



# Wavelet transform-based weighted $\nu$ -twin support vector regression

Lidong Wang<sup>1</sup> · Chuang Gao<sup>1</sup> · Nannan Zhao<sup>1</sup> · Xuebo Chen<sup>1</sup>

Received: 3 September 2018 / Accepted: 23 April 2019 / Published online: 2 May 2019  
© Springer-Verlag GmbH Germany, part of Springer Nature 2019

## Abstract

In this paper, an efficient wavelet transform-based weighted  $\nu$ -twin support vector regression (WTWTSVR) is proposed, inspired by twin support vector regression (TSVR) and  $\nu$ -twin support vector machine-based regression. TSVR and its improved models work faster than support vector regression because they solve a pair of smaller-sized quadratic programming problems. However, they give the same emphasis to the training samples, i.e., all of the training samples share the same weights, and prior information is not used, which leads to the degradation of performance. Motivated by this, samples in different positions in the proposed WTWTSVR model are given different penalty weights determined by the wavelet transform. The weights are applied to both the quadratic empirical risk term and the first-degree empirical risk term to reduce the influence of outliers. The final regressor can avoid the overfitting problem to a certain extent and yield great generalization ability. Numerical experiments on artificial datasets and benchmark datasets demonstrate the feasibility and validity of our proposed algorithm.

**Keywords** Machine learning · Support vector regression · Twin support vector regression · Wavelet transform

## 1 Introduction

Artificial intelligence is the development of traditional control methods or learning methods. In recent years, artificial intelligence methods have been extensively studied and applied [1–6]. The support vector machine (SVM) is proposed based on the principle of statistical learning theory and Vapnik–Chervonenkis (VC) dimensional theory, which is a promising machine learning approach that has been adopted in classification and regression [7, 8]. The adoption of the kernel trick can extend the algorithm for use in nonlinear cases. The basic idea of the SVM is to construct a hyperplane that can partition the samples from different

classes with minimum generalization error by maximizing the margin. Compared with other supervised learning methods, it has many strong points, such as being with sparse solution constructed by ‘support vectors’, having generalization error minimization, and needing only a small-scale dataset. It has become one of the most powerful tools for pattern classification and regression [9].

To improve the performance of the standard SVM, many modified methods were proposed. Jayadeva et al. [10] proposed a twin SVM (TSVM) by seeking two non-parallel bound functions, and each one is close to one class of samples and as far as possible from the other. The TSVM can increase the computational speed by solving two small-size quadratic programming problems (QPPs) rather than a large one as in the standard SVM. Later, Peng [11] extended this strategy to regression applications, resulting in the twin support vector regression (TSVR); the target of the regression problem is to get the relationship between inputs and their corresponding outputs. Solving the SVM is based on QPP problems subject to linear inequality constraints, which is with heavy computational costs. Suyken et al. [12, 13] proposed the least squares SVM (LS-SVM) for large-scale dataset problems. It follows a similar spirit as that of the SVM except for the constraints being linear equalities, which can reduce the

---

✉ Xuebo Chen  
x-bchen@163.com

Lidong Wang  
wangld5600@163.com

Chuang Gao  
13500422153@163.com

Nannan Zhao  
723306003@qq.com

<sup>1</sup> School of Electronic and Information Engineering,  
University of Science and Technology Liaoning, Anshan,  
Liaoning, People’s Republic of China

computational complexity significantly. The  $\nu$ -support vector regression ( $\nu$ -SVR) [14] extends the standard SVR via enforcing a fraction of the data samples to lie inside an  $\epsilon$  tube, and minimizing the width of the tube by introducing a parameter  $\nu$ . Inspired by the  $\nu$ -SVR and combined with the concept of pinball loss [15–18]. Xu [19] proposed asymmetric  $\nu$ -twin support vector regression (Asy- $\nu$ -TSVR), which is a kind of twin SVR suitable for dealing with asymmetric noise. However, these methods only take into account the empirical risk minimization rather than the structural risk minimization, which will produce the overfitting problem. To overcome the overfitting problem, Shao et al. [20] introduced regularization terms into the objective functions of TSVR. Recently, Rastogi et al. proposed a  $\nu$ -twin support vector machine-based regression method ( $\nu$ -TWSVR) [21]: the regularization terms and the insensitive bound values  $\epsilon$  in the method were all added into the objective functions to minimize the risks, which can adjust the values of  $\epsilon$  automatically. However, in the methods mentioned above, such as TSVR and  $\nu$ -TWSVR, all of the training samples are considered as having the same status and are given the same penalties, which may degrade the performance due to the influence of noise. Therefore, it is useful to give the training samples different weights depending on their importance. Xu et al. [22] proposed the K-nearest neighbor (KNN)-based weighted twin support vector regression, in which the local information of data was used for improving the prediction accuracy. KNN-based methods are suitable for regression situations with clustered samples but not for time series due to the nature of the KNN.

In this paper, a novel TSVR-based regression model termed WTWTSVR is proposed. Some of the features of the proposed WTWTSVR are as follows.

1. The prior information of the training samples is incorporated into the proposed algorithm. A weight matrix is introduced into the quadratic empirical risks term, and a weight vector is introduced into the first-degree empirical risks term to reduce the influence of outliers.
2. Wavelet transform theory is adopted to calculate the weight matrix, which is a new angle for preprocessing samples. Wavelet transform is a kind of time-frequency representation for signals, and the proposed method based on the wavelet theory is suitable for dealing with time series samples due to the character of the wavelet transform.
3. The proposed model also takes into account the regularization terms and the insensitive bound values  $\epsilon$  as penalty terms. The effectiveness of our WTWTSVR is demonstrated by experiments on artificial datasets and benchmark datasets. The numerical results show that the

proposed algorithm has better generalization ability than traditional ones.

This paper is organized as follows: Sect. 2 briefly describes TSVR and  $\nu$ -TWSVR. Section 3 proposes our WTWTSVR. Experimental results are described in Sect. 4 to investigate the validity of our proposed algorithm, and Sect. 5 ends the paper with concluding remarks.

## 2 Brief introduction to TSVR and $\nu$ -TWSVR

In this section, the linear TSVR and  $\nu$ -TWSVR algorithms are described briefly. Nonlinear cases can be solved by Kernel methods. Given a training set  $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$ , where  $x_i \in \mathfrak{R}^n$  and  $y_i \in \mathfrak{R}$ ,  $i = 1, 2, \dots, m$ . Then the output vector of the training data can be denoted as  $Y = (y_1, y_2, \dots, y_m)^T \in \mathfrak{R}^m$  and the input matrix as  $A = (A_1, A_2, \dots, A_m)^T \in \mathfrak{R}^{m \times n}$ , and the  $i$ th row  $A_i = (x_{i1}, x_{i2}, \dots, x_{in})^T$  is the  $i$ th training sample. We let  $e$  and  $I$  be a ones vector and an identity matrix of appropriate dimensions, respectively.

### 2.1 Twin support vector regression (TSVR)

TSVR proposed by Peng [11] is an efficient regression method with low computational costs compared with SVR [23, 24]. TSVR has two  $\epsilon$ -insensitive bounds: the down-bound is denoted as  $f_1(x) = w_1^T x + b_1$ , the up-bound is denoted as  $f_2(x) = w_2^T x + b_2$ , and the regressor  $f(x)$  can be obtained by taking the mean of the two bound functions, i.e.

$$\begin{aligned} f(x) &= \frac{1}{2}(f_1(x) + f_2(x)) \\ &= \frac{1}{2}(w_1 + w_2)^T x + \frac{1}{2}(b_1 + b_2). \end{aligned} \quad (1)$$

The two  $\epsilon$ -insensitive bound functions can be obtained by solving the following pair of QPPs:

$$\begin{aligned} \min_{w_1, b_1, \xi_1} & \frac{1}{2} \|Y - \epsilon e_1 - (Aw_1 + eb_1)\|^2 + C_1 e^T \xi_1 \\ \text{s.t.} & Y - (Aw_1 + eb_1) \geq \epsilon e_1 - \xi_1, \xi_1 \geq 0, \end{aligned} \quad (2)$$

and

$$\begin{aligned} \min_{w_2, b_2, \xi_2} & \frac{1}{2} \|Y + \epsilon e_2 - (Aw_2 + eb_2)\|^2 + C_2 e^T \xi_2 \\ \text{s.t.} & (Aw_2 + eb_2) - Y \geq \epsilon e_2 - \xi_2, \xi_2 \geq 0, \end{aligned} \quad (3)$$

where  $\epsilon_1, \epsilon_2 \geq 0$  are insensitive parameters, which are introduced to construct the  $\epsilon$ -insensitive zone, ' $\epsilon$  tube'.  $\xi_1$  and  $\xi_2$  are slack vectors to measure the errors of samples outside the ' $\epsilon$  tube'.  $C_1, C_2 > 0$  are parameters used to make the trade-off between the fitting errors and the slack errors. After

introducing the Lagrangian functions of (2) and (3), and considering the corresponding Karush–Kuhn–Tucker (KKT) optimality conditions, the dual QPPs can be obtained as

$$\min_{\alpha} \frac{1}{2} \alpha^T G(G^T G)^{-1} G^T \alpha - f_1^T G(G^T G)^{-1} G^T \alpha + f_1^T \alpha \tag{4}$$

s.t.  $0e \leq \alpha \leq C_1 e$ ,

and

$$\min_{\beta} \frac{1}{2} \beta^T G(G^T G)^{-1} G^T \beta + f_2^T G(G^T G)^{-1} G^T \beta - f_2^T \beta \tag{5}$$

s.t.  $0e \leq \beta \leq C_2 e$ ,

where  $G = [A e]$ ,  $f_1 = Y - e\epsilon_1$ ,  $f_2 = Y + e\epsilon_2$ , and  $\alpha, \beta \in \mathfrak{R}^m$  are Lagrangian multiplies vectors.

After solving the dual QPPs (4) and (5), we can get solutions  $\alpha$  and  $\beta$ , and then the regression parameters can be obtained as  $[w_1^T, b_1]^T = (G^T G)^{-1} G^T (f_1 - \alpha)$  and  $[w_2^T, b_2]^T = (G^T G)^{-1} G^T (f_2 + \beta)$ .

### 2.2 $\nu$ -Twin support vector regression ( $\nu$ -TWSVR)

Rastogi proposed a modification of TSVR. Similar to the TSVR model,  $\nu$ -TWSVR also solves a pair of optimization problems (6) and (7):

$$\min_{w_1, b_1, \xi_1, \epsilon_1} \frac{1}{2} \|Y - (Aw_1 + eb_1)\|^2 + \frac{1}{2} c_1 (w_1^T w_1 + b_1^2) + c_2 \left( \nu_1 \epsilon_1 + \frac{1}{m} e^T \xi_1 \right) \tag{6}$$

s.t.  $Y - (Aw_1 + eb_1) \geq -\epsilon_1 e - \xi_1$   
 $\xi_1 \geq 0e, \epsilon_1 \geq 0$ ,

and

$$\min_{w_2, b_2, \xi_2, \epsilon_2} \frac{1}{2} \|Y - (Aw_2 + eb_2)\|^2 + \frac{1}{2} c_3 (w_2^T w_2 + b_2^2) + c_4 \left( \nu_2 \epsilon_2 + \frac{1}{m} e^T \xi_2 \right) \tag{7}$$

s.t.  $(Aw_2 + eb_2) - Y \geq -\epsilon_2 e - \xi_2$   
 $\xi_2 \geq 0e, \epsilon_2 \geq 0$ ,

where  $\nu_1, \nu_2, c_1, c_2, c_3, c_4 > 0$  are user-specified parameters, and  $\epsilon_1, \epsilon_2, \xi_1$ , and  $\xi_2$  are the same as in TSVR. Different from the TSVR model, the insensitive parameters  $\epsilon_1$  and  $\epsilon_2$  are not given parameters but are optimized. After introducing the Lagrangian functions of (6) and (7), and considering the KKT conditions, the dual QPPs can be obtained as

$$\min_{\alpha} \frac{1}{2} \alpha^T G(G^T G + c_1 I)^{-1} G^T \alpha - Y^T G(G^T G + c_1 I)^{-1} G^T \alpha + Y^T \alpha \tag{8}$$

s.t.  $0e \leq \alpha \leq \frac{c_2}{m} e$ ,  
 $e^T \alpha \leq c_2 \nu_1$ ,

and

$$\min_{\beta} \frac{1}{2} \beta^T G(G^T G + c_3 I)^{-1} G^T \beta + Y^T G(G^T G + c_3 I)^{-1} G^T \beta - Y^T \beta$$

s.t.  $0e \leq \beta \leq \frac{c_4}{m} e$ ,  
 $e^T \beta \leq c_4 \nu_2$ ,

where  $G = [A e]$  and  $\alpha, \beta \in \mathfrak{R}^m$  are Lagrangian multiplies vectors.

After solving the dual QPPs (8) and (9), we can get solutions  $\alpha$  and  $\beta$ , and then the regression parameters can be obtained as  $[w_1^T, b_1]^T = (c_1 I + G^T G)^{-1} G^T (Y - \alpha)$ ,  $[w_2^T, b_2]^T = (c_3 I + G^T G)^{-1} G^T (Y + \beta)$ . Then, the regressor  $f(x)$  can be obtained by taking the mean of the two bound functions:  $f(x) = \frac{1}{2}(f_1(x) + f_2(x)) = \frac{1}{2}(w_1 + w_2)^T x + \frac{1}{2}(b_1 + b_2)$ .

## 3 Wavelet transform-based weighted $\nu$ -twin support vector regression

In this section, a novel wavelet transform-based weighted  $\nu$ -TSVR is presented. Similar to the  $\nu$ -TWSVR model, the WTWTSVR is constructed by two nonparallel hyperplanes, down-bound  $f_1(x)$ , and up-bound  $f_2(x)$ ; each hyperplane determines the  $\epsilon$ -insensitive bound regressor, and the end regressor is  $f(x) = \frac{1}{2}(f_1(x) + f_2(x))$ .

### 3.1 Linear wavelet transform-based weighted $\nu$ -TSVR

Linear WTWTSVR also solves a pair of optimization problems, which are described as follows:

$$\min_{w_1, b_1, \xi_1, \epsilon_1} \frac{1}{2} (Y - (Aw_1 + eb_1))^T D (Y - (Aw_1 + eb_1)) + \frac{1}{2} c_1 (w_1^T w_1 + b_1^2) + c_2 \left( \nu_1 \epsilon_1 + \frac{1}{m} d^T \xi_1 \right) \tag{10}$$

s.t.  $Y - (Aw_1 + eb_1) \geq -\epsilon_1 e - \xi_1$   $\xi_1 \geq 0e$   $\epsilon_1 \geq 0$ ,

$$\min_{w_2, b_2, \xi_2, \epsilon_2} \frac{1}{2} (Y - (Aw_2 + eb_2))^T D (Y - (Aw_2 + eb_2)) + \frac{1}{2} c_3 (w_2^T w_2 + b_2^2) + c_4 \left( \nu_2 \epsilon_2 + \frac{1}{m} d^T \xi_2 \right) \tag{11}$$

s.t.  $(Aw_2 + eb_2) - Y \geq -\epsilon_2 e - \xi_2$   $\xi_2 \geq 0e$   $\epsilon_2 \geq 0$ ,

where  $\nu_1, \nu_2, c_1, c_2, c_3, c_4 > 0$  are parameters chosen a priori by the user,  $\epsilon_1$  and  $\epsilon_2$  are insensitive parameters,  $\xi_1$  and  $\xi_2$  are slack vectors to measure the errors of samples outside the ‘ $\epsilon$  tube’, and  $m$  is the number of training points.  $d \in \mathfrak{R}^m$  is a weighting vector, and the diagonal matrix  $D = \text{diag}(d) \in \mathfrak{R}^{m \times m}$  is a weighting matrix, which will be discussed later. Note that when  $D = I$  and  $d = e$ , the proposed WTWTSVR will be degraded to  $\nu$ -TWSVR, and thus the WTWTSVR is the extension of  $\nu$ -TWSVR.

The first term in the objective function of (10) is the sum of weighted squared distances from training points to the

down-bound function. The second term is a regularization term, which can make  $f_1(x)$  as smooth as possible. The third term aims to make the  $\epsilon$  tube as narrow as possible and minimize the sum of errors of the points to lower than the down-bound  $f_1(x)$ , which brings the possibility of overfitting the training points. The ratio of three penalty terms in the objective function of (10) can be adjusted by the choice of  $c_1, c_2, v_1$ . For the optimization problem (11), we have similar illustrations.

To solve the optimization problems (10) and (11), their dual problems must be derived. Now the Lagrangian function for the problem (10) is introduced as follows:

$$\begin{aligned}
 L(w_1, b_1, \xi_1, \epsilon_1, \alpha, \beta, \gamma) &= \frac{1}{2}(Y - (Aw_1 + eb_1))^T D(Y - (Aw_1 + eb_1)) \\
 &+ \frac{1}{2}c_1(w_1^T w_1 + b_1^2) + c_2\left(v_1\epsilon_1 + \frac{1}{m}d^T \xi_1\right) \\
 &- \alpha^T(Y - (Aw_1 + eb_1) + \epsilon_1 e + \xi_1) \\
 &- \beta^T \xi_1 - \gamma\epsilon_1,
 \end{aligned} \tag{12}$$

where  $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_m)^T, \beta = (\beta_1, \beta_2, \dots, \beta_m)^T$ , and  $\gamma$  are nonnegative Lagrangian multipliers. The KTT optimality conditions are given by

$$\frac{\partial L}{\partial w_1} = -A^T D(Y - (Aw_1 + eb_1)) + c_1 w_1 + A^T \alpha = 0, \tag{13}$$

$$\frac{\partial L}{\partial b_1} = -e^T D(Y - (Aw_1 + eb_1)) + c_1 b_1 + e^T \alpha = 0, \tag{14}$$

$$\frac{\partial L}{\partial \xi_1} = \frac{c_2}{m}d - \alpha - \beta = 0, \tag{15}$$

$$\frac{\partial L}{\partial \epsilon_1} = c_2 v_1 - e^T \alpha - \gamma = 0, \tag{16}$$

$$Y - (Aw_1 + eb_1) \geq -\epsilon_1 e - \xi_1, \quad \xi_1 \geq 0, \quad \epsilon_1 \geq 0, \tag{17}$$

$$\alpha^T(Y - (Aw_1 + eb_1) + \epsilon_1 e + \xi_1) = 0, \tag{18}$$

$$\beta^T \xi_1 = 0, \quad \gamma\epsilon_1 = 0, \tag{19}$$

$$\alpha \geq 0e, \quad \beta \geq 0e, \quad \gamma \geq 0. \tag{20}$$

It follows from (15), (16), and (20) that the following is true:

$$\begin{aligned}
 0e \leq \alpha \leq \frac{c_2}{m}d, \\
 e^T \alpha \leq c_2 v_1.
 \end{aligned}$$

Defining  $G = [A \ e]$  and  $u_1 = [w_1^T \ b_1]^T$ , and combining (13) and (14) would result in the equation

$$-G^T D Y + (G^T D G + c_1 I)u_1 + G^T \alpha = 0,$$

from which the following can be determined:

$$u_1 = (G^T D G + c_1 I)^{-1} G^T (D Y - \alpha).$$

After substituting the value of  $u_1$  into (12), and using the above KTT optimality conditions, the dual problem of the proposed method can obtained as follows:

$$\begin{aligned}
 \max_{\alpha} L &= \max_{\alpha} \frac{1}{2}(Y - Gu_1)^T D(Y - Gu_1) \\
 &+ \frac{1}{2}c_1 u_1^T u_1 + c_2 v_1 \epsilon_1 + \frac{c_2}{m}d^T \xi_1 \\
 &- \alpha^T(Y - Gu_1 + \epsilon_1 e + \xi) - \beta^T \xi_1 - \gamma \epsilon_1 \\
 &= \max_{\alpha} \frac{1}{2}(Y^T D - u_1^T G^T D)(Y - Gu_1) \\
 &+ \frac{1}{2}c_1 u_1^T u_1 - \alpha^T Y + \alpha^T Gu_1 \\
 &= \max_{\alpha} \frac{1}{2}Y^T D Y - \frac{1}{2}u_1^T G^T D Y - \frac{1}{2}Y^T D Gu_1 \\
 &+ \frac{1}{2}u_1^T (G^T D G + c_1 I)u_1 + \alpha^T Gu_1 - \alpha^T Y \\
 &= \max_{\alpha} \frac{1}{2}Y^T D Y - \frac{1}{2}u_1^T G^T D Y - \frac{1}{2}Y^T D Gu_1 \\
 &+ \frac{1}{2}u_1^T G^T D Y - \frac{1}{2}u_1^T G^T \alpha - \alpha^T Y + \alpha^T Gu_1 \\
 &= \max_{\alpha} \left(-\frac{1}{2}Y^T D G + \frac{1}{2}\alpha^T G\right)u_1 - \alpha^T Y \\
 &= \max_{\alpha} \left(-\frac{1}{2}Y^T D G + \frac{1}{2}\alpha^T G\right) \\
 &\quad \times (G^T D G + c_1 I)^{-1} G^T (D Y - \alpha) - \alpha^T Y
 \end{aligned} \tag{21}$$

Note that  $D = D^T$  and the terms that are not related to  $\alpha$  [such as the term  $\frac{1}{2}Y^T D Y$  in (21)] are omitted; this does not affect the maximization process. The dual problem of (12) can be expressed as

$$\begin{aligned}
 \min \frac{1}{2}\alpha^T G(G^T D G + c_1 I)^{-1} G^T \alpha - Y^T \\
 DG(G^T D G + c_1 I)^{-1} G^T \alpha + Y^T \alpha \\
 s.t. 0e \leq \alpha \leq \frac{c_2}{m}d, \\
 e^T \alpha \leq c_2 v_1.
 \end{aligned} \tag{22}$$

Similarly, the dual problem of (11) is deduced as

$$\begin{aligned}
 \min \frac{1}{2}\lambda^T G(G^T D G + c_3 I)^{-1} G^T \lambda + Y^T \\
 DG(G^T D G + c_3 I)^{-1} G^T \lambda - Y^T \lambda \\
 s.t. 0e \leq \lambda \leq \frac{c_4}{m}d, \\
 e^T \lambda \leq c_4 v_2,
 \end{aligned} \tag{23}$$

and

$$u_2 = [w_2^T \ b_2]^T = (G^T DG + c_3 I)^{-1} G^T (DY + \lambda).$$

Thus, the down-bound  $f_1(x) = w_1^T x + b_1$  and the up-bound  $f_2(x) = w_2^T x + b_2$  are obtained to calculate the regressor  $f(x) = \frac{1}{2}(f_1(x) + f_2(x))$ .

### 3.2 Nonlinear wavelet transform-based weighted v-TSVR

The SVR can be used to solve linear regression problems. A line/plane/hyperplane can be adopted to regress the relationship between inputs and outputs by training samples. Unfortunately, the line/plane/hyperplane is only suitable for linear problems. The adoption of kernel mapping can extend the algorithm to nonlinear cases, which are the majority of cases in the real world based on Vapnik’s theory [7]. The kernel trick is adopted to map the input into a higher-dimensional feature space. The following kernel-generated functions are considered: down-bound  $f_1(x) = K(x^T, A^T)w_1 + b_1$  and up-bound  $f_2(x) = K(x^T, A^T)w_2 + b_2$ , where  $K$  is an appropriately chosen kernel. Therefore, the end regressor is  $f(x) = \frac{1}{2}(f_1(x) + f_2(x))$ . The corresponding optimization problems can be described as follows:

$$\begin{aligned} \min_{w_1, b_1, \xi_1, \epsilon_1} & \frac{1}{2}(Y - (K(A, A^T)w_1 + eb_1))^T D(Y - (K(A, A^T)w_1 + eb_1)) \\ & + \frac{1}{2}c_1(w_1^T w_1 + b_1^2) + c_2(v_1 \epsilon_1 + \frac{1}{m}d^T \xi_1) \\ \text{s.t.} & Y - (K(A, A^T)w_1 + eb_1) \geq -\epsilon_1 e - \xi_1 \quad \xi_1 \geq 0 \quad \epsilon_1 \geq 0, \end{aligned} \tag{24}$$

$$\begin{aligned} \min_{w_2, b_2, \xi_2, \epsilon_2} & \frac{1}{2}(Y - (K(A, A^T)w_2 + eb_2))^T D(Y - (K(A, A^T)w_2 + eb_2)) \\ & + \frac{1}{2}c_3(w_2^T w_2 + b_2^2) + c_4(v_2 \epsilon_2 + \frac{1}{m}d^T \xi_2) \\ \text{s.t.} & (K(A, A^T)w_2 + eb_2) - Y \geq -\epsilon_2 e - \xi_2 \quad \xi_2 \geq 0 \quad \epsilon_2 \geq 0. \end{aligned} \tag{25}$$

We define the Lagrangian function for the QPP problem (24) as

$$\begin{aligned} L(w_1, b_1, \xi_1, \epsilon_1, \alpha, \beta, \gamma) & = \frac{1}{2}(Y - (K(A, A^T)w_1 + eb_1))^T \\ & \times D(Y - (K(A, A^T)w_1 + eb_1)) \\ & + \frac{1}{2}c_1(w_1^T w_1 + b_1^2) + c_2\left(v_1 \epsilon_1 + \frac{1}{m}d^T \xi_1\right) \\ & - \alpha^T(Y - (K(A, A^T)w_1 + eb_1) + \epsilon_1 e + \xi_1) \\ & - \beta^T \xi_1 - \gamma \epsilon_1, \end{aligned} \tag{26}$$

where  $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_m)^T$ ,  $\beta = (\beta_1, \beta_2, \dots, \beta_m)^T$ , and  $\gamma$  are nonnegative Lagrangian multipliers. The KTT optimality conditions are given by

$$\begin{aligned} \frac{\partial L}{\partial w_1} & = -K(A, A^T)^T D(Y - (K(A, A^T)w_1 + eb_1)) \\ & + c_1 w_1 + K(A, A^T)^T \alpha = 0, \end{aligned} \tag{27}$$

$$\begin{aligned} \frac{\partial L}{\partial b_1} & = -e^T D(Y - (K(A, A^T)w_1 + eb_1)) \\ & + c_1 b_1 + e^T \alpha = 0, \end{aligned} \tag{28}$$

$$\frac{\partial L}{\partial \xi_1} = \frac{c_2}{m}d - \alpha - \beta = 0, \tag{29}$$

$$\frac{\partial L}{\partial \epsilon_1} = c_2 v_1 - e^T \alpha - \gamma = 0, \tag{30}$$

$$Y - (K(A, A^T)w_1 + eb_1) \geq -\epsilon_1 e - \xi_1, \quad \xi_1 \geq 0, \quad \epsilon_1 \geq 0, \tag{31}$$

$$\alpha^T(Y - (K(A, A^T)w_1 + eb_1) + \epsilon_1 e + \xi_1) = 0, \tag{32}$$

$$\beta^T \xi_1 = 0, \quad \gamma \epsilon_1 = 0, \tag{33}$$

$$\alpha \geq 0e, \quad \beta \geq 0e, \quad \gamma \geq 0. \tag{34}$$

Defining  $H = [K(A, A^T) \ e]$  and  $u_1 = [w_1^T \ b_1]^T$ , and then combining (27) and (28) would result in the following equation:

$$-H^T DY + (H^T DH + c_1 I)u_1 + H^T \alpha = 0.$$

Thus, we have

$$u_1 = (H^T DH + c_1 I)^{-1} G^T (DY - \alpha). \tag{35}$$

By replacing  $w_1$  and  $b_1$  in (26) by (35), and applying the above KTT optimality conditions, the dual problem can be obtained as follows:

$$\begin{aligned} \min & \frac{1}{2} \alpha^T H (H^T DH + c_1 I)^{-1} H^T \alpha \\ & - Y^T DH (H^T DH + c_1 I)^{-1} H^T \alpha + Y^T \alpha \\ \text{s.t.} & 0e \leq \alpha \leq \frac{c_2}{m}d, \\ & e^T \alpha \leq c_2 v_1. \end{aligned} \tag{36}$$

Similarly, the dual problem of (25) can be expressed by

$$\begin{aligned} \min & \frac{1}{2} \lambda^T H (H^T DH + c_3 I)^{-1} H^T \lambda \\ & + Y^T DH (H^T DH + c_3 I)^{-1} G^T \lambda - Y^T \lambda \\ \text{s.t.} & 0e \leq \lambda \leq \frac{c_4}{m}d, \\ & e^T \lambda \leq c_4 v_2, \end{aligned} \tag{37}$$

and we can have

$$u_2 = [w_2^T \ b_2]^T = (H^T DH + c_3 I)^{-1} G^T (DY + \lambda). \tag{38}$$

### 3.3 Determination of weighting parameters by wavelet transform

The parameters mentioned in the first subsection,  $d \in \mathfrak{R}^m$  and  $D \in \mathfrak{R}^{m \times m} (D = \text{diag}(d))$ , are the weighting vector and matrix, respectively. They should be determined beforehand according to the importance of training data. By direct observation of the objective functions of TSVR and  $v$ -TWSVR, it is easy to see that all of the samples have the same penalties, which may reduce the performance of the regressors due to the impact of data with too much noise. Instinctively, different training samples should be given different weights; a larger given weight means that the sample is more important. Motivated by this idea and noting that the Gaussian function can reflect this trend, the weighting parameter  $d (= [d_1, d_2, \dots, d_m]^T)$  is determined by the Gaussian function described as follows:

$$d = E \exp \left( -|Y - \hat{Y}|^2 / \sigma^2 \right), \tag{39}$$

where  $E$  is the amplitude,  $\sigma$  represents the standard deviation, and the selection of  $E$  and  $\sigma$  is determined by the statistical characteristics of training data and is usually adjusted by experiments. The inappropriate selection of  $E$  and  $\sigma$  makes (39) unable to reflect the statistical characteristics of training data and reduces the regression performance.  $\hat{Y} (= [\hat{y}_1, \hat{y}_2, \dots, \hat{y}_m]^T)$  is the estimation value vector of output  $Y (= [y_1, y_m, \dots, y_m]^T)$ . Any time series filter can be used to calculate  $\hat{Y}$ , but the wavelet filter is applicable to short-term signal processing, which is characteristic of most actual time series signals. Wavelet transform is a transform analysis method. It inherits and develops the idea of short-time Fourier transform localization, and overcomes the shortcomings of window size not changing with frequency. It can provide a time-frequency window that changes with frequency. It is a relatively ideal tool for time-frequency analysis and processing signals. Therefore, wavelet filtering is adopted to calculate  $\hat{Y}$  by three stages, as described below.

1. Wavelet transform may be considered a form of time-frequency representation for signals, and thus are related to harmonic analysis. Discrete wavelet transforms (DWT) use discrete-time filter banks. The DWT of a signal in the  $l$ -th decomposition step  $x_l^a(n)$  is calculated by a series of filters. The samples are passed through a low-pass filter with impulse response  $\phi(t)$  resulting in the approximation coefficients  $x_{l+1}^a$ , and through a high-pass filter with impulse response  $\psi(t)$  resulting in the detail coefficients  $x_{l+1}^d$ .

$$x_{l+1}^a(n) = \sum_k \phi(k - 2n)x_l^a(k) \tag{40}$$

$$x_{l+1}^d(n) = \sum_k \psi(k - 2n)x_l^a(k) \tag{41}$$

The approximation coefficients  $x_{l+1}^a$  can be decomposed further to get  $x_{l+2}^a$  and  $x_{l+2}^d$ . This process of decomposition is represented as a binary tree, as illustrated in Fig. 1a.

2. The obtained  $l$  groups of decomposed sequence  $(x_1^d, x_2^d, \dots, x_l^d, x_l^a)$  after  $l$  steps of decomposition are processed by an appropriate algorithm to remove noise. In this paper, the high frequency part of the frequency domain signal is directly zeroed as denoising algorithm. Then the denoised sequence  $(x_1^{d'}, x_2^{d'}, \dots, x_l^{d'}, x_l^{a'})$  is obtained.
3. In this stage, the estimation value of output  $\hat{y}$  is reconstructed by the denoised sequence  $(x_1^{d'}, x_2^{d'}, \dots, x_l^{d'}, x_l^{a'})$ .

$$x_{l-1}^{a'}(n) = \sum_k \phi(n - 2k)x_l^{a'}(k) + \sum_k \psi(n - 2k)x_l^{d'}(k) \tag{42}$$

This process of reconstruction is carried on further, and after  $l$  steps of reconstruction the estimation value of output  $\hat{Y}$  can be obtained, i.e.,  $\hat{Y} = x_0^{a'}$ . We next substitute  $\hat{Y}$  into (39), and then the weighting vector  $d$  and matrix  $D$  can be calculated. The process of signal decomposition and reconstruction is illustrated in Fig. 1.

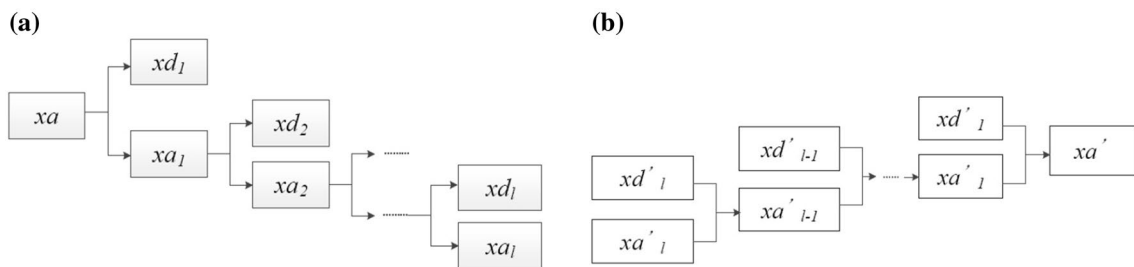


Fig. 1 Wavelet transform-based signal processing. **a** Decomposition process, and **b** reconstruction process



The wavelet decomposition layers  $l$  can affect the performance of denoising. Usually, too many decomposition layers will result in serious loss of signal information. After denoising, the signal-to-noise ratio may decrease and the computational complexity will increase. If the number of decomposition layers is too small, the denoising effect is not ideal and the signal-to-noise ratio is not improved much. The general decomposition layers are determined by empirical methods.

### 3.4 Algorithm summary and discussion

In this subsection, we illustrate the proposed WTWTSVR algorithm.

**Algorithm 1:** Wavelet transform-based weighted  $\nu$ -twin support vector regression.

Input: Training data input matrix  $A$ ; training data output vector  $Y$ ; the appropriate parameters  $c_1, c_2, c_3, c_4, \nu_1$  and  $\nu_2$ , in (24) and (25); Gaussian function parameters  $E, \sigma^2$  in (39).

Output: The regression function  $f(x)$ .

Process:

1. Preprocess  $Y$  by the wavelet transform method described in subsection (3.3) and get  $\hat{Y}$ . Calculate  $d$  by (39).
2. By calculating the QPP problems in Eqs. (36) and (37), we can get  $\alpha$  and  $\lambda$ , respectively.
3. Calculate  $u_1$  and  $u_2$  by (35) and (38), respectively.
4. Compute  $f(x) = \frac{1}{2}(w_1 + w_2)^T x + \frac{1}{2}(b_1 + b_2)$ .

The computation complexity of the proposed algorithm is mainly determined by the computations of a pair of QPPs and a pair of inverse matrices. If the number of training samples is  $l$ , then the training complexity of dual QPPs of the proposed algorithm is about  $O(2l^3)$  while the training complexity of the traditional SVR is about  $O(8l^3)$ , which implies that the training speed of SVR is about four times slower than that of the proposed algorithm. Furthermore, a pair of inverse matrices with the size  $(l + 1)(l + 1)$  in QPPs have the same computational cost  $O(l^3)$ . During the training process, it is good to cache the inverse matrices with some memory cost in order to avoid repeated computations. In addition, the proposed algorithm contains the wavelet transform weighted matrix, and a Db-3 wavelet with a length of 6 is used in this paper; then, the complexity of the wavelet transform is less than  $12l$ . By comparing with the computations of QPPs and the inverse matrix, the complexity of computing the wavelet matrix can be ignored. Therefore, the computation complexity of the proposed algorithm is about  $O(3l^3)$ .

We mention that weights are inserted into both quadratic and first-degree terms in the proposed method, which is significantly different from previous works [22, 25]. The weights adopted in [22, 25] are only inserted into the quadratic term. Intuitively, the WTWTSVR can make full use

of the prior information of training samples. The weight matrix and weight vector, which represent the distance of noised samples and its ‘real position,’ can reflect the prior information of the training samples. The larger weight is given to samples with smaller noise and the smaller weight is given to samples with larger noise. The adding of the weight matrix and weight vector to the objective function can reduce the impact of noise and outliers.

The weight calculation method given in [22] is also used to make use of the prior information of training data. One of the differences between the WTWTSVR and the method proposed in [22] lies in the calculation method of weights. In [22], the K-nearest neighbor (KNN) algorithm is adopted. In the algorithm, the KNN is adopted to calculate the penalty weights of samples. The idea of the KNN is that a sample point  $x$  is important if it has a larger number of K-nearest neighbors, whereas it is not important if it is an outlier that has a small number of K-nearest neighbors. We can see that the KNN algorithm is suitable for dealing with clustered samples but not time series. The wavelet transform theory adopted in our proposed algorithm is a new angle for preprocessing samples. Wavelet transform is a kind of time-frequency representation for signals, and the proposed method based on the wavelet theory is suitable for dealing with time series samples.

## 4 Experimental results

In this section, some experiments are made to examine the performance of our proposed WTWTSVR. WTWTSVR is compared with TSVR [11],  $\nu$ -TWSVR [21], Asy- $\nu$ -TSVR [19], KNNUPWTSVR [26], and WL- $\epsilon$ -TSVR [25] using three artificial datasets and eight benchmark datasets. The computer simulations are implemented in a Matlab R2014a environment on a PC with an Intel Core i5 processor (3.3 GHz) with 8 GB RAM. In this paper, the Gaussian nonlinear kernel is adopted for all of the datasets, that is

$$K(a^T, b^T) = \exp\left(-\frac{\|a - b\|^2}{e}\right), \quad (43)$$

where  $e$  determines the width of the Gaussian function. The choice of parameters is essential for the performance of algorithms. In this paper, parameter values are chosen by the grid search method from the set of values  $\{10^i | i = -4, -3, \dots, 5\}$ . In order to degrade the computational complexity of parameter selection, let  $C_1 = C_2 = C$  and  $\epsilon_1 = \epsilon_2 = \epsilon$  in TSVR;  $C_1 = C_2 = C$  and  $\nu_1 = \nu_2 = \nu$  in Asy- $\nu$ -TSVR; and  $c_1 = c_3, c_2 = c_4$ , and  $\nu_1 = \nu_2 = \nu$  in  $\nu$ -TWSVR and the proposed WTWTSVR.

The performance of our proposed WTWTSVR and the three other methods are evaluated by selected criteria. The number of testing samples is denoted by  $l$ ,  $y_i$  denotes the real value of a testing sample point  $x_i$ ,  $\bar{y} = \sum_{i=1}^l \frac{1}{l} y_i$  is the mean value of  $y_1, y_2, \dots, y_l$ , and  $\hat{y}_i$  denotes the predicted value of  $x_i$ . The criteria are specified as follows.

- SSE: Sum squared error of testing samples, defined as  $SSE = \sum_{i=1}^l (y_i - \hat{y}_i)^2$ .
- SST: Sum squared deviation of testing samples, defined as  $SST = \sum_{i=1}^l (y_i - \bar{y})^2$ .
- SSR: Sum squared deviation that can be explained by the estimator, defined as  $SSR = \sum_{i=1}^l (\hat{y}_i - \bar{y})^2$ .

SSE represents the fitting precision, but too small of an SSE value may mean overfitting of the regressor due to the fitting of noise. SST represents the variance of the testing samples, and SSR reflects the explanation ability of the regressor.

- SSE/SST: Ratio between sum squared error and sum squared deviation of testing samples, defined as  $SSE/SST = \sum_{i=1}^l (y_i - \hat{y}_i)^2 / \sum_{i=1}^l (y_i - \bar{y})^2$ .
- SSR/SST: Ratio between interpretable sum squared deviation and real sum squared deviation of testing samples, defined as  $SSR/SST = \sum_{i=1}^l (\hat{y}_i - \bar{y})^2 / \sum_{i=1}^l (y_i - \bar{y})^2$ .

In most cases, a small SSE/SST represents good agreement between estimations and real values; however, too small of an SSE/SST also probably means overfitting of the regressor. SSR/SST is the variance ratio of estimated data over sample data; if  $SSR/SST = 1$ , the estimated data share the same variance as training samples.

#### 4.1 Experiments on artificial datasets

To demonstrate the performance of the proposed algorithm on artificial datasets, consider three noised functions for approximation denoted as  $y = f(x) + n$ , where the noise  $n \sim N(0, \zeta^2)$  is Gaussian additive noise with mean zero, and variance  $\zeta^2$  is 0.1<sup>2</sup> and 0.2<sup>2</sup>, respectively. Set the number of training points to be 300. The Db-3 wavelet used in WTWTSVR is selected to denoise the training data and the number of decomposition layers is 5. Five algorithms (TSVR [11],  $\nu$ -TWSVR [21], Asy- $\nu$ -TSVR [19], KNNUPWTSVR [26], and WL- $\epsilon$ -TSVR [25]) are adopted for performance comparisons with the proposed WTWTSVR.

Figure 2 shows the one-run approximation functions obtained by the six algorithms.

The selection of parameters can be seen in Table 1. The adopted functions for regression are defined as follows.

$$\text{Function 1: } f(x) = \frac{\sin x}{x}$$

$$\text{Function 2: } f(x) = x^{2/3}$$

$$\text{Function 3: } f(x) = \frac{4}{|x|+2} + \cos(2x) + \sin(3x)$$

To avoid biased comparisons, for each case, ten independent groups of noised samples are generated randomly, and the number of training points and testing points are all set to 300. Testing data is selected randomly under a uniform distribution and is assumed to be noise-free. Other settings can be seen in Table 1.

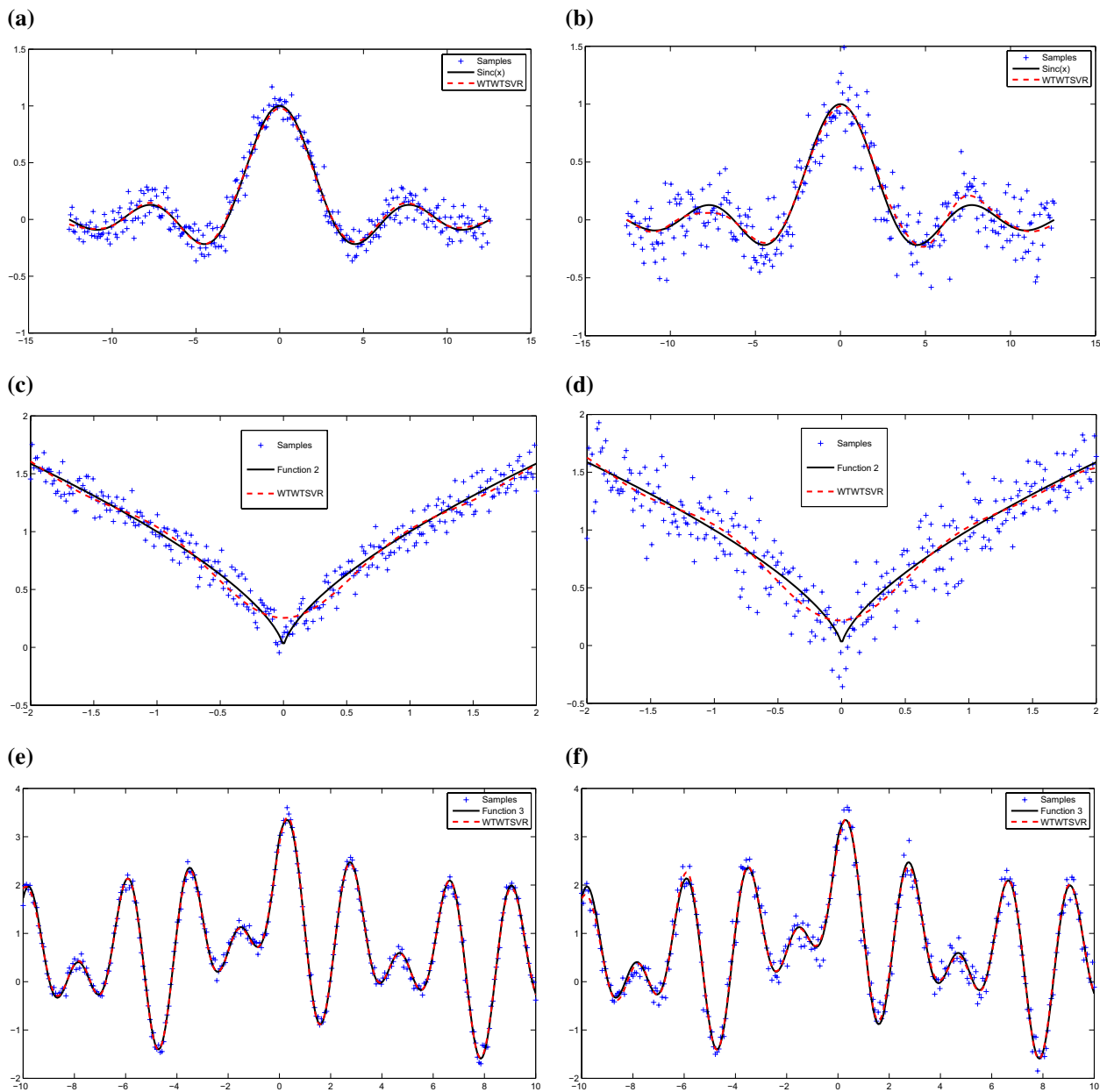
The average results are summarized in Table 2. In the numerical value columns, the first column shows the mean value of ten testing results, and the second column represents the plus or minus the standard deviation. Obviously, the proposed WTWTSVR obtains the lowest testing errors compared with TSVR, Asy- $\nu$ -TSVR,  $\nu$ -TWSVR, KNNUPWTSVR, and WL- $\epsilon$ -TSVR, and it gets the smallest SSE and SSE/SST values. Although the SSR/SST value of WTWTSVR is not the largest, it is close to that obtained by other algorithms. From the comparisons of the six algorithms, we can see that the proposed WTWTSVR has better performance in most criteria.

Figure 3 shows the impact of variance  $\sigma^2$  in (39) on the performance index SSE. The  $x$  coordinate is  $\sigma^2$ , and the  $y$  coordinate is the SSE value. The function for approximation is Function 1, and the Gaussian additive noise is with mean zero, and variance  $\zeta^2 = 0.1^2$  or  $0.2^2$ . The result is averaged by 500 independent tests. We can see that the variance  $\sigma^2$  of the Gaussian function for weighting has a great influence on the regression performance of WTWTSVR. The curve is convex, and for fixed noise, optimal  $\sigma^2$  can be obtained. The parameter  $\sigma^2$  determines the width of the Gaussian function. A large  $\sigma^2$  leads to a fat Gaussian curve and similar weights for samples, while a small  $\sigma^2$  leads to a narrow Gaussian curve and largely different weights for samples. Therefore, the value of  $\sigma^2$  determines the impact of different samples, and the curve in Fig. 3 decreases sharply and increases slowly.

The selection of the kernel function is important for the performance of regression algorithms. In order to test the effect of kernel functions, we study the performance of six kernel functions for artificial datasets. Function 1 includes the Gaussian kernel adopted in the proposed method. The function expressions and performance are listed in Table 3. It is easy to see that the Gaussian kernel achieves the best results.

To test the effect of weighting parameter  $D$ , experiments on the SSE value of regression of artificial functions are conducted. Table 4 shows the results of tests for noise variances  $\zeta^2$  of 0.1<sup>2</sup> and 0.2<sup>2</sup>. The first row in the table represents the performance of the proposed WTWTSVR (with  $D$ ), and the second row shows that of the same algorithm





**Fig. 2** Performance of WTWSVR on **a** Function 1 with noise  $\zeta = 0.1$ , **b** Function 1 with noise  $\zeta = 0.2$ , **c** Function 2 with noise  $\zeta = 0.1$ , **d** Function 2 with noise  $\zeta = 0.2$ , **e** Function 3 with noise  $\zeta = 0.1$ , and **f** Function 3 with noise  $\zeta = 0.2$

except when setting  $D$  as an identity matrix ( $D = \mathbf{1}$ ). The selected parameters can be seen in Table 1. It is seen that the SSE values of WTWSVR are low when introducing the weighting parameter  $D$  calculated by the wavelet transform method; this verifies the effectiveness of adding the weighting parameter  $D$ .

To test the sensitivity of parameter selection on WTWSVR regression performance, we study the influence of parameters  $c_1, c_2, \nu, E,$  and  $\sigma^2$  on SSE for artificial datasets with Function 1 with noise variance  $\sigma^2 = 0.1^2$ . From

Figure 4 we can see that the SSE curves show convexity, i.e., with the increase in a parameter, performance value first decrease and then increase, which ensures that the optimal value can be found. However, the sensitivities to the change in parameter values are different. The performance is sensitive to  $c_1, E,$  and  $\sigma^2$ , but not to  $c_2$  and  $\nu$ .

To test the robustness of the proposed algorithm to outliers, we verified the performance of the proposed method on Function 1 with Gaussian additive noise, and with mean zero and variance  $\zeta^2 = 0.1^2$ . Figure 5 shows that the prediction

**Table 1** Parameter selection for the six algorithms on artificial datasets with different variance of Gaussian noises

Datasets	Regressors	Parameters	Parameter values
Function 1	WTWTSVR	$c_1, c_2, \nu, E, \sigma^2$	1,0.1,0.1,0.1,0.01
$N(0, 0.1^2)$	TSVR	$C, \epsilon$	0.1, 0.1
	$\nu$ -TWSVR	$c_1, c_2, \nu$	1, 0.1, 0.1
	Asy- $\nu$ -TSVR	$C, \nu, p$	0.1, 0.1, 0.5
	KNNUPWTSVR	$c_1/c_2, c_3/c_4$	0.1, 0.1
	WL- $\epsilon$ -TSVR	$c_1/c_2, \nu_1/\nu_2$	0.1, 1
Function 1	WTWTSVR	$c_1, c_2, \nu, E, \sigma^2$	1, 0.1, 0.1, 0.1, 0.1
$N(0, 0.2^2)$	TSVR	$C, \epsilon$	0.1, 0.1
	$\nu$ -TWSVR	$c_1, c_2, \nu$	1, 0.1, 0.1
	Asy- $\nu$ -TSVR	$C, \nu, p$	0.1, 0.1, 0.5
	KNNUPWTSVR	$c_1/c_2, c_3/c_4$	0.1, 0.1
	WL- $\epsilon$ -TSVR	$c_1/c_2, \nu_1/\nu_2$	0.1, 1
Function 2	WTWTSVR	$c_1, c_2, \nu, E, \sigma^2$	0.001, 0.1, 0.1, 10, 0.01
$N(0, 0.1^2)$	TSVR	$C, \epsilon$	1, 0.01
	$\nu$ -TWSVR	$c_1, c_2, \nu$	0.001, 0.1, 0.1
	Asy- $\nu$ -TSVR	$C, \nu, p$	0.1, 0.1, 0.5
	KNNUPWTSVR	$c_1/c_2, c_3/c_4$	0.1, 0.01
	WL- $\epsilon$ -TSVR	$c_1/c_2, \nu_1/\nu_2$	0.1, 0.001
Function 2	WTWTSVR	$c_1, c_2, \nu, E, \sigma^2$	0.001, 0.1, 0.1, 10, 0.1
$N(0, 0.2^2)$	TSVR	$C, \epsilon$	0.1, 0.1
	$\nu$ -TWSVR	$c_1, c_2, \nu$	0.001, 0.1, 0.1
	Asy- $\nu$ -TSVR	$C, \nu, p$	0.1, 0.1, 0.5
	KNNUPWTSVR	$c_1/c_2, c_3/c_4$	0.1, 0.01
	WL- $\epsilon$ -TSVR	$c_1/c_2, \nu_1/\nu_2$	0.1, 0.001
Function 3	WTWTSVR	$c_1, c_2, \nu, E, \sigma^2$	0.001, 0.1, 0.1, 10, 0.01
$N(0, 0.1^2)$	TSVR	$C, \epsilon$	1, 0.01
	$\nu$ -TWSVR	$c_1, c_2, \nu$	0.001, 0.1, 0.1
	Asy- $\nu$ -TSVR	$C, \nu, p$	0.1, 0.1, 0.5
	KNNUPWTSVR	$c_1/c_2, c_3/c_4$	10, 0.01
	WL- $\epsilon$ -TSVR	$c_1/c_2, \nu_1/\nu_2$	10, 0.001
Function 3	WTWTSVR	$c_1, c_2, \nu, E, \sigma^2$	0.001, 0.1, 0.1, 10, 0.1
$N(0, 0.2^2)$	TSVR	$C, \epsilon$	0.1, 0.1
	$\nu$ -TWSVR	$c_1, c_2, \nu$	0.001, 0.1, 0.1
	Asy- $\nu$ -TSVR	$C, \nu, p$	0.1, 0.1, 0.5
	KNNUPWTSVR	$c_1/c_2, c_3/c_4$	10, 0.01
	WL- $\epsilon$ -TSVR	$c_1/c_2, \nu_1/\nu_2$	10, 0.001

performance of the proposed method is satisfying (see the blue line); SSE, SSE/SST, and SSR/SST values were 0.1678, 0.0058, and 0.9601, respectively. By comparing with the results in Table 2, it can be concluded that the performance of the proposed method with outliers is still better than that of the competitive algorithms with no outliers.

### 4.2 Experiments on benchmark datasets

To further evaluate the performance of the proposed algorithm, experiments on benchmark datasets from the UCI machine learning repository [27] are conducted. The selected datasets are all time series. For all of the real-word examples considered in this paper, the original data is normalized as  $\tilde{s}_i = (s_i - s_{\min}) / (s_{\max} - s_{\min})$ , where  $s_i$  is the input value of the  $i$ th sample,  $\tilde{s}_i$  is its corresponding normalized value, and  $s_{\min}$  and  $s_{\max}$  denote the minimum and maximum values, respectively. The selection of parameters can be seen in Table 5. Table 6 show the performance of the six algorithms for the following benchmark datasets.

*Energy*: Energy use prediction of appliances. This dataset studies the energy use of appliances in a low-energy building based on 29 attributes, such as temperature and humidity conditions, averaged for 10-min periods. The number of instances is 19,735. We choose 1000 instances, including 500 training samples and 500 test samples with 22 attributes (dimensions of input) for regression and testing, respectively.

*PM2.5*: Beijing PM2.5 data. This dataset aims to predict the hourly PM2.5 concentration data at the US embassy in Beijing. 43,824 instances and 13 attributes, such as dew point, temperature, and wind speed, are included; and 1000 instances and four attributes are adopted in our experiments.

*AirQuality*: Air quality. The dataset contains hourly averaged responses from an array of five metal oxide chemical sensors. 822 instances and nine attributes are adopted.

*DowJones*: Dow Jones index. This dataset predicts the Dow Jones index value; 720 instances and four attributes are adopted in our experiments.

*Istanbul1/2/4/8* is dataset of the Istanbul stock exchange; 536 samples with one, two, four, and eight attributes (input dimensions) are adopted independently.

Odd-numbered samples are selected as training data, while even-numbered samples are selected as test data; thus the number of the training data and test data are the same. The number ( $m$ ) and dimension ( $n$ ) of samples are listed in Table 6. In order to not destroy the time series characteristic of the data, one-shot experiments but not cross-validation experiments are conducted. From Table 6 we can see that our proposed WTWTSVR outperforms the other five algorithms. On most datasets, the WTWTSVR obtains the smallest SSE and SSE/SST values. Only on *Istanbul1* does our WTWTSVR have the second smallest value. For SSR/SST criteria, WTWTSVR achieves the second or the third values close to 1 on *Energy* to *DowJones*, and achieves the closest value to 1 on *Istanbul1* to *Istanbul8*. We can also see from Table 6 that the SSE value decreases with increasing input dimension, i.e., more input information leads to more accurate estimation. Based on the above analysis, it can be

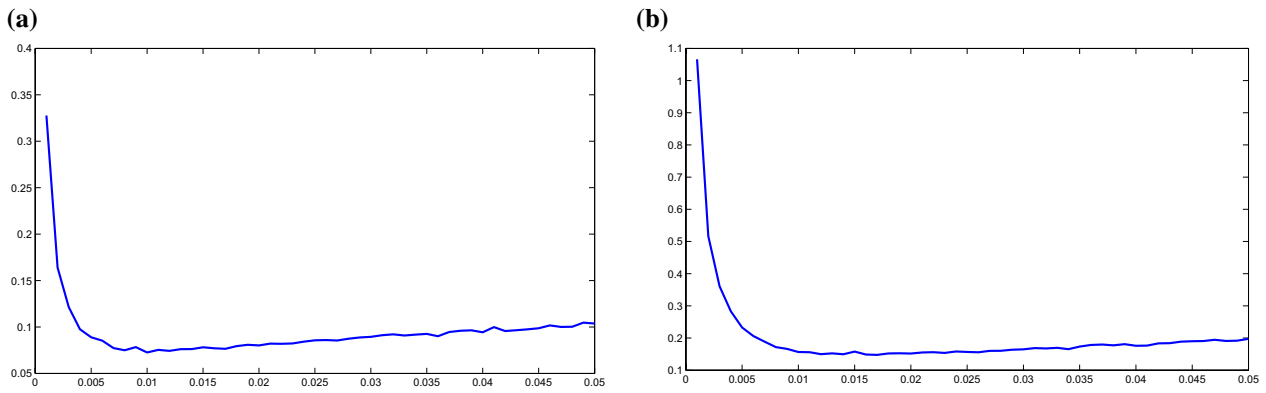
**Table 2** Performance comparison of algorithms on artificial datasets with different variance of Gaussian noises

Datasets	Regressors	SSE	SSE/SST	SSR/SST
Function 1 $N(0, 0.1^2)$	WTWTSVR	$0.1642 \pm 0.0472$	$0.0056 \pm 0.0016$	$0.9616 \pm 0.0339$
	TSVR	$0.2345 \pm 0.0803$	$0.0080 \pm 0.0028$	$1.0100 \pm 0.0358$
	$\nu$ -TWSVR	$0.1727 \pm 0.0628$	$0.0059 \pm 0.0022$	$1.0074 \pm 0.0357$
	Asy- $\nu$ -TSVR	$0.2247 \pm 0.0755$	$0.0077 \pm 0.0026$	$1.0095 \pm 0.0363$
	KNNUPWTSVR	$0.1850 \pm 0.0705$	$0.0063 \pm 0.0024$	$1.0079 \pm 0.0353$
	WL- $\epsilon$ -TSVR	$0.1727 \pm 0.0628$	$0.0059 \pm 0.0022$	$1.0074 \pm 0.0357$
Function 1 $N(0, 0.2^2)$	WTWTSVR	$0.6246 \pm 0.2103$	$0.0217 \pm 0.0073$	$0.9801 \pm 0.0443$
	TSVR	$1.0411 \pm 0.4150$	$0.0362 \pm 0.0144$	$1.0297 \pm 0.0396$
	$\nu$ -TWSVR	$0.7481 \pm 0.2856$	$0.0260 \pm 0.0099$	$1.0163 \pm 0.0352$
	Asy- $\nu$ -TSVR	$1.0071 \pm 0.4116$	$0.0350 \pm 0.0143$	$1.0288 \pm 0.0381$
	KNNUPWTSVR	$0.8716 \pm 0.3149$	$0.0303 \pm 0.0109$	$1.0345 \pm 0.0373$
	WL- $\epsilon$ -TSVR	$0.7482 \pm 0.2857$	$0.0260 \pm 0.0099$	$1.0163 \pm 0.0352$
Function 2 $N(0, 0.1^2)$	WTWTSVR	$0.6162 \pm 0.0449$	$0.0123 \pm 0.0009$	$0.9813 \pm 0.0290$
	TSVR	$0.8159 \pm 0.0437$	$0.0162 \pm 0.0009$	$0.9828 \pm 0.0406$
	$\nu$ -TWSVR	$0.7391 \pm 0.0340$	$0.0147 \pm 0.0007$	$0.9816 \pm 0.0296$
	Asy- $\nu$ -TSVR	$0.8657 \pm 0.0402$	$0.0172 \pm 0.0008$	$0.9763 \pm 0.0302$
	KNNUPWTSVR	$0.7684 \pm 0.0367$	$0.0153 \pm 0.0007$	$0.9782 \pm 0.0288$
	WL- $\epsilon$ -TSVR	$0.7280 \pm 0.0351$	$0.0145 \pm 0.0007$	$0.9828 \pm 0.0286$
Function 2 $N(0, 0.2^2)$	WTWTSVR	$0.8768 \pm 0.1467$	$0.0177 \pm 0.0030$	$1.0060 \pm 0.0575$
	TSVR	$1.1546 \pm 0.1349$	$0.0233 \pm 0.0027$	$0.9867 \pm 0.0628$
	$\nu$ -TWSVR	$1.0295 \pm 0.1273$	$0.0208 \pm 0.0026$	$0.9876 \pm 0.0610$
	Asy- $\nu$ -TSVR	$1.1790 \pm 0.1365$	$0.0238 \pm 0.0028$	$0.9864 \pm 0.0615$
	KNNUPWTSVR	$1.0555 \pm 0.1446$	$0.0213 \pm 0.0029$	$1.0005 \pm 0.0572$
	WL- $\epsilon$ -TSVR	$1.0219 \pm 0.1353$	$0.0206 \pm 0.0027$	$0.9904 \pm 0.0583$
Function 3 $N(0, 0.1^2)$	WTWTSVR	$0.4689 \pm 0.1060$	$0.0014 \pm 0.0003$	$1.0011 \pm 0.0126$
	TSVR	$1.0964 \pm 0.1080$	$0.0034 \pm 0.0003$	$0.9776 \pm 0.0145$
	$\nu$ -TWSVR	$0.5674 \pm 0.0689$	$0.0018 \pm 0.0002$	$0.9942 \pm 0.0127$
	Asy- $\nu$ -TSVR	$2.0892 \pm 0.1227$	$0.0065 \pm 0.0004$	$0.9464 \pm 0.0118$
	KNNUPWTSVR	$0.5352 \pm 0.0819$	$0.0017 \pm 0.0003$	$1.0022 \pm 0.0153$
	WL- $\epsilon$ -TSVR	$0.4915 \pm 0.0843$	$0.0015 \pm 0.0003$	$1.0043 \pm 0.0171$
Function 3 $N(0, 0.2^2)$	WTWTSVR	$1.0662 \pm 0.3388$	$0.0030 \pm 0.0009$	$1.0036 \pm 0.0184$
	TSVR	$2.2686 \pm 0.3427$	$0.0063 \pm 0.0010$	$0.9612 \pm 0.0190$
	$\nu$ -TWSVR	$1.2184 \pm 0.2494$	$0.0034 \pm 0.0007$	$0.9965 \pm 0.0191$
	Asy- $\nu$ -TSVR	$2.5237 \pm 0.3242$	$0.0070 \pm 0.0009$	$0.9544 \pm 0.0185$
	KNNUPWTSVR	$1.2444 \pm 0.3051$	$0.0035 \pm 0.0008$	$1.0002 \pm 0.0224$
	WL- $\epsilon$ -TSVR	$1.5722 \pm 0.3142$	$0.0044 \pm 0.0009$	$1.0051 \pm 0.0236$

concluded that the performance of algorithms depends not only on the algorithm itself and the statistical characteristics of the data but also on the choice of parameters. The experimental results verify the effectiveness of training data preprocessing by the wavelet transform method, which can utilize the prior information of samples. The experimental results also show that the introduction of weighting parameters into both quadratic and first-degree empirical risk terms can improve the performance of the model.

### 4.3 Statistical analysis of experimental results

Nonparametric statistical tests are carried out to validate the experimental performance results of the proposed algorithm. The Friedman test [28] ranks the algorithms over six artificial datasets and eight real datasets. The Friedman test uses  $\alpha = 0.05$ , and is distributed according to chi-square with 5 degrees of freedom. The Friedman statistic is 42.04 and 43.01 with  $p$  values of  $5.7845e-8$  and  $3.6806e-8$  for SSE and SSE/SST, respectively. The average ranks obtained by the six algorithms for SSE and SSE/SST are shown in Table 7. We can see that the average SSE and SSE/SST ranks of the WTWTSVR are lower than those of other algorithms.



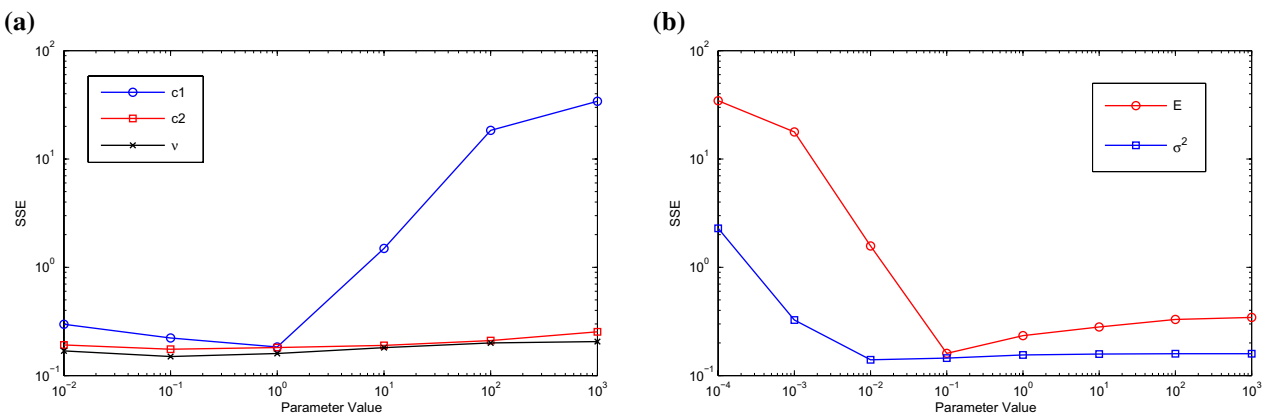
**Fig. 3** SSE value over  $\sigma^2$  of Gaussian function. **a**  $\zeta = 0.1$ , and **b**  $\zeta = 0.2$

**Table 3** Comparisons of WTWTSVR on Function 1 using different kernel functions

Kernel name	Function definition	SSE	SSE/SST	SSR/SST
Gaussian	$\exp(-\ x - x_i\ ^2/e)$	$0.1401 \pm 0.0522$	$0.0041 \pm 0.0015$	$0.9616 \pm 0.0329$
Linear	$xx_i^T$	$27.4681 \pm 0.3409$	$1.0204 \pm 0.0127$	$0.0197 \pm 0.0126$
Polynomial	$(axx_i^T + c)^d$	$21.7358 \pm 0.2398$	$0.6806 \pm 0.0075$	$0.3595 \pm 0.0492$
Sigmoid	$\tanh(axx_i^T + c)$	$0.7425 \pm 0.1419$	$0.0251 \pm 0.0048$	$0.8683 \pm 0.0304$
Multiquadric	$(\ x-x_i\ ^2 + \sigma^2)^{0.5}$	$0.3674 \pm 0.1121$	$0.0110 \pm 0.0034$	$0.8985 \pm 0.0496$
Log	$-\log(1 + \ x-x_i\ ^\sigma)$	$0.2407 \pm 0.1098$	$0.0062 \pm 0.0028$	$0.9578 \pm 0.0419$

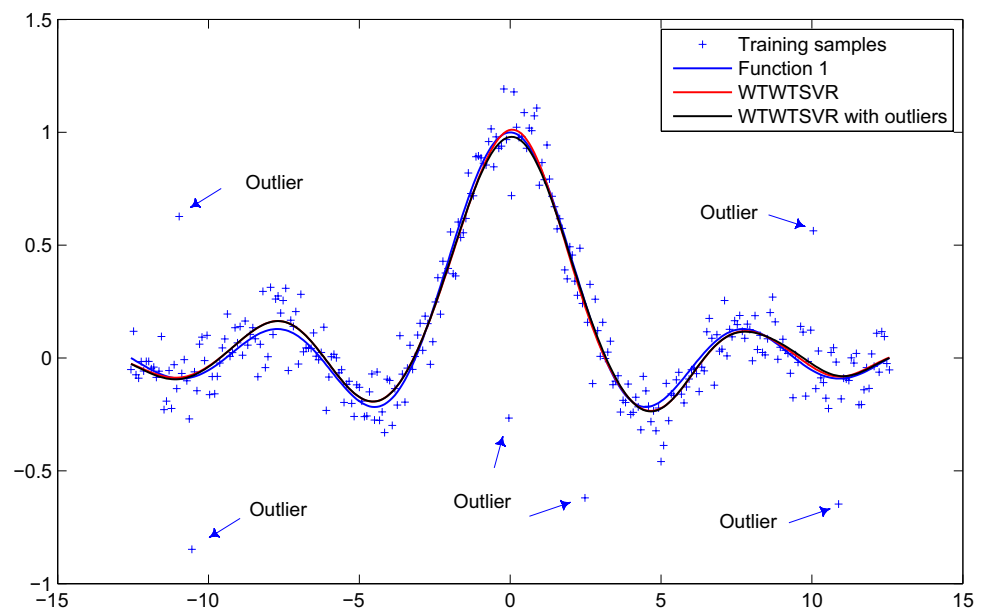
**Table 4** SSE value of regression of artificial functions

SSE	Function 1		Function 2		Function 3	
	$\zeta^2 = 0.1^2$	$\zeta^2 = 0.2^2$	$\zeta^2 = 0.1^2$	$\zeta^2 = 0.2^2$	$\zeta^2 = 0.1^2$	$\zeta^2 = 0.2^2$
With $D$	0.0980	0.4068	2.3739	2.1193	0.5291	1.1793
$D = 1$	0.1095	0.4106	3.7936	3.5630	0.7454	1.2818



**Fig. 4** The influence of parameters  $c_1, c_2, v, E$ , and  $\sigma^2$  on SSE for artificial datasets with Function 1 with noise variance  $\sigma^2 = 0.1^2$

**Fig. 5** Regression performance of WTWTSVR on noised Function 1 with outliers



This shows the superiority of the proposed algorithm, which achieves better performance. In order to check the significant differences between the algorithms, the  $p$ -value is calculated through the Bonferroni–Dunn nonparametric test by computing multiple pairwise comparisons. The test assumes that the performances of two algorithms are significantly different if their average ranks differ by at least some critical value [28]. Table 8 lists the  $p$  values of the Bonferroni–Dunn test for the SSE criteria and SSE/SST ranking results obtained by the Friedman procedure. The null hypothesis of these tests is that there is no difference between the result distributions. From Table 8 we can see that the  $p$  values are very small, which means that the WTWTSVR is quite different from other methods, and therefore confirms the superiority of our method.

From the above analysis, we can conclude that the proposed WTWTSVR can actually improve the performance. The introduction of the weighting parameters into both quadratic and first-degree empirical risk terms are feasible and effective, and the preprocessing of training data by the wavelet transform method can utilize the prior information of samples. It should be noticed that wavelet theory is a powerful denoising tool for a time series signal. Therefore, the proposed method is suitable for dealing with time series datasets. If we deal with non-series samples by the proposed WTWTSVR, performance may be decreased. Additionally, the proposed model is suitable for small sample size datasets, and a large number of training samples will bring about tremendous computational cost.

## 5 Conclusions

In this paper, a novel wavelet transform-based weighted  $\nu$ -twin support vector regression (WTWTSVR) is proposed. Unlike the cases in TSVR and  $\nu$ -TWSVR, samples in different positions in the proposed WTWTSVR model are given by different weights according to the distance between samples, and results are preprocessed by the wavelet transform. Computational comparisons between WTWTSVR and other existing methods are performed on artificial and benchmark datasets to show the better performance of the WTWTSVR, which demonstrates the effectiveness of the proposed method. Furthermore, since one of the theoretical bases of the proposed method is wavelet theory, which is a powerful denoising tool for time series signals, the proposed method is suitable for dealing with time series datasets. If we deal with non-series samples using the proposed WTWTSVR, performance may be decreased. Additionally, SVR is suitable for small sample size datasets, and a large number of training samples will bring about tremendous computational cost. Furthermore, one can notice that the precision of preprocessing and the selection of variance  $\sigma^2$  of the Gaussian function can affect the performance of WTWTSVR. Therefore, for future work, study of parameter selection and the method of preprocessing should be carried out.



**Table 5** Parameter selection for the six algorithms on benchmark datasets

EnergyDatasets	Regressors	Parameters	Parameter values
<i>Energy</i>	WTWTSVR	$c_1, c_2, \nu, E, \sigma^2$	0.01, 100, 0.01, 10, 0.1
	TSVR	$C, \epsilon$	0.01, 10
	$\nu$ -TWSVR	$c_1, c_2, \nu$	0.01, 100, 0.01
	Asy- $\nu$ -TSVR	$C, \nu, p$	0.01, 0.01, 0.5
	KNNUPWTSVR	$c_1/c_2, c_3/c_4$	10, 0.01
	WL- $\epsilon$ -TSVR	$c_1/c_2, \nu_1/\nu_2$	10, 0.01
<i>PM2.5</i>	WTWTSVR	$c_1, c_2, \nu, E, \sigma^2$	0.0001, 1, 0.1, 10, 1
	TSVR	$C, \epsilon$	0.01, 10
	$\nu$ -TWSVR	$c_1, c_2, \nu$	0.0001, 1, 0.1
	Asy- $\nu$ -TSVR	$C, \nu, p$	0.01, 0.1, 0.5
	KNNUPWTSVR	$c_1/c_2, c_3/c_4$	0.0001, 1
	WL- $\epsilon$ -TSVR	$c_1/c_2, \nu_1/\nu_2$	1, 0.0001
<i>AirQuality</i>	WTWTSVR	$c_1, c_2, \nu, E, \sigma^2$	0.001, 1, 0.1, 10, 0.1
	TSVR	$C, \epsilon$	0.001, 0.1
	$\nu$ -TWSVR	$c_1, c_2, \nu$	0.001, 1, 0.1
	Asy- $\nu$ -TSVR	$C, \nu, p$	0.001, 0.1, 0.4
	KNNUPWTSVR	$c_1/c_2, c_3/c_4$	1, 0.0001
	WL- $\epsilon$ -TSVR	$c_1/c_2, \nu_1/\nu_2$	1, 0.001
<i>DowJones</i>	WTWTSVR	$c_1, c_2, \nu, E, \sigma^2$	0.001, 100, 0.01, 10, 0.1
	TSVR	$C, \epsilon$	0.1, 10
	$\nu$ -TWSVR	$c_1, c_2, \nu$	0.001, 100, 0.01
	Asy- $\nu$ -TSVR	$C, \nu, p$	0.1, 0.01, 0.4
	KNNUPWTSVR	$c_1/c_2, c_3/c_4$	10, 0.01
	WL- $\epsilon$ -TSVR	$c_1/c_2, \nu_1/\nu_2$	100, 0.01
<i>Istanbul1</i>	WTWTSVR	$c_1, c_2, \nu, E, \sigma^2$	0.1, 100, 0.001, 10, 0.1
	TSVR	$C, \epsilon$	0.01, 10
	$\nu$ -TWSVR	$c_1, c_2, \nu$	0.1, 100, 0.001
	Asy- $\nu$ -TSVR	$C, \nu, p$	0.01, 0.001, 0.5
	KNNUPWTSVR	$c_1/c_2, c_3/c_4$	1000, 0.01
	WL- $\epsilon$ -TSVR	$c_1/c_2, \nu_1/\nu_2$	100, 0.01
<i>Istanbul2</i>	WTWTSVR	$c_1, c_2, \nu, E, \sigma^2$	0.1, 10, 0.1, 10, 0.1
	TSVR	$C, \epsilon$	0.01, 10
	$\nu$ -TWSVR	$c_1, c_2, \nu$	0.1, 10, 0.1
	Asy- $\nu$ -TSVR	$C, \nu, p$	0.01, 0.1, 0.6
	KNNUPWTSVR	$c_1/c_2, c_3/c_4$	100, 0.01
	WL- $\epsilon$ -TSVR	$c_1/c_2, \nu_1/\nu_2$	10, 0.1
<i>Istanbul4</i>	WTWTSVR	$c_1, c_2, \nu, E, \sigma^2$	0.1, 10, 0.1, 10, 0.1
	TSVR	$C, \epsilon$	0.01, 10
	$\nu$ -TWSVR	$c_1, c_2, \nu$	0.1, 10, 0.1
	Asy- $\nu$ -TSVR	$C, \nu, p$	0.01, 0.1, 0.5
	KNNUPWTSVR	$c_1/c_2, c_3/c_4$	100, 0.01
	WL- $\epsilon$ -TSVR	$c_1/c_2, \nu_1/\nu_2$	10, 0.1
<i>Istanbul8</i>	WTWTSVR	$c_1, c_2, \nu, E, \sigma^2$	0.1, 1000, 0.1, 1, 0.01
	TSVR	$C, \epsilon$	0.01, 10
	$\nu$ -TWSVR	$c_1, c_2, \nu$	0.1, 1000, 0.1
	Asy- $\nu$ -TSVR	$C, \nu, p$	0.01, 0.1, 0.5
	KNNUPWTSVR	$c_1/c_2, c_3/c_4$	100, 0.1
	WL- $\epsilon$ -TSVR	$c_1/c_2, \nu_1/\nu_2$	100, 0.1

**Table 6** Performance comparison of the six algorithms on benchmark datasets

Datasets ( $m * n$ )	Regressors	SSE	SSE/SST	SSR/SST
<i>Energy</i> (500*22)	WTWTSVR	3.6822	0.4861	1.1418
	TSVR	3.8750	0.5116	1.2348
	$\nu$ -TWSVR	3.7306	0.4925	1.1420
	Asy- $\nu$ -TSVR	3.8892	0.5134	1.2384
	KNNUPWTSVR	3.7101	0.4898	1.0769
	WL- $\epsilon$ -TSVR	3.7077	0.4895	1.1190
<i>PM2.5</i> (500*4)	WTWTSVR	6.1543	0.5490	0.4149
	TSVR	6.3487	0.5663	0.3706
	$\nu$ -TWSVR	6.1765	0.5509	0.3863
	Asy- $\nu$ -TSVR	6.3441	0.5659	0.3686
	KNNUPWTSVR	6.2910	0.5612	0.4194
	WL- $\epsilon$ -TSVR	6.1935	0.5524	0.3898
<i>AirQuality</i> (411*9)	WTWTSVR	1.0349	0.0643	0.8615
	TSVR	1.0703	0.0665	0.8314
	$\nu$ -TWSVR	1.0708	0.0666	0.8316
	Asy- $\nu$ -TSVR	1.0708	0.0666	0.8312
	KNNUPWTSVR	1.0626	0.0660	0.8819
	WL- $\epsilon$ -TSVR	1.0844	0.0674	0.8353
<i>DowJones</i> (360*4)	WTWTSVR	0.0458	0.0031	0.9424
	TSVR	0.0509	0.0034	0.9452
	$\nu$ -TWSVR	0.0488	0.0033	0.9423
	Asy- $\nu$ -TSVR	0.0490	0.0033	0.9421
	KNNUPWTSVR	0.0492	0.0033	0.9377
	WL- $\epsilon$ -TSVR	0.0501	0.0033	0.9399
<i>Istanbul1</i> (268*1)	WTWTSVR	5.5722	1.3727	1.1145
	TSVR	6.1766	1.5216	1.5286
	$\nu$ -TWSVR	5.5018	1.3554	1.1387
	Asy- $\nu$ -TSVR	6.1824	1.5230	1.5339
	KNNUPWTSVR	5.7806	1.4240	1.3200
	WL- $\epsilon$ -TSVR	5.7984	1.4284	1.4017
<i>Istanbul2</i> (268*2)	WTWTSVR	5.2725	1.2989	1.2501
	TSVR	5.4127	1.3334	1.4831
	$\nu$ -TWSVR	5.2832	1.3015	1.3316
	Asy- $\nu$ -TSVR	5.4100	1.3327	1.4841
	KNNUPWTSVR	5.6069	1.3813	1.4223
	WL- $\epsilon$ -TSVR	5.2745	1.2994	1.3205
<i>Istanbul4</i> (268*4)	WTWTSVR	4.6502	1.1456	1.4230
	TSVR	4.6725	1.1511	1.6666
	$\nu$ -TWSVR	4.7190	1.1625	1.5161
	Asy- $\nu$ -TSVR	4.6764	1.1520	1.6785
	KNNUPWTSVR	5.1710	1.2739	1.6685
	WL- $\epsilon$ -TSVR	4.7152	1.1616	1.5043
<i>Istanbul8</i> (268*8)	WTWTSVR	3.8933	0.9591	1.2916
	TSVR	4.0682	1.0022	2.0271
	$\nu$ -TWSVR	4.0270	0.9920	1.8629
	Asy- $\nu$ -TSVR	4.0387	0.9949	1.9973
	KNNUPWTSVR	3.9942	0.9840	1.4690
	WL- $\epsilon$ -TSVR	4.3194	1.0641	1.8141

**Table 7** Algorithms' average rankings for SSE and SSE/SST

Datasets	WTWTSVR	TSVR	$\epsilon$ -TSVR	Asy- $\nu$ -TSVR	KNNUPWTSVR	WL- $\epsilon$ -TSVR
SSE	1.0714	4.9286	2.9286	4.9643	3.7143	3.3929
SSE/SST	1.0714	4.9286	3.0714	5.0000	3.6786	3.2500

**Table 8**  $p$  value from Bonferroni–Dunn tests for SSE and SSE/SST, with comparisons to the proposed WTWTSVR

Algorithm	SSE	SSE/SST
TSVR	$4.6033e^{-8}$	$3.2269e^{-8}$
$\epsilon$ -TSVR	$8.4941e^{-3}$	$4.1476e^{-3}$
Asy- $\nu$ -TSVR	$3.4562e^{-8}$	$1.7913e^{-8}$
KNNUPWTSVR	$1.8024e^{-4}$	$1.8626e^{-4}$
WL- $\epsilon$ -TSVR	$1.0029e^{-3}$	$1.7922e^{-3}$

**Acknowledgements** This work was supported by the National Natural Science Foundation of China under Grants 71771112 and 71571091.

## References

- Cheng CT, Wu XY, Chau KW (2005) Multiple criteria rainfall–runoff model calibration using a parallel genetic algorithm in a cluster of computer. *Hydrol Sci J* 50(6):1069–1087
- Wu CL, Chau KW (2011) Rainfall–runoff modeling using artificial neural network coupled with singular spectrum analysis. *J Hydrol* 399(3–4):394–409
- Taormina R, Chau KW, Sivakumar B (2015) Neural network river forecasting through baseflow separation and binary-coded swarm optimization. *J Hydrol* 529(3):1788–1797
- Kazemi SMR, Bidgoli BM, Shamshirband S, Karimi SM, Ghorbani MA, Chau KW, Pour RK (2018) Novel genetic-based negative correlation learning for estimating soil temperature. *Eng Appl Comp Fluid* 12(1):506–516
- Hajikhodaverdikhana P, Nazari M, Mohsenizadeh M, Shamshirband S, Chau KW (2018) Earthquake prediction with meteorological data by particle filter-based support vector regression. *Eng Appl Comp Fluid* 12(1):679–688
- Moazenzadeh R, Mohammadi B, Shamshirband S, Chau KW (2018) Coupling a firefly algorithm with support vector regression to predict evaporation in northern Iran. *Eng Appl Comp Fluid* 12(1):584–597
- Vapnik VN (1998) *Statistical learning theory*. Wiley, New York
- Vapnik VN (1995) *The natural of statistical learning theory*. Springer, New York
- Burges CJC (1998) A tutorial on a support vector machines for pattern recognition. *Data Min Knowl Discov* 2(2):121–167
- Khemchandani JR, Chandra S (2007) Twin support vector machines for pattern classification. *IEEE Trans Pattern Anal* 29(5):905–910
- Peng X (2010) TSVR: an efficient twin support vector machine for regression. *Neural Netw* 23(3):356–372
- Suykens JAK, Lukas L, Dooren V (1999) Least squares support vector machine classifiers: a large scale algorithm. In: *Proceedings of ECCTD, Italy*, pp 839–842
- Suykens JAK, Vandewalle J (1999) Least squares support vector machine classifiers. *Neural Process Lett* 9(3):293–300
- Scholkopf B, Smola AJ, Williamson RC, Bartlett PL (2000) New support vector algorithms. *Neurocomputing* 12(5):1207–1245
- Huang XL, Shi L, Pelckmans K, Suykens JAK (2014) Asymmetric  $\nu$ -tube support vector regression. *Comput Stat Data Anal* 77:371–382
- Huang XL, Shi L, Suykens JAK (2014) Support vector machine classifier with pinball loss. *IEEE Trans Pattern Anal* 36:984–997
- Xu Y, Yang Z, Pan X (2016) A novel twin support vector machine with pinball loss. *IEEE Trans Neural Netw Learn Syst* 28(2):359–370
- Xu Y, Yang Z, Zhang Y, Pan X, Wang L (2016) A maximum margin and minimum volume hyper-spheres machine with pinball loss for imbalanced data classification. *Knowl Based Syst* 95:75–85
- Xu Y, Li X, Pan X, Yang Z (2017) Asymmetric  $\nu$ -twin support vector regression. *Neural Comput Appl* 2:1–16
- Shao Y, Zhang C, Yang Z, Jing L, Deng N (2013) An  $\epsilon$ -twin support vector machine for regression. *Neural Comput Appl* 23:175–185
- Rastogi R, Anand P, Chandra S (2017) A  $\nu$ -twin support vector machine based regression with automatic accuracy control. *Appl Intell* 46:670–683
- Xu Y, Wang L (2014) K-nearest neighbor-based weighted twin support vector regression. *Appl Intell* 41:299–309
- Shevade S, Keerthi S, Bhattacharyya C (2000) Improvements to the SMO algorithm for SVM regression. *IEEE Trans Neural Netw* 11(5):1188–1193
- Lee Y, Hsieh W, Huang C (2005) SSVR: a smooth support vector machine for insensitive regression. *IEEE Trans Knowl Data En* 17(5):678–685
- Ye Y, Bai L, Hua X, Shao Y, Wang Z, Deng N (2016) Weighted Lagrange  $\epsilon$ -twin support vector regression. *Neurocomputing* 197:53–68
- Gupta D (2017) Training primal K-nearest neighbor based weighted twin support vector regression via unconstrained convex minimization. *Appl Intell* 47:962–991
- Blake C, Merz C (1998) UCI repository for machine learning databases. <http://www.ics.uci.edu/mllearn/MLRepository.html>
- Melki G, Cano A, Kecman V, Ventura S (2017) Multi-Target support vector regression via correlation regressor chains. *Info Sci* 415–416:53–69

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.