Hindawi
Mathematical Problems in Engineering
Volume 2017, Article ID 4191789, 12 pages
https://doi.org/10.1155/2017/4191789

Research Article

Reconstruct the Support Vectors to Improve LSSVM Sparseness for Mill Load Prediction

Gangquan Si, Jianquan Shi, Zhang Guo, Lixin Jia, and Yanbin Zhang

State Key Laboratory of Electrical Insulation and Power Equipment, School of Electrical Engineering, Xi'an Jiaotong University, Xi'an, China

Correspondence should be addressed to Jianquan Shi; shijianquan46@126.com

Received 24 October 2016; Revised 14 May 2017; Accepted 25 May 2017; Published 5 July 2017

Academic Editor: Erik Cuevas

The sparse strategy plays a significant role in the application of the least square support vector machine (LSSVM), to alleviate the condition that the solution of LSSVM is lacking sparseness and robustness. In this paper, a sparse method using reconstructed support vectors is proposed, which has also been successfully applied to mill load prediction. Different from other sparse algorithms, it no longer selects the support vectors from training data set according to the ranked contributions for optimization of LSSVM. Instead, the reconstructed data is obtained first based on the initial model with all training data. Then, select support vectors from reconstructed data set according to the location information of density clustering in training data set, and the process of selecting is terminated after traversing the total training data set. Finally, the training model could be built based on the optimal reconstructed support vectors and the hyperparameter tuned subsequently. What is more, the paper puts forward a supplemental algorithm to subtract the redundancy support vectors of previous model. Lots of experiments on synthetic data sets, benchmark data sets, and mill load data sets are carried out, and the results illustrate the effectiveness of the proposed sparse method for LSSVM.

1. Introduction

The ball mill pulverizing system is widely applied to large-and-medium scale power plant in China, which almost uses tubular ball mills to pulverize the coal. A lot of design parameters and unmeasured operating parameters impact the pulverizing circuits. Mill load is an important unmeasurable parameter, which is closely related to the energy efficiency of the pulverizing system. Much research has been presented to measure the mill load in the past decades [1, 2]. The soft sensing technique is a new and low cost method, which selects measurable information to estimate the mill load by building models. The most common of measurable parameters are the noise and vibration data [3, 4]. But how to build a forecasting model which plays a significant role in soft sensing technique is a significant problem. So far, researchers have come up with many methods to build the soft sensing model, such as neural networks [5], the support vector machines (SVM) [6], partial least squares [7], and least squares support vector machine (LSSVM) [8]; these methods are aimed at specific problem. In this paper, we mainly research the mathematical problems of LSSVM for building the soft sensing model.

LSSVM, as proposed by Suykens, have been introduced for reducing the computational complexity of SVM. In LSSVM, the inequality constraints are replaced with equality constraints in solving a quadratic programming. Thus it has faster speed than SVM in the training process. However, there exist two main drawbacks in LSSVM, as its solution suffers from lack of sparseness and robustness [9]. These problems will increase the training time and reduce the model prediction accuracy for the real industrial data sets, which have troublesome characters such as imbalanced distribution, heteroscedasticity, and the explosion of data. Therefore, this paper focuses on the mathematic problem of how to improve LSSVM sparseness and robustness for real industrial data set and applies it to mill load prediction.

Many efforts have been made to mitigate these shortcomings. For example, Suykens et al. introduced a pruning algorithm based on sorted support value spectrum and proposed a sparse Least Squares Support Vector Classifier, which gradually removes the training samples with the smallest

absolute support values and retrains the reduced network. Later, this method was extended to the problem of Least Square Support Vector Regression [10]. Meanwhile, weighted LSSVM have been presented to improve the robustness of LSSVM solution to better [9]. From that point, the sparse algorithms based on strategy of pruning were popping up. Kruif and Vries presented a more sophisticated mechanism of selecting support vectors [11], in which the training sample introducing the smallest approximation error when it is omitted will be pruned. For more on LSSVM pruning algorithms, Hoegaerts et al. [12] provided a comparison among these algorithms and concluded that pruning schemes can be divided into QR decomposition and searching feature vector. Instead of determining the pruning points by errors, Zeng and Chen [13] introduced the sequential minimal optimization method to omit the datum that will lead to minimum changes to a dual objective function. Based on kernel partial least squares identification, Song and Gui [14] presented a method to get base vectors via reducing the kernel matrix by Schmidt orthogonalization.

Generally speaking, the algorithms mentioned above all follow backward pruning strategy. Correspondingly, the methods of forward selecting support vector iteratively recently are used for the sparseness. Yu et al. [15] provided a sparse LSSVM based on active learning, which greedily selected the datum with the biggest absolute approximation errors. Jiao et al. [16] introduced a fast sparse approximation scheme for LSSVM, which picks up the sample with making most contribution to the objective function. Later, an improved method [17] based on partial reduction strategy was extended to LSSVR. Subsequently, a recursive reduced least square support vector machine (RRLSSVM) and an improved algorithm of RRLSSVM (IRRLSSVM) were propose [18, 19]. They all choose the support vector which leads to the largest reductions on the objective function. However, the difference between them is that IRRLSSVM update the weights of the selected support vectors during the selection process, and RRLSSVM is not so. Additionally, RRLSSVM has been applied for online sparse LSSVM [20].

Backward algorithms need higher computational complexity since the full-order matrix is gradually decomposed into submatrix which leads to minimal increment for objective function [21]. Instead, forward algorithms need small computational complexity and small amount of memory required, but convergence of these algorithms has not been proved. In addition, there are some methods to sparse LSSVM from other aspects, for example, based on genetic algorithms [22, 23] and compressive sampling theory [24].

For aforementioned sparse algorithms, the hyperparameters optimized under the original data set remain the same in the process of greedy learning, in which almost all employ the radial basis function (RBF). In other words, the process of greedy learning can be considered as the parameter selects the support vector, because kernel function RBF has a local characteristic, with the measurement of the Euclidean distance between data sets. Therefore, there is another perspective to think about the sparseness. No matter what kind of algorithm, the initial model with all training data will be obtained in advance. And what we finally wanted is to reconstruct the model with the least support vectors and hold nearly approximation accuracy with the initial model. Hence, we attempt to realize the sparseness of LSSVM by reconstructing the support vectors to revert the initial models, and the refracturing strategy corresponds to the parameters of RBF. The reconstructed least square support vector machine for regression problems was proposed, abbreviated RCLSSVR. Moreover, the most noticeable innovation is to analyze the features of industrial data sets and introduce RCLSSVR to improve sparseness and robustness simultaneously. There are some features in different industrial data sets, such as imbalanced data distribution and heteroscedasticity. That would lead to a big difference in the process of sparseness due to cut or added datum with different position, which caused the iterative algorithm to choose more support vectors in order to gain the robustness. So we reconstruct the support vectors according to the initial model and the location of the original data; the problem of robustness and sparseness will be solved simultaneously.

This paper is organized as follows. In Section 2, the preliminaries knowledge is briefly introduced, including the fundamental of LSSVM and the principle of reduced LSSVM. The characteristic of real industrial conditions and our proposed algorithm will be developed in Section 3. Some simulations are taken on some function approximation problems and benchmark data sets in Section 4. Finally, the paper is concluded by Section 5. To facilitate reading, we have made a list of abbreviations for some necessary abbreviations in the table of Abbreviations.

2. Normal LSSVM and the Reduced LSSVM

Given a training data set $\{x_i, y_i\}_{i=1}^N$, where $x_i \in R^m$ is the input with m-dimension and $y_i \in R$ is its corresponding target. The goal of function approximation is to find the underlying relation between the input and the target value. Once this relation is found, the outputs corresponding to the inputs that are not contained in the training set can be approximated.

In the LSSVM, the relation underlying the data set is represented as a function of the following form:

$$\widehat{y}(x) = \omega^T \varphi(\mathbf{x}_i) + b, \tag{1}$$

where φ is a mapping of the vector $\mathbf{x}$ to a high dimensional feature space, b is the bias, and ω is a weight vector of the same dimension as feature space. The mapping $\varphi(\mathbf{x})$ is commonly nonlinear and makes it possible to approximate nonlinear functions. Mappings that are often used result in an approximation by a radial basis function, by polynomial functions, or by linear functions [25].

The approximation error for sample is defined as follows:

$$e_i = y_i - \widehat{y}(x_i). \tag{2}$$

The optimization problem is to search for those weights that give the smallest summed quadratic error of the training

samples in LSSVM. The minimization of the error together with the regularization is given as

$$\min \quad J(\omega, e) = \frac{1}{2}\omega^T\omega + \gamma\frac{1}{2}\sum_{i=0}^{N} e_i^2 \tag{3}$$

with equality constraint

$$y_i = \omega^T\varphi(x_i) + b + e_i; \tag{4}$$

here γ is the regularization parameter; this constrained optimization problem is solved by introducing to an unconstrained Lagrangian function:

$$J(\omega, e) - \sum_{i=0}^{N}\alpha_i\left(\omega^T\varphi(x_i) + b + e_i - y_i\right), \tag{5}$$

where $\alpha_i \in R$ is the Lagrange multiplier of x_i. The optimum can be found by setting the derivatives equal to zero

$$\begin{aligned}
&\frac{\partial L}{\partial \omega} = 0 \longrightarrow \omega = \sum_{i=1}^{N}\alpha_i\varphi(x_i)\\
&\frac{\partial L}{\partial b} = 0 \longrightarrow \sum_{i=1}^{N}\alpha_i = 0\\
&\frac{\partial L}{\partial e_i} = 0 \longrightarrow \alpha_i = \gamma e_i\\
&\frac{\partial L}{\partial \alpha_i} = 0 \longrightarrow \omega^{\mathbf{T}}\varphi(x_i) + b + e_i - y_i = 0, \quad i = 1, \ldots, N.
\end{aligned} \tag{6}$$

Eliminating the variables ω and e_i, we can get the following linear equations:

$$\begin{bmatrix} 0 & \vec{\mathbf{1}}^{\mathbf{T}} \\ \vec{\mathbf{1}} & \boldsymbol{\Omega} + \gamma^{-1}\mathbf{I} \end{bmatrix}\begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{y} \end{bmatrix}, \tag{7}$$

where $\mathbf{y} = [y_1; \ldots; y_N]$, $\vec{\mathbf{1}} = [1; \ldots; 1]$, $\vec{\mathbf{1}} = [1; \ldots; 1]$, $\boldsymbol{\Omega}$ is a square matrix, $\Omega_{mn} = \varphi(\mathbf{x}_m)^T\varphi(\mathbf{x}_n) = \mathbf{K}(\mathbf{x}_m, \mathbf{x}_n)$, $m, n = 1, \ldots, N$, and $\mathbf{K}$ is the kernel function which satisfies Mercers condition. It used to calculate the mapping in input space, instead of the feature space. In LSSVM, we can use linear, polynomial, and RBF kernels and others that satisfy Mercers condition. In this paper we focus on the RBF kernel as follows:

$$\mathbf{K}(\mathbf{x}, \mathbf{x}_i) = \exp\left\{-\frac{\left|\mathbf{x} - \mathbf{x}_i\right|^2}{\delta^2}\right\}. \tag{8}$$

The solution of this set of equations results in a vector of Lagrangian multipliers α and a bias b. The output of the approximation can be calculated for new input values of x with α and b. The predicting value is derived as follows:

$$y(x) = \sum_{i=1}^{N}\alpha_i\mathbf{K}(x_i, x) + b. \tag{9}$$

For the sparse LSSVR, the pruning or reduced strategy is applied to the solution, which let $\omega = \sum_{i\in S}\alpha_i k(x_i, \cdot)$, where S is the index subset of $\{1, \ldots, N\}$. Take ω into (3) and get the following equation:

$$\begin{aligned}
\min \quad & J(\alpha_{\mathbf{S}}, b)\\
&= \frac{1}{2}\alpha_S^T\mathbf{K}_{SS}\alpha_S\\
&\quad + \frac{\gamma}{2}\sum_{i=1}^{N}\left(y_i - \sum_{j\in S}\alpha_j K(x_i, x_j) - b\right)^2,
\end{aligned} \tag{10}$$

where $\mathbf{K}_{SS}$ is $K_{ij} = K(x_i, x_j)$, $i, j \in S$, and $\alpha_{\mathbf{S}}$ is the subset of α with the index subset S. Reformulate (10); we can obtained

$$\begin{aligned}
\min \quad & J(\alpha_S, b)\\
&= \left[b\alpha_S^T\right]\left(\begin{bmatrix} 0 & \mathbf{0}^T \\ \mathbf{0} & \dfrac{\mathbf{K}_{SS}}{\gamma} \end{bmatrix} + \begin{bmatrix} \mathbf{1}^T \\ \mathbf{K} \end{bmatrix}\right)\times\begin{bmatrix} b \\ \alpha_S \end{bmatrix}\\
&\quad - 2\left(\begin{bmatrix} \mathbf{1}^T \\ \widehat{\mathbf{K}}_S \end{bmatrix}\mathbf{y}\right)^T\begin{bmatrix} b \\ \alpha_S \end{bmatrix};
\end{aligned} \tag{11}$$

here $\widehat{\mathbf{K}}_S$ is $K_{ij} = K(x_i, x_j)$, $i \in S$, $j \in N$. $\mathbf{0}$ and $\mathbf{1}$ is an appropriate vector. The solution is given by derivative of (11) with respect to b and α_S.

$$\left(\begin{bmatrix} 0 & \mathbf{0}^T \\ \mathbf{0} & \dfrac{\mathbf{K}_{SS}}{\gamma} \end{bmatrix} + \begin{bmatrix} \mathbf{1}^T \\ \widehat{\mathbf{K}}_S \end{bmatrix}\right)\begin{bmatrix} b \\ \alpha_{\mathbf{S}} \end{bmatrix} = \begin{bmatrix} \mathbf{1}^T \\ \widehat{\mathbf{K}}_S \end{bmatrix}\mathbf{y} \tag{12}$$

So the reduced LSSVR is found:

$$y(x) = \sum_{i\in S}\alpha_i\mathbf{K}(x_i, \mathbf{x}) + b. \tag{13}$$

Comparing (9) and (13), we can know that the subset of training sample makes contribution to the model rather than every training sample. Therefore, the computational complexity and the operation time will be decreased in the prediction.

3. Reconstruct Least Square Support Vector Regression

3.1. The Characteristic of Industrial Data Set. In the real industrial process, there are various reasons resulting in the cases of imbalanced data distribution and heteroscedasticity. Thus both the features need to be briefly introduced. The first feature is the imbalanced distribution, which commonly exists in classification domains, such as spotting unreliable telecommunication customers, text classification, and detection of fraudulent telephone calls [26, 27]. Certain solutions at the data and algorithmic levels are proposed for the class-imbalance problem [28–30]. It is also a problem in sparse approximation. When adding or deleting a support vector from different areas, the change of estimated performance is different, and the samples in the rare areas will be cut off easily

in the sparse process. The second feature is the heteroscedasticity. The problem of heteroscedasticity, non-constant error variance, will bring about severe consequences: the variance of the parameter is not the least, and the prediction precision decreases [31]. The solution is divided into two categories: the data transformation method and the model of heteroscedastic data [32, 33]. In estimation function, the prediction precision will be more sensitive to the samples in the regions with high variance. And the number of support vectors in high variance regions will usually be more. At last, the explosion of data, especially containing imbalance and heteroscedastic data set, has been plaguing the machine learning.

Both forward algorithms and backward algorithms have certain shortcomings on industrial data sets. Firstly, for the imbalanced data sets, since the objective function is to balance the upper bound of the maximum points to the hyperplane with the minimum mean square error, the rare samples make a little contribution to the objective function. Hence, the data in region with more samples is easier to be selected than the data in region with rare samples; secondly, for the heteroscedastic data sets, the data in the area of big variance will be selected or left relatively larger number than the data in the area of small variance; finally, for the excessive data sets, the main problem is excessiveness. The training time is unacceptable for backward greedy methods, and the convergence problem will exist in forward greedy algorithms.

3.2. RCLSSVR and DRCLSSVR. In order to solve the aforementioned problem, we proposed an efficient method via reconstructing support vectors to restore the original model. From (8), we can know common kernel function has a local characteristic, equivalent of normalizing Euclidean distance between data sets based on the parameter. Therefore, it is possible to restore the original model by evenly choosing the support vector near the hyperplane and adjusting the parameter δ. But it is not always stable convergence because of the unknown data distribution, which can be solved by directly selecting the support vector on the original model instead of from the training data set. Thus, according to the position information of training data set and the original model, we can rebuild the selected data set $D = \{x_j, \widehat{y}_j\}_{j=1}^{N}$, where $\widehat{y}$ is the estimated value calculated by (9). Then the reconstructed samples in the data set D can be selected as the support vectors.

The sparse strategy is different from the forward and backward greedy methods, and the process has two steps. The first step is to select uniform samples by passing through the entire data set D. Arbitrarily pick one point as the density center $C = \{x_d, y_d\}$ from the original data set in the beginning and calculate the Euclidean distance vector **Dist** by (14). Find out all of samples **M** within the density center neighborhood radius R and update the density center based on (15).

$$\mathbf{Dist}(d,i) = \sqrt{(x_d - x_i)^2 + (y_d - y_i)^2} \quad i \in \{1,\ldots,N\} \tag{14}$$

$$C = \underset{j \in I_M}{\operatorname{argmin}} \frac{\left|\widehat{y}_j - y_j\right|}{\mathbf{Dist}(d,j)} \tag{15}$$

$$\mathbf{M} = \left\{(x_i, y_i) \mid \mathbf{Dist}(d,i) \le R,\ i = 1,\ldots,N\right\} \tag{16}$$

$$R = r \cdot \mathrm{Dist\_max}, \tag{17}$$

where I_M is the index set of **M** and also the subset of $\{1,\ldots,N\}$, and $|\widehat{y}_j - y_j|$ represents the absolute error between predicted value and true ones. r is the ratio of maximum distance, deciding how many support samples to select. Dist_max is the maximum distance between two samples of original data set. This means that the datum far away from the density center and near the hyperplane will be selected as the new density center. This selecting process will be terminated when traversing the original data set. The second step is to select support vector $\{x_d, \widehat{y}_d\}$ from data set D with correspondence to the density center in each iteration. Based on the selected support vectors, the regularization parameter and kernel parameter will be optimized again by leave-one-out methods. The realizing flowchart of RCLSSVR is depicted in Algorithm 1.

Algorithm 1 (RCLSSVR). Input: a training data set $\{x_i, y_i\}_{i=1}^{N}$, the radius coefficient r, the unlabeled data set $U = \{x_j, y_j\}_{j=1}^{N}$, the support vectors $\mathbf{S} = \phi$, and the neighborhood data set $M = \phi$.

Output: the support vectors **S**.

(1) Train original model via solving (7) with the training data set where the hyperparameter (δ, γ) is found by 10-fold cross validation;

(2) obtain function estimation $\widehat{y}_{i=1}^{N}$ by (9), and get the reconstructed data set $D = \{x_i, \widehat{y}_i\}_{i=1}^{N}$;

(3) calculate the Euclidean distance matrix max Dist and the radius R;

(4) randomly select a density center $C = \{x_d, y_d\}$, and add $\{x_d, \widehat{y}_d\}$ from D to **S**;

(5) update the unlabeled data set $U = U - M - C$, where M and C are update in each iteration;

(6) update the set $M = \{(x_j, y_j) \mid \mathbf{Dist}(d,j) \le R, (x_j, y_j) \in U\}$, and if $\mathbf{M} \neq \phi$, choose the next density center according to (15) and add corresponding datum $\{x_d, \widehat{y}_d\}$ to **S**; else, select the sample $\{(x_j, y_j) \mid \min\mathbf{Dist}(d,j), (x_j, y_j) \in U\}$ as the next density center and add corresponding datum $\{x_d, \widehat{y}_d\}$ to **S**;

(7) if $U = \phi$, then go to next step; else go to step (5);

(8) optimize the parameter (δ, γ) again via leaving one out cross validation on data set **S**, and rebuild the model.

Generally speaking, the rate of convergence and the performance of RCLSSVR are bound up with the parameter r. The smaller r can avoid the underfitting problem, which also multiplies the number of support vectors. But the bigger r will reduce the prediction accuracy. There are two situations worth considering after the end of RCLSSVR; one is that the

prediction precision does not meet the demands; the other is the data set **S** is still allowed to prune. Therefore, for the RCLSSVR, we should take remedial measures in two ways. The first is to improve the prediction precision via gradually adding sample to the support vectors from data set D, and the data with the smallest approximation error when it is added will be selected. The second is to prune the redundant samples with meeting the precision requirement. We define density indicators **Den** to decide which sample is omitted.

$$\mathrm{Den}_i = \sum_{i,m \in s} \exp\left(-\frac{\left\| (x_i, \widehat{y}_i) - (x_m, \widehat{y}_m) \right\|^2}{(r/2)^2} \right), \tag{18}$$

where s is the index of the support vector **S**; the value of r is the same as the parameter r. In the process of pruning, the sample with the biggest values in the **Den** will be pruned. The remedial method is named DRCLSSVR, which is described in Algorithm 2.

Algorithm 2 (DRCLSSVR). (1) Comparing the performance with the set value, if less, go to (5); else go to (2);

(2) calculate the density of all support values **Den** and sort it. Remove sample with the biggest values in the **Den**;

(3) retrain the LSSVM based on the reduced support vectors;

(4) go to (2), unless the performance degrades the set value;

(5) determine the training sample from D and add the data with the performance most greatly increased after selecting;

(6) retrain the LSSVM based on the added support vectors;

(7) go to (5), unless the performance reached to the set.

4. Experimental Results

In order to verify the performance of the proposed RCLSSVR and DRCLSSVR, some kinds of experiments are performed. In Section 4.1, the influence of different parameter r on sparse performance was studied. In Section 4.2, we selected two backward algorithms and one forward algorithm to perform comparative experiments on synthetic data sets and benchmark data sets. In Section 4.3, RCLSSVR is applied to mill load data set. For comparison purpose, all experiments are finished on a platform of Intel Core i5-4460 CPU @3.20 GHz processor with 4.00 GB RAM of windows 7 operation in a Matlab2014a environment, and a toolbox of LS-SVMlab v1.8 from http://www.esat.kuleuven.be/sista/lssvmlab/. The comparison algorithms are normal LSSVM, Suykens pruning algorithm (SLSSVM) [14], backward classic algorithm (PLSSVM) [16], and IRRLSSVM [23]. Among them, PLSSVM is expected to perform best, but it is an extremely expensive algorithm. RBF kernel is used in all of the experiments, and the parameter (γ, δ) is optimized by leaving one out cross validation strategy [22]. In addition, two performance indexes, that is, rooted mean squared errors (RMSE) and RMSE%, are defined to evaluate these algorithms.

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{N} (y_i - \widehat{y}_i)^2}{N}} \tag{19}$$

$$\mathrm{RMSE\%} = \frac{\mathrm{normal\_RMSE}}{\mathrm{RMSE}} \times 100, \tag{20}$$

where normal_RMSE is the RMSE of normal LSSVM.

4.1. Experiment 1: The Performance with Different r. In this subsection, we will utilize sinc function to investigate the performance of our proposed algorithms with different r. Since the DRCLSSVR is the supplement of RCLSSVR, the experiment only explores the performance of RCLSSVR with different parameters. This proves the influence of parameter in situations where we make more than once time pass through the data set. In other words, until certain number of vectors is reached, we will let $U = \{x_j, y_j\}_{j=1}^{N}$ when $U = \phi$. The sinc functions relation between inputs X and outputs Y is described as follows:

$$Y = \frac{10 \sin(X)}{X} + \varepsilon, \tag{21}$$

where $X \in [-10, 10]$, sampling with the same intervals, total 300 data. And ε is Gauss noise whose average value is equal to 0 and the variance is equal to 0.5. For this data set, we randomly select 200 samples as the training data set and the others as the testing data set.

In order to improve the generalization ability, it is necessary to normalize the attributes of training data sets into the closed interval $[0, 1]$ when calculating the Euclidean distance vector **Dist**. The parameter r reflects the number of support vectors of proportion of training data set, which ranged from 0.1 to 0.3 at 0.05 intervals. The simulation on RCLSSVR is plotted in Figure 1.

As can be seen from Figure 1, the larger parameter r leads to the faster convergence speed. When r is bigger than 0.25, the performance will show fluctuation because the algorithm updates the density center by repeatedly passing through the data set. When the parameter is smaller than 0.2, the velocity of convergence will be too slow in that there is a need for more #SV to make a pass through the training data set. The most attractive characteristic is that the performance will reach the same as the normal LSSVR, when the support vectors are up to a certain number. Considering rapid convergence and good stability, the parameter r, in this paper, will be set to 0.2 for all training data sets.

In order to confirm the effectiveness of DRCLSSVR and observe the selected #SV between RCLSSVR and DRCLSSVR, the comparison of sparse result is displayed in Figure 2. It is clear that DRCLSSVR needs less #SV when reaching almost the same generalization performance. It also proves the support vectors of RCLSSVR are redundant.

4.2. Experiment 2: The Simulation on Synthetic Data Sets and Benchmark Data Sets. In this subsection, we compare the

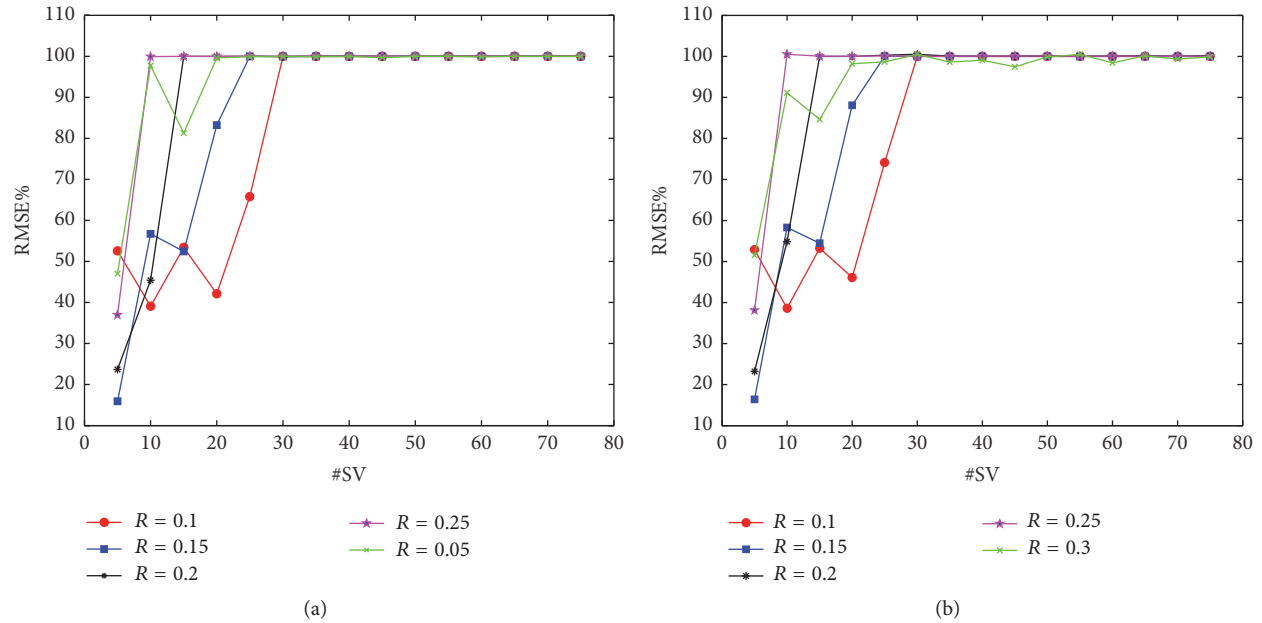

Figure 1: RMSE% of RCLSSVR with different r: (a) RMSE% on training data set, (b) RMSE% on testing data set.

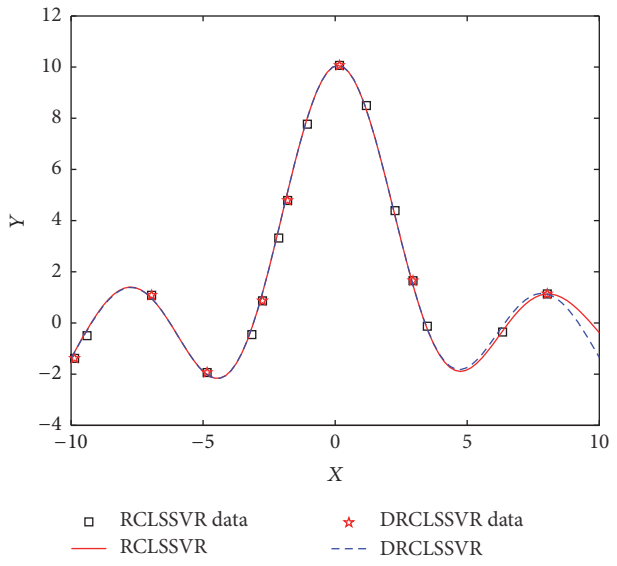

Figure 2: The comparison between RCLSSVR and DRCLSSVR.

performance between different sparse algorithms mentioned above to show the robustness and sparseness of RCLSSVR and DRCLSSVR. The input and output variables of each data set are normalized into closed interval $[0, 1]$. We firstly generate some synthetic data sets according to the properties of real data set to study the heteroscedastic and imbalanced problems. The heteroscedastic data set is generated by the sinc function with two kinds of noise: multiplication noise and heterogeneous variant Gaussian noise. The imbalanced data set also is generated by (20). The difference is that the numbers of samples in different interval are not the same. In addition, the testing data set is directly generated by the sinc function. These data sets are illustrated in Figure 3.

When test data set comes from the original sinc function relative to the original training data set, the accuracy of prediction model is more sensitive. We gradually increase the support vectors to investigate the robustness of RCLSSVR. The results are shown in Figure 4 and Table 1, which can illustrate that the generalization results of RCLSSVR have better robustness than these contrast algorithms. The performance of IRRLSSVM and SLSSVM suffers on heteroscedastic data sets and do well on imbalanced data set. The performance of PLSSVM and RCLSSVR obtains very good results on above-mentioned data sets. However, the PLSSVM are highly time-consuming, especially for large sample size of data. And it is just for comparing the sparse performance in this paper, not as a selected algorithm. The RCLSSVR has good sparse properties as well as suitable time complexity. The most important is that RCLSSVR will have the best performance and robustness after passing through the training data sets, which is little affected when support vectors increase.

Next, we conduct this experiment on some benchmark regression data sets to investigate the sparseness of RCLSSVR and DRCLSSVR, where the data sets Chwirut1 and Nelson are downloaded from http://www.itl.nist.gov/div898/strd/nls/nls_main.shtml; the Housing, Motorcycle, Airfoil_self_noise, and Yacht_hydrodynamics are downloaded from http://archive.ics.uci.edu/ml/datasets.html; the data sets MPG, Pyrim, and Bodyfat are downloaded from http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/. The data sets also contain the feature of heteroscedasticity, imbalance, and high dimension, and the detailed information about these data sets is listed in Table 2.

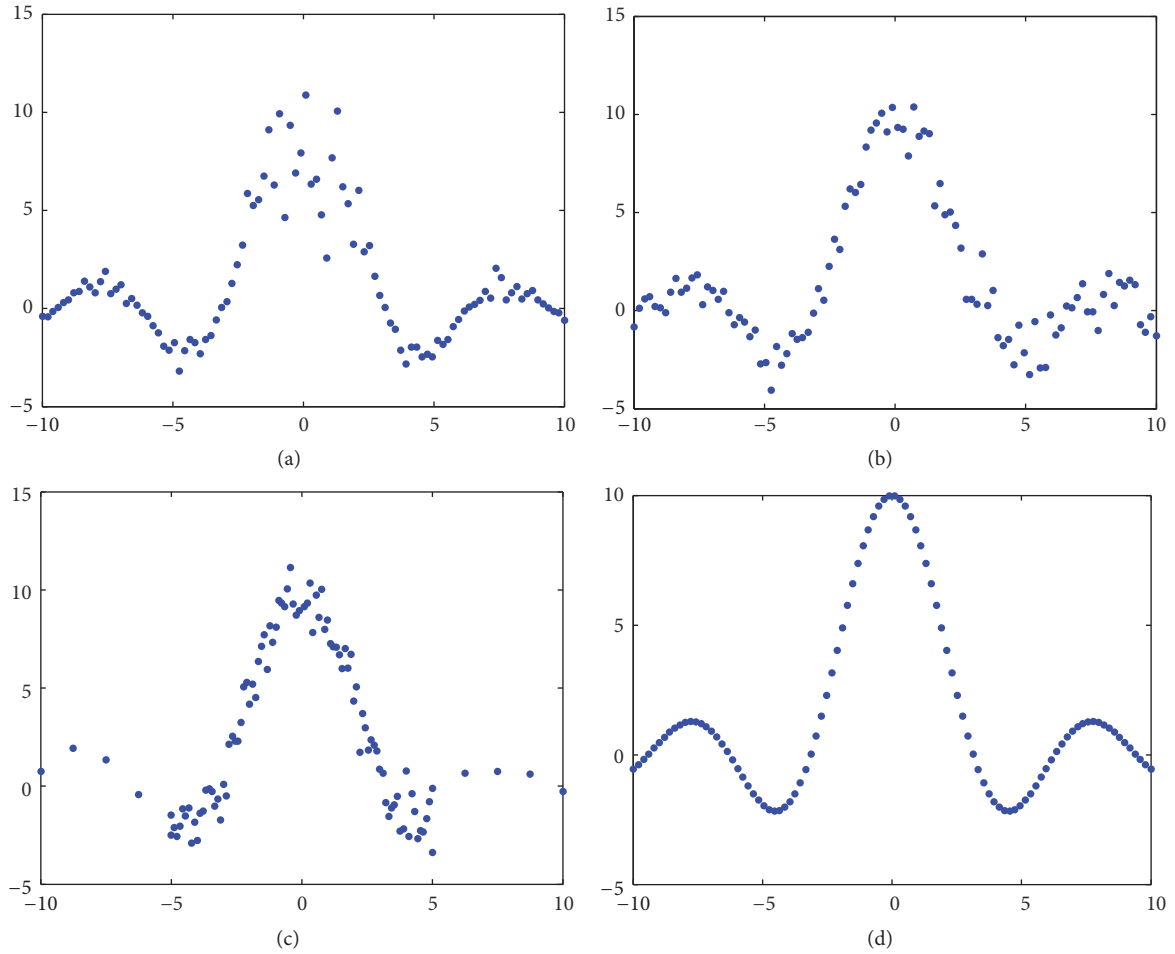

Figure 3: Three synthetic data sets: (a) synthetic data set one with multiplicative noise $y = (1 + \nu) \times y$, where ν is random noise whose mean is zero and variance is 0.1; (b) synthetic data set two with heterogeneous variant Gaussian noise, where noise with zero mean and the variance is 0.4 and 1.2, respectively on $[-10, 0]$ and $[0, 10]$; (c) synthetic data set three is unbalanced data set, which distribute on the interval $[-5, 5]$ about 90% data and 5%, respectively, on $[-10, -5]$ and $[5, 10]$; (d) the testing data set. The number of training samples and testing samples is 100.

Table 1: Compare the performance, among these algorithms; NLSSVM with 100 data points, the others with 15 #SV.

Data set	Synthetic data set one	Synthetic data set two	Synthetic data set three
NLSSVM	**0.0174**	0.0049	0.0558
RCLSSVM	0.0183	**0.0048**	**0.0548**
SLSSVM	10.4719	0.0752	1.3758
PLSSVM	0.0162	0.0038	0.0596
IRRLSSVM	0.0564	0.0088	0.0626

The comparison results for these benchmark data sets are tabulated in Table 3, where the #SV column gives the number of training samples. Due to the performance closer to the value of normal LSSVM as #SV increases, we will compare the number of support vectors when the performance reaches 90% of the normal LSSVM. The forward algorithms will be stopped when the performance exceeds 90%. For the backward algorithms, the process of training will be terminated after the performance becomes less than 90%. Besides, the RMSE is the performance of testing data set, and the parameters γ_1 and σ_1 are optimized on original training data set; the parameters γ_2 and σ_2 are the final optimization results for the RCLSSVR. From Table 3, we can conclude that the proposed algorithms have better sparseness and robustness than other compared methods. As expected, PLSSVM has outstanding performance on RMSE than SLSSVM, but the training time is too expensive. SLSSVM is relatively unfavorable and performs worse in high dimensionality. IRRLSSVM

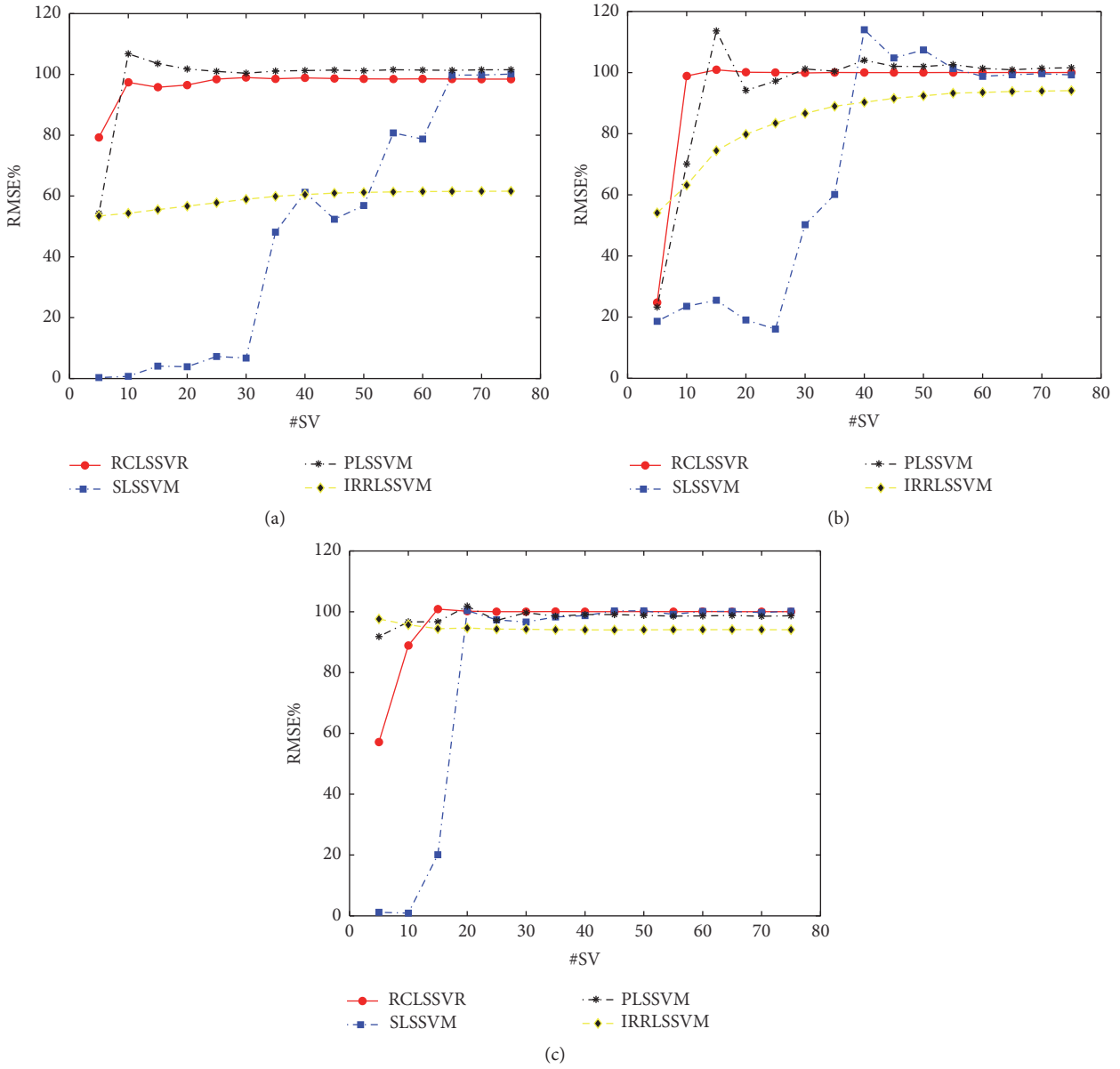

Figure 4: Generalization performance of RCLSSVR with different data sets: (a) synthetic data set one; (b) synthetic data set two; (c) synthetic data set three.

Table 2: Detailed information of benchmark data sets.

Data set	Dimensionality	Training data	Testing data
Chwirut1	1	150	64
Nelson	2	90	38
Boston Housing	13	400	106
Bodyfat	14	180	71
Pyrim	28	60	14
Yacht_hydrodynamics	6	210	99
Airfoil_self_noise	5	1200	303
Motorcycle	1	100	33
MPG	7	300	92

Table 3: Experimental results on benchmark data sets.

Data sets	Algorithms	RMSE	SV	Training time (s)	Testing time (s)
Chwirut1 $\gamma_1 = 5.1539,\ \delta_1^2 = 0.0036$ $\gamma_2 = 3.0e+06,\ \delta_2^2 = 0.3$	NLSSVM	0.014 ± 0.001	150	0.043 ± 0.009	0.023 ± 0.004
	RCLSSVR	0.015 ± 0.002	12	0.241 ± 0.072	0.001 ± 0.000
	DRCLSSVR	**0.016 ± 0.001**	**9**	**0.250 ± 0.076**	**0.001 ± 0.000**
	SLSSVM	0.017 ± 0.002	42	0.484 ± 0.051	0.001 ± 0.000
	PLSSVM	0.017 ± 0.000	12	43.094 ± 1.377	0.001 ± 0.000
	IRRLSSVM	0.016 ± 0.002	16	0.313 ± 0.045	0.001 ± 0.000
Nelson $\gamma_1 = 13.145,\ \delta_1^2 = 0.100$ $\gamma_2 = 53.739,\ \delta_2^2 = 0.013$	NLSSVM	0.016 ± 0.001	90	0.020 ± 0.006	0.005 ± 0.004
	RCLSSVR	0.016 ± 0.001	16	0.376 ± 0.014	0.001 ± 0.000
	DRCLSSVR	**0.017 ± 0.002**	**11**	**0.412 ± 0.036**	**0.001 ± 0.000**
	SLSSVM	0.019 ± 0.005	50	0.219 ± 0.043	0.003 ± 0.001
	PLSSVM	0.019 ± 0.001	130	12.971 ± 1.364	0.002 ± 0.000
	IRRLSSVM	0.018 ± 0.000	16	0.166 ± 0.006	0.001 ± 0.000
Boston Housing $\gamma_1 = 554.003,\ \delta_1^2 = 1.976$ $\gamma_2 = 3.41e+03,\ \delta_2^2 = 2.76$	NLSSVM	0.012 ± 0.001	400	0.062 ± 0.012	0.023 ± 0.006
	RCLSSVR	0.012 ± 0.001	66	2.660 ± 0.107	0.003 ± 0.001
	DRCLSSVR	**0.014 ± 0.001**	**48**	**2.705 ± 0.030**	**0.002 ± 0.001**
	SLSSVM	0.015 ± 0.001	223	1.680 ± 0.127	0.005 ± 0.001
	PLSSVM	0.014 ± 0.000	92	497.73 ± 16.52	0.004 ± 0.001
	IRRLSSVM	0.014 ± 0.001	62	3.680 ± 0.221	0.002 ± 0.001
Bodyfat $\gamma_1 = 10e+05,\ \delta_1^2 = 20$ $\gamma_2 = 1.12e+06,\ \delta_2^2 = 5$	NLSSVM	0.004 ± 0.002	180	0.024 ± 0.000	0.004 ± 0.000
	RCLSSVR	0.004 ± 0.002	18	0.407 ± 0.153	0.001 ± 0.000
	DRCLSSVR	**0.004 ± 0.002**	**14**	**0.415 ± 0.171**	**0.001 ± 0.000**
	SLSSVM	0.006 ± 0.002	50	0.740 ± 0.054	0.001 ± 0.000
	PLSSVM	0.005 ± 0.001	28	45.904 ± 6.650	0.001 ± 0.000
	IRRLSSVM	0.004 ± 0.002	21	0.451 ± 0.049	0.001 ± 0.000
Pyrimt $\gamma_1 = 10.82,\ \delta_1^2 = 1.80$ $\gamma_2 = 9e+06,\ \delta_2^2 = 16.27$	NLSSVM	0.125 ± 0.027	60	0.016 ± 0.008	0.004 ± 0.001
	RCLSSVR	0.135 ± 0.007	9	0.064 ± 0.003	0.001 ± 0.000
	DRCLSSVR	0.136 ± 0.007	8	0.068 ± 0.001	0.001 ± 0.000
	SLSSVM	0.158 ± 0.071	16	0.343 ± 0.003	0.001 ± 0.000
	PLSSVM	**0.167 ± 0.016**	**4**	**4.397 ± 0.162**	**0.001 ± 0.000**
	IRRLSSVM	0.137 ± 0.001	8	0.487 ± 0.006	0.001 ± 0.001
Yacht_hydrodynamics $\gamma_1 = 100,\ \delta_1^2 = 2$ $\gamma_2 = 4e+05,\ \delta_2^2 = 3.20$	NLSSVM	0.006 ± 0.001	210	0.063 ± 0.011	0.026 ± 0.007
	RCLSSVR	0.006 ± 0.001	39	0.543 ± 0.175	0.002 ± 0.001
	DRCLSSVR	**0.010 ± 0.002**	**37**	**0.600 ± 0.150**	**0.002 ± 0.001**
	SLSSVM	0.107 ± 0.001	73	0.828 ± 0.212	0.003 ± 0.001
	PLSSVM	0.007 ± 0.001	62	80.567 ± 2.001	0.003 ± 0.001
	IRRLSSVM	0.012 ± 0.001	91	1.292 ± 0.306	0.004 ± 0.001
Airfoil_self_noise $\gamma_1 = 10.00,\ \delta_1^2 = 0.30$ $\gamma_2 = 3.5e+07,\ \delta_2^2 = 1.78$	NLSSVM	0.010 ± 0.006	1200	0.242 ± 0.013	0.004 ± 0.012
	RCLSSVR	0.011 ± 0.010	78	23.255 ± 3.245	0.003 ± 0.001
	DRCLSSVR	**0.011 ± 0.010**	**75**	**23.506 ± 3.412**	**0.003 ± 0.001**
	SLSSVM	0.012 ± 0.004	332	40.715 ± 4.671	0.004 ± 0.001
	PLSSVM	0.012 ± 0.003	94	$2e+04 \pm 5e+03$	0.003 ± 0.001
	IRRLSSVM	0.012 ± 0.001	116	38.617 ± 2.313	0.004 ± 0.001
Motorcycle $\gamma_1 = 54.809,\ \delta_1^2 = 0.019$ $\gamma_2 = 2.03e+03,\ \delta_2^2 = 0.013$	NLSSVM	0.021 ± 0.004	100	0.015 ± 0.005	0.002 ± 0.000
	RCLSSVR	0.021 ± 0.004	10	0.072 ± 0.009	0.001 ± 0.000
	DRCLSSVR	**0.023 ± 0.004**	**7**	**0.080 ± 0.016**	**0.001 ± 0.000**
	SLSSVM	0.032 ± 0.006	56	0.203 ± 0.073	0.002 ± 0.000
	PLSSVM	0.024 ± 0.001	10	16.366 ± 2.044	0.001 ± 0.000
	IRRLSSVM	**0.022 ± 0.001**	**7**	**0.062 ± 0.012**	**0.001 ± 0.000**

Table 3: Continued.

Data sets	Algorithms	RMSE	SV	Training time (s)	Testing time (s)
MPG $\gamma_1 = 8.985,\ \delta_1^2 = 0.381$ $\gamma_2 = 5.72e+05,\ \delta_2^2 = 0.380$	NLSSVM	0.011 ± 0.003	300	0.016 ± 0.007	0.005 ± 0.002
	RCLSSVR	0.012 ± 0.005	24	0.517 ± 0.011	0.001 ± 0.001
	DRCLSSVR	**0.013 ± 0.004**	**14**	**0.591 ± 0.017**	**0.001 ± 0.000**
	SLSSVM	0.022 ± 0.014	65	1.160±0.270	0.001 ± 0.000
	PLSSVM	0.016 ± 0.002	17	228.07 ± 10.36	0.001 ± 0.000
	IRRLSSVM	0.012 ± 0.001	21	0.437 ± 0.007	0.001 ± 0.000

Table 4: Experimental results on mill load data set.

Algorithms	#SV	RMSE	Training times	Testing times
NLSSVM	600	$2.4014e-04$	1.1820	0.1080
RCLSSVR	70	$2.4651e-04$	4.3930	0.0083
DRCLSSVR	**36**	**2.5139e − 04**	**5.1670**	**0.0066**
SLSSVM	230	$2.4897e-04$	9.2530	0.0265
PLSSVM	130	$2.4724e-04$	$3.5078e+03$	0.0137
IRRLSSVM	140	$6.4476e-04$	17.0970	0.0161

may suffer on the scattered and high dimension data set and is more susceptible to the parameter effects. But the testing time is less than other algorithms when having the same #SV as it does not need to build the model. It is very obvious that the fewer #SV, the less the testing time, and different algorithms have different training time. In practical application, the training time and testing time should be kept in the allowable range, because the sparse algorithms are trained out of line and used online.

4.3. Experiment 3: The Simulation on Mill Load Data Sets. Mill load has not been accurately measured suffering the influence of numerous factors, which is critical in increasing efficiency and energy efficiency of pulverizing system. Paper [34] proposed that the mill load can be represented by mill noise (E_n), mill vibration (E_v), mill current (I), import-export pressure difference (P_D), and export temperature (T). The estimation of mill load is formulated as

$$y_m = f\left(E_n, E_v, I, P_D, T\right). \tag{22}$$

The mill load data set, consisting of 2400 samples, covers three conditions of low load, normal load, and high load. The training and testing sets randomly selected 600 samples and 200 samples, and the model parameters are optimized by the 10-fold cross validation. The comparison results between these sparse algorithms, showed in Figure 5, are coincident with the experimental results on benchmark data sets. RCLSSVR need less #SV to reach the normal performance, with the shorter testing time in the prediction phase. The estimation results of RCLSSVR and DRCLSSVR are illustrated in Figure 6. As can be seen from Figure 6, the testing data set contains different wear conditions, and the model build by RCLSSVR has good estimation performance in all conditions. It is obvious that RCLSSVR considers both sparseness and robustness in the training process and is more conducive to real industrial data sets. The experimental details are tabulated in Table 4.

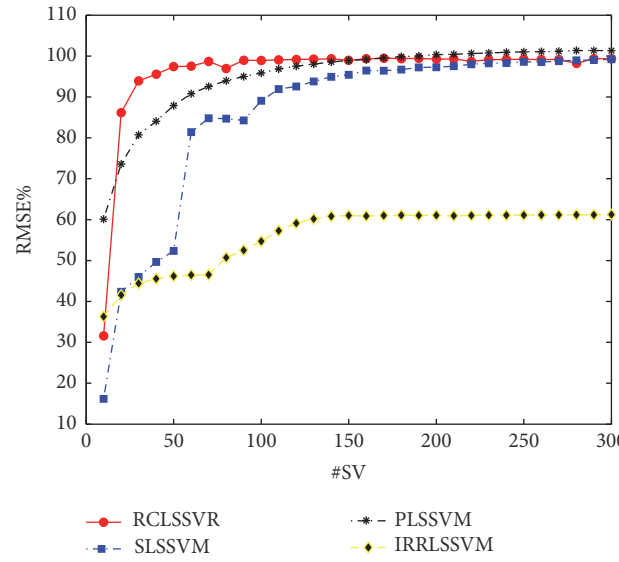

Figure 5: RMSE% against #SV on mill load data set.

5. Conclusion

Aiming at the problem of LSSVM sparseness and robustness, this paper proposes two sparse algorithms called RCLSSVR and DRCLSSVR via reconstructing the support vectors based on training data set and target function. RCLSSVR selects reconstructed data according to the location information of density clustering in training data set and tweaks the

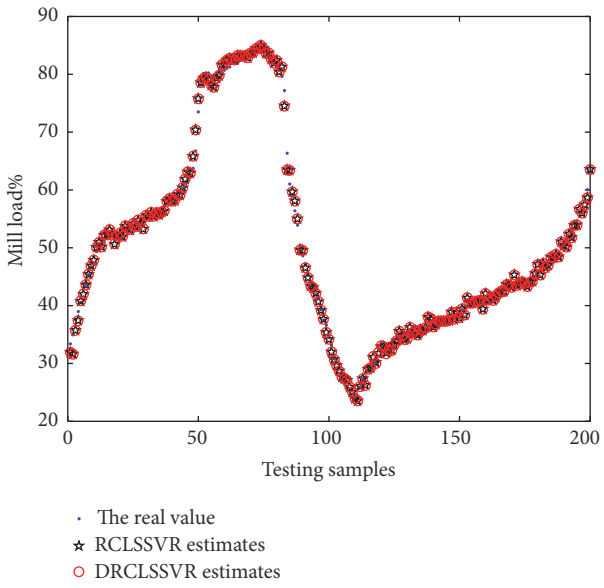

Figure 6: The estimation output of testing data set.

hyperparameter by optimizing the support vector in the training process. DRCLSSVR is proposed for subtracting the redundancy support vectors of RCLSSVR. In order to demonstrate the effectiveness of RCLSSVR and DRCLSSVR, several experiments on function approximation are carried out using sinc data set and benchmark data sets. The experimental results demonstrated that the proposed methods are superior to algorithms compared, not only in the precision of RMSE, but also on the number of support vectors. Finally, RCLSSVR and DRCLSSVR are applied to the mill load prediction and achieve good performance of sparseness and robustness.

Abbreviations

LSSVM:	Least square support vector machine
RCLSSVR:	Reconstructed LSSVM
DRCLSSVR:	The improved RCLSSVR
SLSSVM:	Suykens pruning algorithm [10]
PLSSVM:	Backward classic algorithm [12]
IRRLSSVM:	Forward sparse algorithm [19]
RBF:	Radial basis function
RMSE:	Rooted mean squared errors
RMSE%:	The ratio of RMSE
$\{x_i, y_i\}_{i=1}^N$:	The training data set
$\{x_i, \hat{y}_i\}_{i=1}^N$:	The reconstructed data set
$\{x_d, y_d\}$:	The density center
$\{x_d, \hat{y}_d\}$:	The support vector
(δ, γ):	The RBF hyperparameter
Dist:	The Euclidean distance vector
Den:	The density indicators for support vectors
r:	The ratio of maximum distance
#SV:	The number of support vectors.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by National Natural Science Foundation of China (Grant no. 61304118), Program for New Century Excellent Talents in University (NCET-13-0456), and the Specialized Research Fund for Doctoral Program of Higher Education of China (Grant no. 20130201120011).

References

[1] P. Huang, M.-P. Jia, and B.-L. Zhong, "Investigation on measuring the fill level of an industrial ball mill based on the vibration characteristics of the mill shell," *Minerals Engineering*, vol. 22, no. 14, pp. 1200–1208, 2009.

[2] J. Tang, W. Yu, T. Chai, Z. Liu, and X. Zhou, "Selective ensemble modeling load parameters of ball mill based on multi-scale frequency spectral features and sphere criterion," *Mechanical Systems and Signal Processing*, vol. 66-67, pp. 485–504, 2016.

[3] G. Si, H. Cao, Y. Zhang, and L. Jia, "Experimental investigation of load behaviour of an industrial scale tumbling mill using noise and vibration signature techniques," *Minerals Engineering*, vol. 22, no. 15, pp. 1289–1298, 2009.

[4] J. Tang, T. Chai, W. Yu, Z. Liu, and X. Zhou, "A comparative study that measures ball mill load parameters through different single-scale and multiscale frequency spectra-based approaches," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 6, pp. 2008–2019, 2016.

[5] S. J. Rutherford and D. J. Cole, "Modelling nonlinear vehicle dynamics with neural networks," *International Journal of Vehicle Design*, vol. 53, no. 4, pp. 260–287, 2010.

[6] Y. Bengio, N. Chapados, O. Delalleau et al., "Detonation classification from acoustic signature with the restricted Boltzmann machine," *Computational Intelligence*, vol. 28, no. 2, pp. 261–288, 2012.

[7] Q.-B. Li, H.-L. Yan, L.-N. Li, J.-G. Wu, and G.-J. Zhang, "Application of partial robust M-regression in noninvasive measurement of human blood glucose concentration with near-infrared spectroscopy," *Spectroscopy and Spectral Analysis*, vol. 30, no. 8, pp. 2115–2119, 2010.

[8] L. Chen and H. Liu, "Application of LS-SVM in fault diagnosis for diesel generator set of marine power station," in *Proceedings of the 2013 International Conference on Advanced Computer Science and Electronics Information (Icacsei 2013)*, vol. 41, pp. 101–104, 2013.

[9] J. A. K. Suykens, J. De Brabanter, L. Lukas, and J. Vandewalle, "Weighted least squares support vector machines: robustness and sparce approximation," *Neurocomputing*, vol. 48, no. 1, pp. 85–105, 2002.

[10] J. A. K. Suykens, L. Lukas, and J. Vandewalle, "Sparse approximation using least squares support vector machines," in *Proceedings of the 2000 IEEE International Symposium on Circuits and Systems, Emerging Technologies for the 21st Century (ISCAS '00)*, vol. 2, pp. 757–760, International Conference Center, Geneva, Switzerland, May 2000.

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

[12] 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 *Proceedings of the 11th International Conference Neural Information Processing (ICONIP '04)*,

vol. 3316 of *Lecture Notes in Computer Science*, pp. 1247–1253, Springer Berlin Heidelberg, Calcutta, India, January 2004.

[13] X. Y. Zeng and X. W. Chen, "SMO-based pruning methods for sparse least squares support vector machines," *IEEE Transactions on Neural Networks*, vol. 16, no. 6, pp. 1541–1546, 2005.

[14] H. Y. Song, W. H. Gui, and C. H. Yang, "Sparse least squares support vector machine and its applications," *Information and Control*, vol. 37, no. 3, pp. 334–338, 2008.

[15] Z.-T. Yu, J.-J. Zou, X. Zhao, L. Su, and C.-L. Mao, "Sparseness of least squares support vector machines based on active learning," *Journal of Nanjing University of Science and Technology*, vol. 36, no. 1, pp. 12–17, 2012.

[16] L. C. Jiao, L. F. Bo, and L. Wang, "Fast sparse approximation for least squares support vector machine," *IEEE Transactions on Neural Networks*, vol. 18, no. 3, pp. 685–697, 2007.

[17] Z. Yongping and S. Jianguo, "Fast method for sparseleast squares support vector regression machine," *Control and Decision*, vol. 23, no. 12, pp. 1347–1352, 2008.

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

[19] 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, 2012.

[20] L. G. Sun, C. C. de Visser, Q. P. Chu, and J. A. Mulder, "A novel online adaptive kernel method with kernel centers determined by a support vector regression approach," *Neurocomputing*, vol. 124, pp. 111–119, 2014.

[21] P. B. Nair, A. Choudhury, and A. J. Keane, "Some greedy learning algorithms for sparse regression and classification with Mercer kernels," *The Journal of Machine Learning Research*, vol. 3, pp. 781–801, 2003.

[22] J. P. Silva and A. R. Da Rocha Neto, "Sparse least squares support vector machines via genetic algorithms," in *Proceedings of the 11th Brazilian Congress on Computational Intelligence (BRICS '13)*, pp. 248–253, IEEE, Ipojuca, Brazil, September 2013.

[23] D. A. Silva, J. P. Silva, and A. R. Rocha Neto, "Novel approaches using evolutionary computation for sparse least square support vector machines," *Neurocomputing*, vol. 168, pp. 908–916, 2015.

[24] 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, 2014.

[25] Z. Ying and K. C. Keong, "Fast leave-one-out evaluation and improvement on inference for LS-SVMs," in *Proceedings of the 17th International Conference on Pattern Recognition (ICPR '04)*, vol. 3, pp. 494–497, Cambridge, UK, August 2004.

[26] P. Cao, X. Liu, J. Zhang et al., "2, 1 norm regularized multi-kernel based joint nonlinear feature selection and over-sampling for imbalanced data classification," *Neurocomputing*, vol. 234, pp. 38–57, 2017.

[27] J. Zhang, X. Wu, and V. S. Sheng, "Imbalanced multiple noisy labeling," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 2, pp. 489–503, 2015.

[28] R. Yan, Y. Liu, R. Jin, and A. Hauptmann, "On predicting rare classes with SVM ensembles in scene classification," in *Proceedings of the 2003 IEEE International Conference on Accoustics, Speech, and Signal Processing*, pp. 21–24, Hong Kong, China, April 2003.

[29] S.-J. Yen and Y.-S. Lee, *Cluster-Based under-Sampling Approaches for Imbalanced Data Distributions*, Pergamon Press, Oxford, UK, 2009.

[30] B. Das, N. C. Krishnan, and D. J. Cook, "RACOG and wRACOG: Two probabilistic oversampling techniques," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 1, pp. 222–234, 2015.

[31] H. Wong, F. Liu, M. Chen, and W. C. Ip, "Empirical likelihood based diagnostics for heteroscedasticity in partially linear errors-in-variables models," *Journal of Statistical Planning and Inference*, vol. 139, no. 3, pp. 916–929, 2009.

[32] B. Lejeune, "A diagnostic m-test for distributional specification of parametric conditional heteroscedasticity models for financial data," *Journal of Empirical Finance*, vol. 16, no. 3, pp. 507–523, 2009.

[33] T. Ginker and O. Lieberman, "Robustness of binary choice models to conditional heteroscedasticity," *Economics Letters*, vol. 150, pp. 130–134, 2017.

[34] G. Q. Si, H. Cao, Y. B. Zhang, and L. X. Jia, "Density weighted pruning method for sparse least squares support vector machines," *Xi'an Jiaotong Daxue Xuebao. Journal of Xi'an Jiaotong University*, vol. 43, no. 10, pp. 11–15, 2009.