabanter, J. A. K. Suykens, and B. De Moor, "Optimized fixed-size kernel models for large data sets," *Comput. Statist. Data Anal.*, vol. 54, no. 6, pp. 1484–1504, Jun. 2010.

[13] P. Karsmakers, K. Pelckmans, K. De Brabanter, H. Van Hamme, and J. A. K. Suykens, "Sparse conjugate directions pursuit with application to fixed-size kernel models," *Mach. Learn.*, vol. 85, nos. 1–2, pp. 109–148, Oct. 2011.

[14] J. Lopez, K. D. Brabanter, J. R. Dorronsoro, and J. A. K. Suykens, "Sparse LS-SVMs with L_0-norm minimization," in *Proc. Eur. Symp. Artif. Neural Netw.*, 2011, pp. 189–194.

[15] K. Huang, D. Zheng, J. Sun, Y. Hotta, K. Fujimoto, and S. Naoi, "Sparse learning for support vector classification," *Pattern Recognit. Lett.*, vol. 31, no. 13, pp. 1944–1951, Oct. 2010.

[16] J. Liu, J. Li, W. Xu, and Y. Shi, "A weighted L_q adaptive least squares support vector machine classifiers—Robust and sparse approximation," *Expert Syst. Appl.*, vol. 38, no. 3, pp. 2253–2259, Mar. 2011.

[17] L. Wei, Z. Chen, and J. Li, "Evolution strategies based adaptive L_p LS-SVM," *Inf. Sci.*, vol. 181, no. 14, pp. 3000–3016, Jul. 2011.

[18] L. Jiao, L. Bo, and L. Wang, "Fast sparse approximation for least squares support vector machine," *IEEE Trans. Neural Netw.*, vol. 18, no. 3, pp. 685–697, May 2007.

[19] Y. Zhao and J. Sun, "Recursive reduced least squares support vector regression," *Pattern Recognit.*, vol. 42, no. 5, pp. 837–842, May 2009.

[20] X. Yang, J. Lu, and G. Zhang, "Adaptive pruning algorithm for least squares support vector machine classifier," *Soft Comput.*, vol. 14, no. 7, pp. 667–680, 2010.

[21] Y.-P. Zhao, J.-G. Sun, Z.-H. Du, Z.-A. Zhang, Y.-C. Zhang, and H.-B. Zhang, "An improved recursive reduced least squares support vector regression," *Neurocomputing*, vol. 87, pp. 1–9, Jun. 2012.

[22] J. Yang, A. Bouzerdoum, and S. L. Phung, "A training algorithm for sparse LS-SVM using compressive sampling," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2010, pp. 2054–2057.

[23] L. Yang, S. Yang, R. Zhang, and H. Jin, "Sparse least square support vector machine via coupled compressive pruning," *Neurocomputing*, vol. 131, pp. 77–86, May 2014.

[24] M. M. Fard, Y. Grinberg, J. Pineau, and D. Precup, "Compressed least-squares regression on sparse spaces," in *Proc. 26th AAAI Artif. Intell.*, 2012, pp. 1054–1060.

[25] X. Liang *et al.*, "Fast pruning superfluous support vectors in SVMs," *Pattern Recognit. Lett.*, vol. 34, no. 10, pp. 1203–1209, Jul. 2013.

[26] R. Mall and J. A. K. Suykens, "Sparse reductions for fixed-size least squares support vector machines on large scale data," in *Proc. 17th Pacific-Asia Conf. Knowl. Data Mining*, 2013, pp. 161–173.

[27] J. Shi, G. Si, Z. Guo, Y. Zhang, and S. Ma, "A pruning strategy based on confidence interval for sparse LS-SVM," in *Proc. 12th World Congr. Intell. Control Autom. (WCICA)*, Jun. 2016, pp. 577–582.

[28] R. Mall and J. A. K. Suykens, "Very sparse LSSVM reductions for large-scale data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 5, pp. 1086–1097, May 2015.

[29] S. Zhou, "Sparse LSSVM in primal using Cholesky factorization for large-scale problems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 4, pp. 783–795, Apr. 2016.

[30] S. Zhou and M. Liu, "A new sparse LSSVM method based the revised LARS," in *Proc. Int. Conf. Mach. Vis. Inf. Technol. (CMVIT)*, Feb. 2017, pp. 46–51.

[31] G. Si, J. Shi, Z. Guo, and Y. Zhang, "Density clustering pruning method based on reconstructed support vectors for sparse LSSVM," in *Proc. Chin. Control Decis. Conf. (CCDC)*, May 2016, pp. 3582–3587.

[32] F. Ebuchi and T. Kitamura, "Fast sparse least squares support vector machines by block addition," in *Porc. 14th Int. Symp. Neural Netw.*, 2017, pp. 60–70.

[33] B. Sun, W. W. Y. Ng, and P. P. K. Chan, "Improved sparse LSSVMS based on the localized generalization error model," *Int. J. Mach. Learn. Cybern.*, vol. 8, no. 6, pp. 1853–1861, Dec. 2017.

[34] D. A. Silva and A. R. R. Neto, "Multi-objective genetic algorithms for sparse least square support vector machines," in *Proc. Int. Conf. Intell. Data Eng. Automated Learn.*, 2014, pp. 158–166.

[35] L. Chen and S. Zhou, "Sparse algorithm for robust LSSVM in primal space," *Neurocomputing*, vol. 275, pp. 2880–2891, Jan. 2018.

[36] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, Jun. 2014.

[37] V. Strassen, "Gaussian elimination is not optimal," *Numerische Math.*, vol. 13, no. 4, pp. 354–356, Aug. 1969.

[38] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA, USA: MIT Press, 2011, p. 768.

[39] X. Zhang, W. Chen, B. Wang, and X. Chen, "Intelligent fault diagnosis of rotating machinery using support vector machine with ant colony algorithm for synchronous feature selection and parameter optimization," *Neurocomputing*, vol. 167, pp. 260–279, Nov. 2015.

[40] Y. Zhang and P. Zhang, "Machine training and parameter settings with social emotional optimization algorithm for support vector machine," *Pattern Recognit. Lett.*, vol. 54, pp. 36–42, Mar. 2015.

[41] A. Bablani, D. R. Edla, D. Tripathi, S. Dodia, and S. Chintala, "A synergistic concealed information test with novel approach for EEG channel selection and SVM parameter optimization," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 11, pp. 3057–3068, Nov. 2019.

[42] Z. Tao, L. Huiling, W. Wenwen, and Y. Xia, "GA-SVM based feature selection and parameter optimization in hospitalization expense modeling," *Appl. Soft Comput.*, vol. 75, pp. 323–332, Feb. 2019.

Yuefeng Ma received the B.S. degree in computer science from Xi'an Jiaotong University, Xi'an, China, in 1997, the M.S. degree in management science from Shandong Normal University, Jinan, China, in 2006, and the Ph.D. degree in computer science from the Department of Computer Science, Renmin University of China, Beijing, China, in 2017.

His main research areas are machine learning, artificial intelligence, and social networks.

Authorized licensed use limited to: Qufu Normal University. Downloaded on February 06,2021 at 08:09:25 UTC from IEEE Xplore. Restrictions apply.

Xun Liang (Senior Member, IEEE) received the B.Sc. and Ph.D. degrees in computer engineering from Tsinghua University, Beijing, China, in 1989 and 1993, respectively, and the M.Sc. degree in operations research from Stanford University, Palo Alto, CA, USA, in 1999.

He worked as a Post-Doctoral Fellow with the Institute of Computer Science and Technology, Peking University, Beijing, from 1993 to 1995, and with the Department of Computer Engineering, University of New Brunswick, Fredericton, NB, Canada, from 1995 to 1997. He worked as a CTO, leading over ten intelligent information products in RixoInfo Ltd., CA, USA, from 2000 to 2007, and was the Director of the Data Mining Lab, Institute of Computer Science and Technology, Peking University, from 2005 to 2009. He is currently a Distinguished Professor with the School of Information Science, Qufu Normal University, Jining, China. His research interests include support vector machines, web mining, and social computing.

Gang Sheng received the B.Sc. degree in computer science from Qufu Normal University, Qufu, China, in 2000, and the M.Sc. and Ph.D. degrees in computer science from Northeastern University, Shenyang, China, in 2005 and 2015, respectively.

He worked as a Post-Doctoral Fellow with the College of Mathematics and Information Science, Guangzhou University, Guangzhou, China, from 2015 to 2017. His research interests include applied cryptography, cloud computing, and outsourced computation.

Dr. Sheng is a member of the Chinese Association for Cryptologic Research.

James T. Kwok (Fellow, IEEE) received the Ph.D. degree in computer science from the Hong Kong University of Science and Technology, Hong Kong, in1996.

He was an Assistant Professor with the Department of Computer Science, Hong Kong Baptist University, Hong Kong. He is currently a Professor with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. His current research interests include kernel methods, machine learning, pattern recognition, and artificial neural networks.

Dr. Kwok received the IEEE Outstanding Paper Award in 2004 and the Second Class Award in Natural Sciences from the Ministry of Education, China, in 2008. He was a Program Co-Chair for a number of international conferences and served as an Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS from 2006 to 2012. He is currently an Associate Editor of *Neurocomputing*.

Maoli Wang received the Ph.D. degree from Harbin Engineering University, Harbin, China, in 2008.

He is currently a Professor with the School of Computer, Qufu Normal University, Jining, China. His current research interests include adaptive control, fault-tolerant control, machinery intelligence, and digit control technique.

Guangshun Li received the Ph.D. degree from Harbin Engineering University, Harbin, China, in 2008.

He was a Visiting Scholar with The Hong Kong Polytechnic University, Hong Kong, in the second half year of 2019. He is currently an Associate Professor with the School of Information Science and Engineering, Qufu Normal University, Jining, China. His research interests include wireless networks, the Internet of Things (IoT), and big data.

Authorized licensed use limited to: Qufu Normal University. Downloaded on February 06,2021 at 08:09:25 UTC from IEEE Xplore. Restrictions apply.