Applied Intelligence (2019) 49:3061–3081
https://doi.org/10.1007/s10489-019-01422-7

# A projection wavelet weighted twin support vector regression and its primal solution

Lidong Wang[1] · Chuang Gao[1] · Nannan Zhao[1] · Xuebo Chen[1]

Published online: 25 February 2019

**Abstract**
In this paper, an efficient projection wavelet weighted twin support vector regression (PWWTSVR) algorithm is proposed. PWWTSVR determines the regression function by solving a pair of smaller unconstrained minimization problems in primal space, which can reduce computational costs. Classical SVR algorithms give the same emphasis to all training samples, which degrades performance. PWWTSVR gives samples penalty weights determined by wavelet transforms. These are applied to both the quadratic empirical risks term and the first-degree empirical risks term to reduce the influence of outliers. A projection axis in each objective function is sought to minimize the variance of the projected points due to the utilization of a priori information of training data. Therefore, data structure terms are added to the penalty functions. The final regressor can avoid the overfitting problem to a certain extent, and yields great generalization ability. Numerical experiments on artificial and benchmark datasets demonstrate the feasibility and validity of the proposed algorithm.

**Keywords** Machine learning · Support vector regression · Projection variance · Unconstrained minimization · Wavelet transform

## 1 Introduction

Support vector machine (SVM) is a computationally powerful machine learning method that can be used for classification and regression; the SVM for regression is termed an SVR. The adoption of kernel mapping can extend the algorithm to nonlinear cases, which typify real-world problems based on Vapnik's theory [1, 2]. One of the advantages of SVM over other supervised learning methods, such as neural networks, is that it reduces the upper bound of generalization error due to its structural risk-minimization principle. Other strong points include having sparse solutions and operating with a small-scale dataset, which result in its wide use in pattern-recognition.

Along with continuous research progress, many modifications have been proposed in recent years [3–14, 26]. Jayadeva et al. [3] proposed a twin SVM (TSVM) by finding two nonparallel bound functions, which can increase the computational speed by solving two smaller quadratic programming problems (QPPs) instead of one large one in standard SVM. In 2010, Peng [4] extended this strategy to regression applications, resulting in twin support vector regression (TSVR). To reduce the heavy computational costs of QPP problems, Suyken et al. [5, 6] proposed least squares SVM (LS-SVM) for large-scale dataset problems. $\nu$-support vector regression ($\nu$-SVR) [7] extends standard SVR by forcing a fraction of the data samples to lie inside an $\varepsilon$ tube and minimizing the width of the tube by introducing a parameter $\nu$. Inspired by $\nu$-SVR and integrating the concept of pinball loss [8–11], Xu [12] proposed asymmetric $\nu$-twin support vector regression (Asy-$\nu$ -TSVR), which is a kind of twin SVR suitable for dealing with asymmetric noise. In 2013, Shao et al. [13] introduced reg-

✉ Xuebo Chen
x-bchen@163.com

Lidong Wang
wangld5600@163.com

Chuang Gao
13500422153@163.com

Nannan Zhao
723306003@qq.com

[1] School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan, Liaoning, People's Republic of China

ularization terms in objective functions of TSVR, which resulted in $\varepsilon$-twin support vector machine for regression ($\varepsilon$-TSVR), taking account of structural risk minimization rather than the empirical risk minimization principle. $\varepsilon$-TSVR can improve the performance of regression because it partly solves the overfitting problem. Reshma Rastogi et al. proposed a $\nu$-twin support vector machine-based regression method ($\nu$-TWSVR) [14]. The regularization terms and insensitive-bound values $\varepsilon$ in the method were all added into the objective functions to minimize the risks, which can adjust the values of $\varepsilon$ automatically. Peng et al. [15] introduced a pair of projection terms to the optimization problems, with the advantage that the data's structural information is embedded into the learning process, resulting in the decrease of empirical variance values. Melki et al. [16] studied multi-target regression and presented several models for problems with multiple outputs, [18] studied an SVM multiple-instance formulation, and [19] proposed an online learning algorithm for solving L1 SVM. Ding et al. [17] adopted a wavelet kernel function based on the wavelet kernel features and proposed a novel TSVM.

However, all of the training samples in the above methods are considered to have the same status and are given the same penalties, which may degrade performance due to the influence of noise or outliers. It is useful to give the training samples different weights depending on their importance. Xu et al. [20] proposed K-nearest neighbor (KNN)-based weighted twin support vector regression, in which the local information of data is used to improve prediction accuracy. Gupta [21] later presented several solutions in the primal space based on KNN. KNN-based methods are suitable for regression with clustered samples, but not for time series.

Additionally, the traditional regressors of SVR are calculated in approximate dual space. However, it was demonstrated [22] that the approximate dual solution may not result in a good primal approximate solution. Several papers, such as the one mentioned above [21], deal with optimization problems by directly solving the primal problems.

In this paper, we present a TSVR called projection wavelet weighted twin support vector regression (PWWTSVR), which is suitable for time-series data, and is solved in the primal space. Combining the idea of the projection axis [15] and $\varepsilon$-TSVR, PWWTSVR finds a pair of bound functions by solving two smaller-sized optimization problems. More importantly, it introduces a weight matrix based on a wavelet transform that can reduce the influence of noise in the training samples. The regularization terms and data structure terms are all embedded in the objection functions, which can combine the merits of $\varepsilon$-TSVR and the projection method. Moreover, regressors of the proposed algorithm are solved in terms of unconstrained primal problems rather than solving their dual problems, which can improve learning speed. The numerical results show that the algorithm has better generalization ability. The key contributions of this work include the following:

1. A weight matrix is introduced to both the quadratic and first-degree empirical risks terms to reduce the influence of outliers. This is significantly different from [23] and [20], and is an expansion of traditional methods. The weight matrix, which represents the distance of noised samples and its 'real position,' can reflect the prior information of the training samples. Larger weights are given to samples with less noise, and smaller weights to those with more noise. The addition of $D$ to the objection function can reduce the impact of noise and outliers.
2. Wavelet transform theory is adopted to calculate a weight matrix, which is a new angle of preprocessing samples. A wavelet transform is a kind of time-frequency representation for signals, and the proposed method based on wavelet theory is suitable for dealing with time-sequence samples due to the character of the wavelet transform.
3. The proposed objective functions are proved mathematically to be convex, so global and unique solutions can be obtained. The objective functions are added projection terms and are solved in the primal space, which can improve performance and reduce the computational complexity.

This paper is organized as follows. Section 2 briefly describes $\varepsilon$ -TSVR and TPSVR. Section 3 proposes PWWTSVR. Experimental results are described in Section 4 to investigate the validity of the proposed algorithm, and Section 5 provides concluding remarks.

## 2 Brief introduction to $\varepsilon$-TSVR and TPSVR

In this section, the classical $\varepsilon$-TSVR [13] and TPSVR [15] algorithms are described briefly. Assume a training set $T = \{(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)\}$, where $x_i \in \Re^n$ and $y_i \in \Re$, $i = 1, 2, ..., m$. Then the output vector of the training data can be denoted as $Y = (y_1, y_2, ..., y_m)^T \in \Re^m$ and the input matrix as $A = (A_1, A_2, ..., A_m)^T \in \Re^{m\times n}$ , and the $i$-th row $A_i = (x_{i1}, x_{i2}, ..., x_{in})^T$ is the $i$-th training sample. Let $e$ and $I$ be a ones vector and an identity matrix of appropriate dimensions, respectively. The kernel mapping can be expressed as $K(\cdot, \cdot)$.

### 2.1 Twin support vector regression ($\varepsilon$-TSVR)

Proposed by Shao et. al., $\varepsilon$-TSVR is an efficient regression method with low computational costs compared to SVR [24, 25]. $\varepsilon$-TSVR has two $\varepsilon$-insensitive-bounds: down-bound denotes $f_1(x) = w_1^T x + b_1$ , and up-bound denotes $f_2(x) = w_2^T x + b_2$, where $w_1, w_2 \in \Re^n$, $b_1, b_2 \in \Re$. Then, the regressor $f(x)$ can be gotten by taking the mean of the two bound functions, i.e.,

$$f(x) = \frac{1}{2}(f_1(x) + f_2(x)) = \frac{1}{2}(w_1 + w_2)^T x + \frac{1}{2}(b_1 + b_2). \tag{1}$$

By introducing the regularization terms $\frac{1}{2}(w_1^T w_1 + b_1^2)$ and $\frac{1}{2}(w_2^T w_2 + b_2^2)$, the primal problems can be expressed as follows:

$$\begin{aligned} &\min_{w_1, b_1, \xi} \ \tfrac{1}{2}c_3(w_1^T w_1 + b_1^2) + \tfrac{1}{2}\|Y - (Aw_1 + eb_1)\|^2 + c_1 e^T \xi \\ &\text{s.t.} \ \ Y - (Aw_1 + eb_1) \geqslant -\varepsilon_1 e - \xi, \ \xi \geqslant 0 \end{aligned} \tag{2}$$

and

$$\begin{aligned} &\min_{w_2, b_2, \eta} \ \tfrac{1}{2}c_4(w_2^T w_2 + b_2^2) + \tfrac{1}{2}\|Y - (Aw_2 + eb_2)\|^2 + c_2 e^T \eta \\ &\text{s.t.} \ \ (Aw_2 + eb_2) - Y \geqslant -\varepsilon_2 e - \eta, \ \eta \geqslant 0 \end{aligned} \tag{3}$$

where $c_1$, $c_2$, $c_3$, $c_4$, $\varepsilon_1$ and $\varepsilon_2$ are positive parameters. The main advantage of $\varepsilon$-TSVR is the introduction of the extra regularization terms $\frac{1}{2}c_3(w_1^T w_1 + b_1^2)$ and $\frac{1}{2}(w_2^T w_2 + b_2^2)$, thus the structural risk-minimization principle is implemented.

After introducing the Lagrangian functions of (2) and (3), and considering the corresponding Karush-Kuhn-Tucker (K.K.T.) necessary and sufficient optimality conditions, the dual QPPs can be obtained as

$$\begin{aligned} &\min_{\alpha} \tfrac{1}{2}\alpha^T G(G^T G + c_3 I)^{-1} G^T \alpha - Y^T G(G^T G + c_3 I)^{-1} G^T \alpha + (e_1^T \varepsilon + Y^T)\alpha \\ &\text{s.t.} \qquad 0 \leqslant \alpha \leqslant c_1 e \end{aligned} \tag{4}$$

and

$$\begin{aligned} &\min_{\gamma} \tfrac{1}{2}\beta^T G(G^T G + c_4 I)^{-1} G^T \beta + Y^T G(G^T G + c_4 I)^{-1} G^T \beta - (Y^T - e_2^T \varepsilon)\beta \\ &\text{s.t.} \qquad 0 \leqslant \beta \leqslant c_2 e \end{aligned} \tag{5}$$

where $G = [\,A \ \ e\,]$. Once the QPPs (4) and (5) are solved, the optimized parameters $u_1, u_2$ can be obtained as

$$u_1 = [\,w_1^T \ \ b_1\,]^T = (G^T G + c_3 I)^{-1} G^T (Y - \alpha) \tag{6}$$
$$u_2 = [\,w_2^T \ \ b_2\,]^T = (G^T G + c_4 I)^{-1} G^T (Y + \beta). \tag{7}$$

The final regressor can be calculated by (1). It can be seen that the structural risk is taken into account in the model, which can partly overcome the problem of over-fitting. However, the prior information of training data is not used. That is, the two bound functions do not depict the data structure.

## 2.2 Twin projection support vector regression (TPSVR)

Peng proposed a modified TSVR, called TPSVR [15], which also finds a pair of insensitive bound functions by two smaller-sized QPPS. In each QPP, TPSVR finds a projection axis, $\hat{w}_k = [w_k; -1]$, $k = 1, 2$, which is normal to the line of the bound regression functions. The projection axis is meant to make the projected zone or the variance of the projected noise as small as possible. In mathematics, this idea can be expressed as:

$$\begin{aligned} &\min \frac{1}{2n}\sum_{i=1}^{n}(\hat{w}_k^T z_i - \hat{w}_k \mu_z)^2 \\ &= \min \frac{1}{2n}\sum_{i=1}^{n}\hat{w}_k^T (z_i - \mu_z)(z_i - \mu_z)^T \hat{w}_k^T \\ &= \min \frac{1}{2}\hat{w}_k^T \Sigma_z \hat{w}_k \\ &= \min \frac{1}{2}(w_k^T \Sigma_x w_k + \Sigma_y) - w_k^T \Sigma_{xy} \end{aligned} \tag{8}$$

where $z_i$ is the training point $z_i = (x_i; y_i)$, $i = 1, ..., n$, $\mu_z$ is the centroid point of $z_i$, and $\Sigma_z$ is the covariance matrix of $z_i$, $\Sigma_x$ and $\Sigma_y$ are the empirical covariance matrices of inputs and responses, $\Sigma_{xy}$ is the empirical correlation coefficient matrix between the inputs and responses. They are defined as

$$\begin{aligned}
\Sigma_z &= \frac{1}{n}\sum_{i=1}^{n}(z_i-\mu_z)(z_i-\mu_z)^T,\\
\Sigma_x &= \frac{1}{n}\sum_{i=1}^{n}(x_i-\mu_x)(x_i-\mu_x)^T,\\
\Sigma_y &= \frac{1}{n}\sum_{i=1}^{n}(y_i-\mu_y)(y_i-\mu_y)^T,\\
\Sigma_{xy} &= \frac{1}{n}\sum_{i=1}^{n}x_i y_i-\mu_x\mu_y, \qquad (9)
\end{aligned}$$

where $\mu_x$ and $\mu_y$ are the respective centroid points of the inputs and outputs. Noting that $\Sigma_y$ has a fixed value for a given training example, the objective function (8) can be expressed as

$$\min \frac{1}{2}\hat{w}_k^T\Sigma_z\hat{w}_k = \min \frac{1}{2}w_k^T\Sigma_x w_k - w_k^T\Sigma_{xy} \qquad (10)$$

The TPSVR model is as follows:

$$\begin{aligned}
\min\ & \tfrac{1}{2}w_1^T w_1 + \tfrac{\lambda_1}{2}\hat{w}_1^T\Sigma_z\hat{w}_1 - \tfrac{\upsilon_1}{n}\sum_{i=1}^{n}(w_1^T x_i + b_1) + \tfrac{c_1}{n}\sum_{i=1}^{n}\xi_i \\
\text{s.t.}\ & y_i \geq w_1^T x_i + b_1 - \xi_i,\ \xi_i \geq 0,
\end{aligned} \qquad (11)$$

$$\begin{aligned}
\min\ & \tfrac{1}{2}w_2^T w_2 + \tfrac{\lambda_2}{2}\hat{w}_2^T\Sigma_z\hat{w}_2 - \tfrac{\upsilon_2}{n}\sum_{i=1}^{n}(w_2^T x_i + b_2) + \tfrac{c_2}{n}\sum_{i=1}^{n}\eta_i \\
\text{s.t.}\ & y_i \leq w_2^T x_i + b_2 + \eta_i,\ \eta_i \geq 0,
\end{aligned} \qquad (12)$$

where $\lambda_k, \upsilon_k, c_k,\ k = 1, 2$, are the trade-off factors of the terms.

Solutions of the TPSVR model and other details referred to Peng [15]. This model is suitable for many problems due to the introduction of projection axes, where the projected points in the normal directions have minimized variances. Although some performance improvement is obtained through the model, samples with different amounts of noise have the same effect on the regression functions, which degrades performance.

## 3 Projection wavelet weighted twin support vector regression (PWWTSVR)

Inspired by $\varepsilon$-TSVR and TPSVR, a projection wavelet weighted TSVR (PWWTSVR) is presented in this section. Similar to other TSVRs, PWWTSVR is constructed by two non-parallel hyperplanes, down-bound $f_1(x) = w_1^T x + b_1$ and up-bound $f_2(x) = w_2^T x + b_2$, where each determines the $\varepsilon$ -insensitive bound regressor, and the final regressor is also $f(x) = \frac{1}{2}(f_1(x) + f_2(x))$.

### 3.1 Linear PWWTSVR

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

$$\begin{aligned}
\min_{w_1,b_1,\xi_1}\ & L_1 = \tfrac{1}{2}(Y-(Aw_1+eb_1))^T D(Y-(Aw_1+eb_1)) + \tfrac{1}{2}c_{11}(w_1^T w_1 + b_1^2) \\
& +\tfrac{1}{2}c_{12}\hat{w}_1^T\Sigma_z\hat{w}_1 + c_{13}e^T D\xi_1 \\
\text{s.t.}\ & Y-(Aw_1+eb_1) \geqslant -\varepsilon_1 e - \xi_1,\ \xi_1 \geqslant 0e\,,
\end{aligned} \qquad (13)$$

and

$$\begin{aligned}
\min_{w_2,b_2,\xi_2}\ & L_2 = \tfrac{1}{2}(Y-(Aw_2+eb_2))^T D(Y-(Aw_2+eb_2)) + \tfrac{1}{2}c_{21}(w_2^T w_2 + b_2^2) \\
& +\tfrac{1}{2}c_{22}\hat{w}_2^T\Sigma_z\hat{w}_2 + c_{23}e^T D\xi_2 \\
\text{s.t.}\ & (Aw_2+eb_2)-Y \geqslant -\varepsilon_2 e - \xi_2,\ \xi_2 \geqslant 0e\,,
\end{aligned} \qquad (14)$$

where $c_{11}, c_{12}, c_{13}, c_{21}, c_{22}, c_{23} > 0$ are parameters chosen a priori by the user; $\varepsilon_1, \varepsilon_2$ are insensitive parameters; and $\xi_1, \xi_2$ are slack vectors to measure the errors of samples outside the "$\varepsilon$ tube". $D \in \Re^{m\times m}$ is a weighting matrix, which will be discussed later, and $m$ is the number of training points. Figure 1 shows the geometric interpretation of linear PWWTSVR as an example.

If we omit the weight matrix $D$, the first term in the objective function of (13) has the same expression as the second term of the $\epsilon$-TSVR objective function (2). It is the sum of weighted squared distances from training points to the down-bound function, which is called empirical risk. Minimizing this causes the function $f_1(x)$ to fit the training samples and avoid under-fitting. The second term is a regularization term, which can make $f_1(x)$ as smooth as possible. The structural risk minimization is implemented by minimizing the regularization term $\frac{1}{2}(w_1^T w_1 + b_1^2)$. A small value of $\frac{1}{2}(w_1^T w_1 + b_1^2)$ corresponds to the function (13) being flat. The third term, the data structure term, can minimize empirical variance values of projected points on the down-bound functions. The fourth term aims to minimize the sum of errors of the points lower than the down-bound $f_1(x)$, which can possibly over-fit the training points. The ratios of the four penalty terms in the objective function of (13) can be adjusted by the choice of $c_{11}$, $c_{12}$, and $c_{13}$.

To solve the optimization problems (13) and (14), we convert the constrained problems to a pair of unconstrained problems by introducing the plus function $(\cdot)_+$ as

$$\min_{w_1,b_1} L_1 = \tfrac{1}{2}(Y-(Aw_1+eb_1))^T D(Y-(Aw_1+eb_1)) + \tfrac{1}{2}c_{11}(w_1^T w_1 + b_1^2) + \frac{1}{2}c_{12}\hat{w}_1^T \Sigma_z \hat{w}_1 + c_{13}e^T D((Aw_1+eb_1)-(Y+\varepsilon_1 e))_+ \tag{15}$$

and

$$\min_{w_2,b_2} L_2 = \tfrac{1}{2}(Y-(Aw_2+eb_2))^T D(Y-(Aw_2+eb_2)) + \tfrac{1}{2}c_{21}(w_2^T w_2 + b_2^2) + \frac{1}{2}c_{22}\hat{w}_2^T \Sigma_z \hat{w}_2 + c_{23}e^T D((Y-\varepsilon_2 e)-(Aw_1+eb_1))_+ \tag{16}$$

Plus functions that are not differentiable can be replaced by smooth approximate functions $\mathbf{p}(\cdot)$, and the fourth terms of (15) and (16) are respectively replaced by

$$L_{12} = c_{13}e^T D\mathbf{p}(Gu_1 - f_1), \tag{17}$$

and

$$L_{22} = c_{23}e^T D\mathbf{p}(f_2 - Gu_2), \tag{18}$$

where $f_1 = (Y+\varepsilon_1 e)$, $f_2 = (Y-\varepsilon_2 e)$, $u_1 = [w_1^T, b_1]^T$, $u_2 = [w_2^T, b_2]^T$, and $G = [A, e]$. For simplicity, we define

$$L_{11}(w_1,b_1) = \frac{1}{2}(Y-(Aw_1+eb_1))^T D(Y-(Aw_1+eb_1)) + \frac{1}{2}c_{11}(w_1^T w_1 + b_1^2) + \frac{1}{2}c_{12}\hat{w}_1^T \Sigma_z \hat{w}_1, \tag{19}$$

$$L_{21}(w_2,b_2) = \frac{1}{2}(Y-(Aw_2+eb_2))^T D(Y-(Aw_2+eb_2)) + \frac{1}{2}c_{21}(w_2^T w_2 + b_2^2) + \frac{1}{2}c_{22}\hat{w}_2^T \Sigma_z \hat{w}_2. \tag{20}$$

Then the objective functions (15) and (16) are converted to differentiable functions as follows:

$$\min_{w_1,b_1} L_1 = L_{11} + L_{12} \tag{21}$$

$$\min_{w_2,b_2} L_2 = L_{21} + L_{22} \tag{22}$$

In this paper, we adopt the sigmoid integral function as a smooth function; it is defined as

$$\mathbf{p}(x) = x + \frac{1}{\alpha}\log(1+\exp(-\alpha x)). \tag{23}$$

where $\alpha$ is a positive real constant. Note that $L_1$ in (21) and $L_2$ in (22) are convex (see Theorem 1 ). Global and unique solutions can be gotten, and the Newton iterative approach adopted later in this paper will be convergent. In Section 3.2, the Newton iterative approach will be adopted to solve the minimization problems.

*Remark 1* The differences between TPSVR and PWWTSVR are as follows.

(1) The weighting matrix $D$ is inserted in the first and last term in (13). The weighting matrix $D$ is calculated by wavelet transform theory, and the addition of $D$ can decrease the influence of noise and outliers.
(2) The second term, $\frac{1}{2}c_{11}(w_1^T w_1 + b_1^2)$, in (13) is different from the first term, $\frac{1}{2}w_1^T w_1$, in (11). Adding $b_1$ to the objective function can optimize the offset parameter $b_1$ and improve the performance.
(3) The third term in the objective function of (11) is meant to optimize the sum of estimated values of training points, and optimizing it causes the regression function $f_1(x)$ to be as large as possible [15]. The constitution of this form has benefited in the deduction of dual QPP (see (20) in [15]). Compared with the third term in the objective function of (11), the first term in the objective function of (13) of the proposed algorithm has a definite meaning. It is the sum of weighted squared distances from training points to the down-bound function, and it can minimize the empirical risk.

For the optimization problem (14), we have similar illustrations.

## 3.2 Newton iterative approach

First, substitute (10) into (19) and rewrite (19) as

$$L_{11}(w_1,b_1) = \frac{1}{2}(Y-(Aw_1+eb_1))^T D(Y-(Aw_1+eb_1))$$

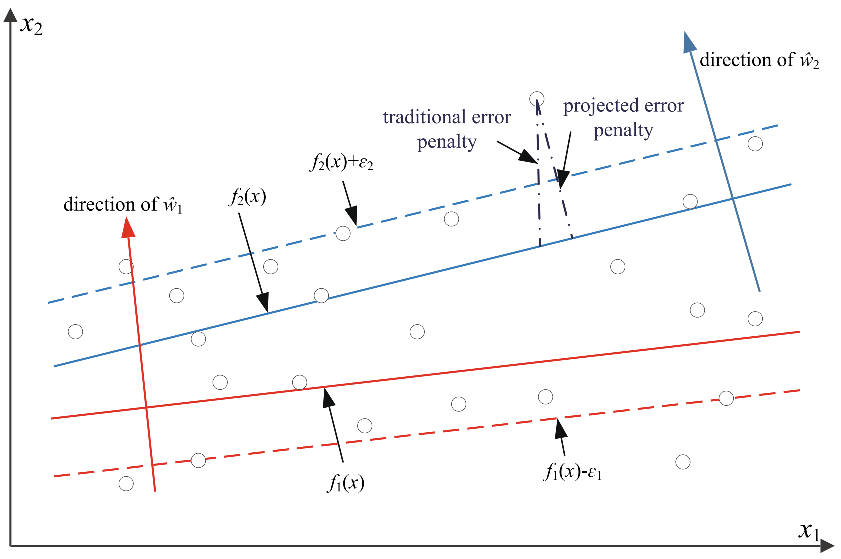


**Fig. 1** Geometric interpretation of PWWTSVR

$$
\begin{aligned}
&+\frac{1}{2}c_{11}(w_1^T w_1 + b_1^2)\\
&+\frac{1}{2}c_{12}w_1^T \Sigma_x w_1 - c_{12}w_1^T \Sigma_{xy}\\
=\ &\frac{1}{2}Y^T DY - Y^T D(Aw_1 + eb_1)\\
&+\frac{1}{2}w_1^T A^T DAw_1 + e^T b_1 DAw_1 + \frac{1}{2}b_1^2 e^T De\\
&+\frac{1}{2}c_{11}(w_1^T w_1 + b_1^2) + \frac{1}{2}c_{12}w_1^T \Sigma_x w_1\\
&-c_{12}w_1^T \Sigma_{xy}\\
=\ &\frac{1}{2}w_1^T(c_{11}I + A^T DA + \Sigma_x)w_1\\
&+(-Y^T DA + e^T b_1 DA - c_{12}\Sigma_{xy}^T)w_1\\
&+\frac{1}{2}(c_{11} + e^T De)b_1^2 - Y^T Deb_1 + \frac{1}{2}Y^T DY \qquad (24)
\end{aligned}
$$

The gradient of $L_{11}$ over $w_1$ and $b_1$ can be deduced as

$$
\begin{aligned}
\nabla L_{11}(w_1) = &(c_{11}I + A^T DA + \Sigma_x)w_1\\
&+A^T Deb_1 - (A^T DY + c_{12}\Sigma_{xy}) \qquad (25)
\end{aligned}
$$

$$\nabla L_{11}(b_1) = e^T DAw_1 + (c_{11} + e^T De)b_1 - e^T DY \qquad (26)$$

Let $u_1 = [w_1^T, b_1]^T$ as previous, and define $Q_1 = \begin{bmatrix} c_{11}I + A^T DA + \Sigma_x & A^T De \\ e^T DA & c_{11} + e^T De \end{bmatrix}$, $P_1 = \begin{bmatrix} A^T DY + c_{12}\Sigma_{xy} \\ e^T DY \end{bmatrix}$. The gradient of $L_{11}$ over $u_1$ is

$$\nabla L_{11}(u_1) = Q_1 u_1 - P_1. \qquad (27)$$

The second-order gradient of $L_{11}$ over $u_1$ can be deduced as

$$\nabla^2 L_{11}(u_1) = Q_1. \qquad (28)$$

Similarly, the second-order gradients of $L_{21}$ are expressed as

$$\nabla L_{21}(u_1) = Q_2 u_2 - P_2, \qquad (29)$$

$$\nabla^2 L_{21}(u_2) = Q_2. \qquad (30)$$

where $u_2 = [w_2^T, b_2]^T$, $Q_2 = \begin{bmatrix} c_{21}I + A^T DA + \Sigma_x & A^T De \\ e^T DA & c_{21} + e^T De \end{bmatrix}$, and $P_2 = \begin{bmatrix} A^T DY + c_{22}\Sigma_{xy} \\ e^T DY \end{bmatrix}$.

Second, the first- and second-order gradients of $L_{12}$ and $L_{22}$ can be calculated as

$$\nabla L_{12}(u_1) = c_{13}G^T Ddiag\left(\frac{1}{1 + \exp(-\alpha(Gu_1 - f_1))}\right) \qquad (31)$$

$$\nabla^2 L_{12}(u_1) = \alpha c_{13}G^T Ddiag\left(\frac{\exp(-\alpha(Gu_1 - f_1))}{1 + \exp(-\alpha(Gu_1 - f_1))^2}\right)G \qquad (32)$$

$$\nabla L_{22}(u_2) = -c_{23}G^T Ddiag\left(\frac{1}{1 + \exp(-\alpha(f_2 - Gu_2))}\right) \qquad (33)$$

$$\nabla^2 L_{22}(u_2) = \alpha c_{23}G^T Ddiag\left(\frac{\exp(-\alpha(f_2 - Gu_2))}{1 + \exp(-\alpha(f_2 - Gu_2))^2}\right)G. \qquad (34)$$

Thus the first- and second-order gradients of $L_1$ in (21) are deduced, and substituting (27), (28), (31), and (32) in it results in

$$\nabla L_1(u_1) = Q_1 u_1 - P_1 + c_{13} G^T Ddiag\left(\frac{1}{1+\exp(-\alpha(Gu_1 - f_1))}\right) \quad (35)$$

$$\nabla^2 L_1(u_1) = Q_1 + \alpha c_{13} G^T Ddiag\left(\frac{\exp(-\alpha(Gu_1 - f_1))}{1+\exp(-\alpha(Gu_1 - f_1))^2}\right) G. \quad (36)$$

Similarly, for $L_2$,

$$\nabla L_2(u_2) = Q_2 u_2 - P_2 - c_{23} G^T Ddiag\left(\frac{1}{1+\exp(-\alpha(f_2 - Gu_2))}\right) \quad (37)$$

$$\nabla^2 L_2(u_2) = Q_2 + \alpha c_{23} G^T Ddiag\left(\frac{\exp(-\alpha(f_2 - Gu_2))}{1+\exp(-\alpha(f_2 - Gu_2))^2}\right) G. \quad (38)$$

The iterative solutions of minimization problems (15) and (16) can be obtained by adopting the Newton method and using (35)–(38 ), as follows:

$$u_1^{k+1} = u_1^k - (\nabla^2 L_1(u_1))^{-1} (\nabla L_1(u_1)), \quad (39)$$

$$u_2^{k+1} = u_2^k - (\nabla^2 L_2(u_2))^{-1} (\nabla L_2(u_2)). \quad (40)$$

**Theorem 1** *$L_1$ in (21) and $L_2$ in (22) are convex.*

To prove Theorem 1, the following lemmas are needed.

**Lemma 1** [27] *Let $A \in \Re^{n\times n}$ be a real symmetric matrix. Then A is positive definite if and only if there is a $B \in \Re^{m\times n}$ such that $A = B^T B$.*

**Lemma 2** [27] *Suppose that $A = [a_{ij}] \in \Re^{n\times n}$ is real symmetric and strictly diagonally dominant. If $a_{ii} > 0$ for all $i = 1, 2, ...n$, then A is positive definite.*

**Lemma 3** [28] *Let A be a real symmetric matrix partitioned as $A = \begin{bmatrix} A_{11} & A_{12} \\ A_{12}^T & A_{22} \end{bmatrix}$, in which $A_{22}$ is square and nonsingular. Then $A > 0$ (A is positive definite, the same below) if and only if both $A_{22} > 0$ and $A/A_{22} > 0$. Here, $A/A_{22}$ is the Schur complement of $A_{22}$ in A, i.e., $A/A_{22} = A_{11} - A_{12} A_{22}^{-1} A_{12}^T$.*

*Proof* Here, $L_1$ is an example for proving the theorem. It can be seen that if the second-order gradients of $L_1$, $\nabla^2 L_1(u_1) > 0$, then $L_1$ is convex. Now, we prove $\nabla^2 L_1(u_1) > 0$. Rewrite (36) as

$$\begin{aligned} \nabla^2 L_1(u_1) &= \nabla^2 L_{11}(u_1) + \nabla^2 L_{12}(u_1) \\ &= \begin{bmatrix} c_{11} I + A^T DA + \Sigma_x & A^T De \\ e^T DA & c_{11} + e^T De \end{bmatrix} \\ &\quad + \alpha c_{13} G^T Ddiag\left(\frac{\exp(-\alpha(Gu_1 - f_1))}{1+\exp(-\alpha(Gu_1 - f_1))^2}\right) G. \end{aligned} \quad (41)$$

The diagonal matrix $D$ and $diag\left(\frac{\exp(-\alpha(Gu_1 - f_1))}{1+\exp(-\alpha(Gu_1 - f_1))^2}\right)$ are positive, so the real symmetric matrix $Ddiag\left(\frac{\exp(-\alpha(Gu_1 - f_1))}{1+\exp(-\alpha(Gu_1 - f_1))^2}\right)$ can be decomposed as $B^T B$, where $B$ is a square root matrix of $Ddiag\left(\frac{\exp(-\alpha(Gu_1 - f_1))}{1+\exp(-\alpha(Gu_1 - f_1))^2}\right)$. It can be checked that $\nabla^2 L_{12}(u_1)$ can be decomposed as

$$\nabla^2 L_{12}(u_1) = \alpha c_{13} (GB)^T (GB). \quad (42)$$

Note that $\alpha$ and $c_{13}$ are positive scalars; based on Lemma 1, it can be seen that $\nabla^2 L_{12}(u_1)$ is positive definite.

For the first term, the Schur complement of $c_{11} + e^T De$ in $\nabla^2 L_{11}(u_1)$ can be written in the for

$$\begin{aligned} \nabla^2 L_{11}(u_1)/(c_{11} + e^T De) &= c_{11} I + A^T DA + \Sigma_x - A^T De (c_{11} + e^T De)^{-1} e^T DA \\ &= c_{11} I + A^T DA + \Sigma_x - \frac{1}{c_{11} + \sum d_i} A^T \begin{bmatrix} d_1^2 & d_1 d_2 & \dots & d_1 d_m \\ d_1 d_2 & & & \\ \dots & & \dots & \\ d_1 d_m & & & d_m^2 \end{bmatrix} A \\ &= c_{11} I + \Sigma_x + \frac{1}{c_{11} + \sum d_i} A^T \begin{bmatrix} (c_{11} + \sum d_i) d_1 - d_1^2 & -d_1 d_2 & \dots & -d_1 d_m \\ -d_1 d_2 & & & \\ \dots & & \dots & \\ -d_1 d_m & & & (c_{11} + \sum d_i) d_m - d_m^2 \end{bmatrix} A \end{aligned} \quad (43)$$

where positive scalar $d_i$ is the $i$th main diagonal element of $D$, $i = 1, 2, ..., m$.

Let

$$M = \begin{bmatrix} (c_{11} + \sum d_i)d_1 - d_1^2 & -d_1 d_2 & \dots & -d_1 d_m \\ -d_1 d_2 & & & \\ \dots & & \dots & \\ -d_1 d_m & & & (c_{11} + \sum d_i)d_m - d_m^2 \end{bmatrix} \tag{44}$$

For the $j$th column of $M$, $j = 1, 2, ..., m$, main diagonal element minus other elements, it can be obtained that

$$\begin{aligned} &(c_{11} + \sum_{i=1}^{m} d_i)d_j - d_j^2 - \sum_{i=1, i\neq j}^{m} d_i d_j \\ &= (c_{11} + \sum_{i=1}^{m} d_i)d_j - \sum_{i=1}^{m} d_i d_j \\ &= c_{11} > 0 \end{aligned} \tag{45}$$

It can be seen from (45) that $M$ is real symmetric and strictly diagonally dominant, and the coefficients of $M$, $1/(c_{11} + \sum d_i) > 0$, so, based on Lemma 2, the third term of (43) is positive definite.

From the definition of $\Sigma_x$, it is known that if $\Sigma_x > 0$ and $c_{11}I > 0$, then (43) is positive definite. According to Lemma 3 , it can be obtained that $\nabla^2 L_1(u_1) > 0$, and similarly, $\nabla^2 L_2(u_2) > 0$. Then, Theorem 1 is proved. $\square$

## 3.3 Nonlinear PWWTSVR

To extend the application of PWWTSVR to nonlinear cases in the real world, a kernel trick is adopted to map the input to a higher-dimensional feature space, i.e., $x \to \varphi(x)$. However, $\varphi(x)$ lacks an explicit formulation due to the higher dimensions, which prevents the computation of $\Sigma_{\varphi(x)}$. In Peng's work [15], the eigenvalue decomposition method is adopted to explicitly map to the empirical feature space. Let $K(A, A^T)$ denote an $m \times m$ matrix of rank $r$, where $K$ is an appropriately chosen kernel. Since $K(A, A^T)$ is a symmetric positive-semidefinite matrix, it can be decomposed as $K(A, A^T) = P_{m\times r} \Lambda P_{r\times m}^T$ where $\Lambda$ is a diagonal matrix containing only the $r$ positive eigenvalues of $K(A, A^T)$ in decreasing order, and $P_{m\times r}$ consists of the eigenvectors corresponding to the positive eigenvalues. The mapping from the input data space to the kernel space is expressed as $x \to \varphi(x) = \Lambda^{-1/2} P^T K(x, A^T)$.

Then, for nonlinear cases, the matrix $A$ in linear cases can be replaced by $K(A, A^T)$, and $x$ by $\varphi(x)$. For example, (24) and (17) can be transformed as

$$\begin{aligned} L_{11}(w_1, b_1) = &\frac{1}{2} w_1^T (c_{11} I + K(A, A^T)^T D K(A, A^T) \\ &+ \Sigma_{\varphi(x)}) w_1 + (-Y^T D K(A, A^T) \\ &+ e^T b_1 D K(A, A^T) - c_{12} \Sigma_{\varphi(x)y}^T) w_1 \\ &+ \frac{1}{2} (c_{11} + e^T D e) b_1^2 - Y^T D e b_1 + \frac{1}{2} Y^T D Y \end{aligned} \tag{46}$$

and

$$L_{12} = c_{13} e^T D \mathbf{p}(G^* u_1 - f_1), \tag{47}$$

respectively, where $G^* = [K(A, A^T), e]$. Other formulas can be transformed similarly.

After calculating $u_1 = [w_1^T, b_1]^T$ and $u_2 = [w_2^T, b_2]^T$, the regressor $f(x)$ can be gotten by (1).

## 3.4 Weighting parameters determined by wavelet transform

The parameter mentioned in the first subsection, $D \in \Re^{m\times m}$, is a weighting matrix. It should be determined beforehand according to the importance of training data. By direct observation of the objective functions of $\varepsilon$-TSVR and TPSVR, 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, where a larger weight means a sample is more important. Motivated by this idea, the weighting parameter $D$ is determined by the following Gaussian function:

$$D = diag\left(E \exp\left(-\left|Y - \hat{Y}\right|^2 / \sigma^2\right)\right), \tag{48}$$

where $E$ is the peak value of the Gaussian function, $\sigma$ represents the standard deviation, and $\hat{Y}$ is the estimated value vector of the output $Y$. Figure 2 shows the weight value of PWWTSVR and unweighted TSVR over the error of samples as an example.

The wavelet transform is a method to deal with time sequences. For non-time sequences, other regressing algorithms, such as TSVR, as mentioned above, can be adopted as a pre-regressor to calculate $\hat{Y}$. In this paper, wavelet filters are adopted to calculate $\hat{Y}$ by three stages, as follows.

(1). Wavelet transforms may be considered forms of time-frequency representation for signals, so they are related to harmonic analysis. Discrete wavelet transforms (DWTs) use a discrete-time filterbank. The DWT of a signal in the $l$-th decomposition step $xa_l(n)$ is calculated by a series of filters. The samples are passed through a low-pass filter with

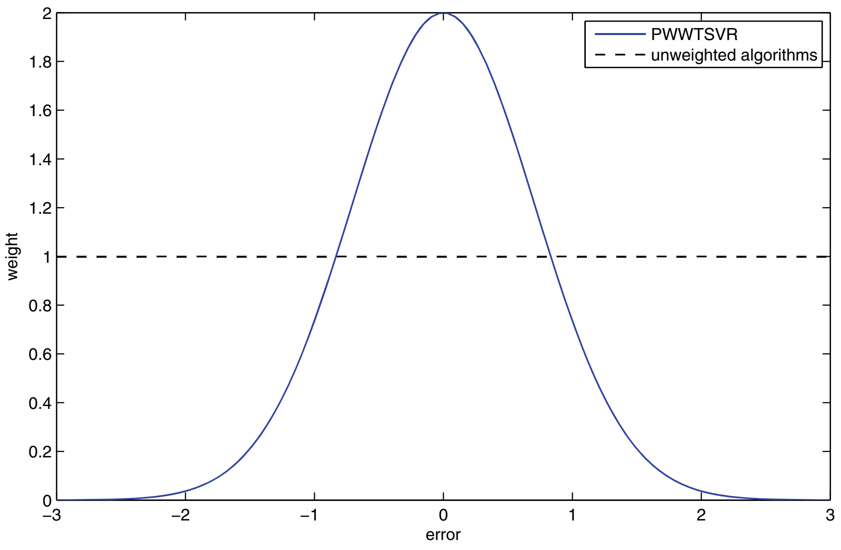


**Fig. 2** Weight values of PWWTSVR and unweighted algorithms over error of samples

impulse response $\phi(t)$, resulting in the approximation coefficients $xa_{l+1}$, and a high-pass filter with impulse response $\psi(t)$, resulting in the detail coefficients $xd_{l+1}$,

$$xa_{l+1}(n) = \sum_k \phi(k-2n)xa_l(k) \tag{49}$$

$$xd_{l+1}(n) = \sum_k \psi(k-2n)xa_l(k) \tag{50}$$

The approximation coefficients $xa_{l+1}$ can be decomposed further to get $xa_{l+2}$ and $xd_{l+2}$.

(2). The obtained $l$ groups of decomposed sequences $(xd_1, xd_2, ..., xd_l, xa_l)$ after $l$ steps of decomposition are processed by an appropriate algorithm to remove noise. In this paper, the high frequency is set to zero directly as a denoising algorithm. Then, the denoised sequence $(xd_1', xd_2', ..., xd_l', xa_l')$ is obtained.

(3). In this stage, the estimated value of output $\hat{y}$ is reconstructed by the denoised sequence $(xd_1', xd_2', ..., xd_l', xa_l')$.

$$xa_{l-1}'(n) = \sum_k \phi(n-2k)xa_l'(k) + \sum_k \psi(n-2k)xd_l'(k) \tag{51}$$

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., $\hat{y} = xa_0'$. Substitute $\hat{Y}$ in (48), and the weighting matrix $D$ can be calculated. The process of signal decomposition and reconstruction is illustrated in Fig. 3.

## 3.5 Algorithm summary

In this subsection, the proposed PWWTSVR algorithm is illustrated.

**Algorithm 1** Projection wavelet weighted twin support vector regression.

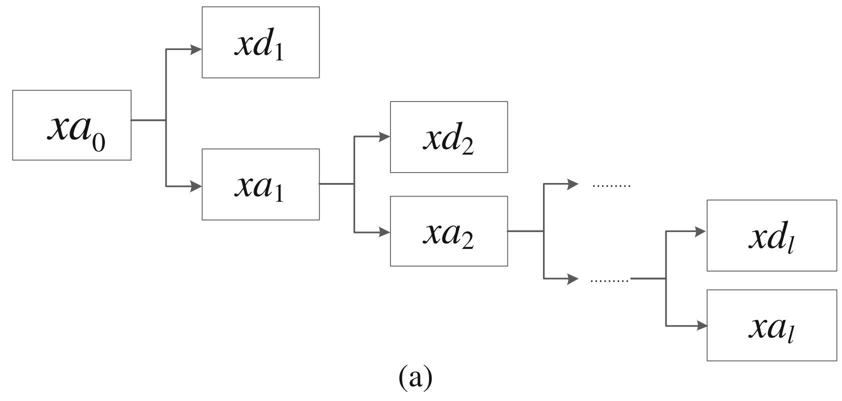
Input: The appropriate parameters $c_{11}, c_{12}, c_{13}, c_{21}, c_{22}, c_{23}, \varepsilon_1, \varepsilon_2$, the kernel parameters, Gaussian function parameters, and the training data matrix $A$.
Output: The regression function $f(x)$.
Process:

1. Calculate $D$ by (48), preprocess the data points by the empirical kernel technique for the nonlinear case, compute $\Sigma_x$ or $\Sigma_{\varphi(x)}$, $\Sigma_{xy}$ or $\Sigma_{\varphi(x)y}$.
2. Start with any $u_1^0$, and $u_2^0$. Using $u_1^i$, and $u_2^i$, compute $u_1^{i+1}$, and $u_2^{i+1}$ by (39) and (40) until $\left\|u_1^{i+1} - u_1^i\right\|$ and $\left\|u_2^{i+1} - u_2^i\right\|$ are less than some prescribed error.
3. Compute $f(x) = \frac{1}{2}(w_1 + w_2)^T x + \frac{1}{2}(b_1 + b_2)$.

The algorithm can be summarized as in Fig. 4.

## 4 Experimental results

In this section, some experiments are conducted to examine the performance of PWWTSVR, which is compared to TSVR [4], $\varepsilon$-TSVR [13], TPSVR [15], $\nu$-TWSVR [14], Asy-$\nu$-TSVR [12], KNNUPWTSVR [21], and WL-$\varepsilon$-TSVR [23], using four artificial datasets and eight benchmark datasets. The computer programs for simulation are implemented in a MATLAB R2014a environment on a PC



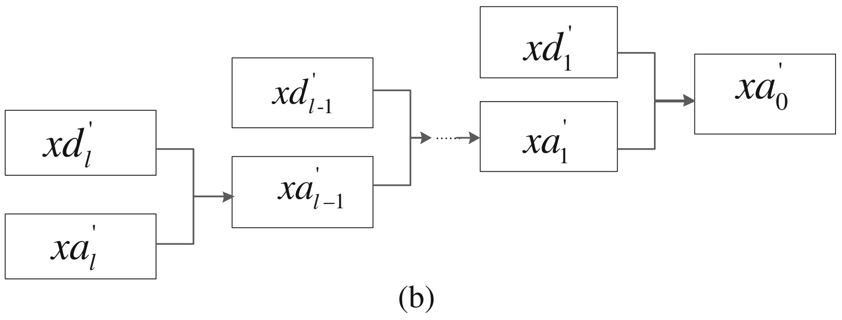


**Fig. 3** The process of decomposition and reconstruction of the time series by wavelet. **a** Decomposition process, **b** Reconstruction process

with an Intel Core i5 processor (3.3 Ghz) with 8 GB RAM. In this paper, a Gaussian nonlinear kernel is adopted for all datasets, i.e.,

$$K(a^T, b^T) = exp\left(-\frac{\|a-b\|^2}{e}\right), \tag{52}$$

where $a$ and $b$ are vectors, and $e$ 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 $\{10^i | i = -4, -3, ..., 5\}$. To degrade the computational complexity of parameter selection, let $c_{11} = c_{21} = c_1, c_{12} = c_{22} = c_2, c_{13} = c_{23} = c_3$ in PWWTSVR; $C_1 = C_2 = C, \varepsilon_1 = \varepsilon_2 = \varepsilon$ in TSVR; $c_1 = c_2, c_3 = c_4$ in $\varepsilon$-TSVR; $c_1 = c_2 = c, \nu_1 = \nu_2 = \nu, \lambda_1 = \lambda_2 = \lambda$ in TPSVR; $c_1 = c_3, c_2 = c_4, \nu_1 = \nu_2 = \nu$, in $\nu-$TWSVR; $C_1 = C_2 = C, \nu_1 = \nu_2 = \nu$ in Asy-$\nu$-TSVR; $c_1 = c_2$, $c_3 = c_4, \varepsilon_1 = \varepsilon_2 = \varepsilon$ in KNNUPWTSVR; and $c_1 = c_2 = c$, $\nu_1 = \nu_2 = \nu$, in WL-$\varepsilon$-TSVR.

The performance of PWWTSVR and the other seven methods is evaluated by selected criteria. The number of testing samples is denoted by $l$, $y_i$ denotes the real value of a testing sample point $x_i$, $\bar{y} = \sum_i \frac{1}{l} y_i$ is the mean value of $y_1, y_2, ..., y_l$, 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=$\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$.

SST: Sum squared deviation of testing samples, defined as SST=$\sum_{i=1}^{m}(y_i - \bar{y}_i)^2$.

SSR: Sum squared deviation that can be explained by the estimator, defined as SSR=$\sum_{i=1}^{m}(\hat{y}_i - \bar{y}_i)^2$.

SSE represents the fitting precision. Too small an SSE value may mean overfitting of the regressor due to the fitting of

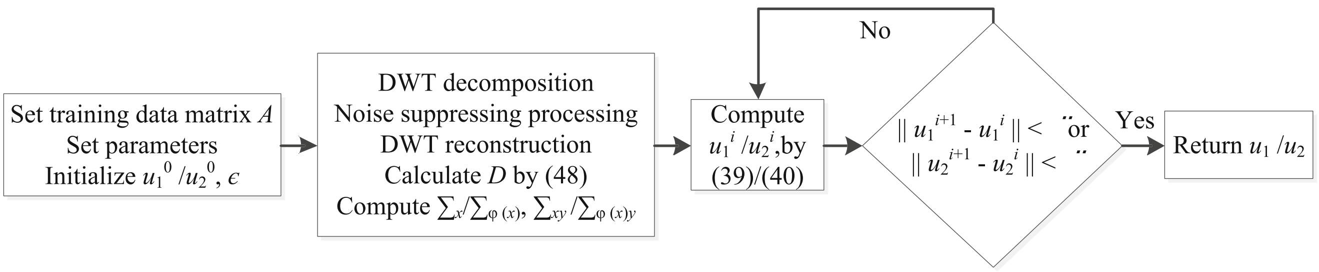

Set training data matrix $A$
Set parameters
Initialize $u_1^0 / u_2^0$, $\epsilon$
DWT decomposition
Noise suppressing processing
DWT reconstruction
Calculate $D$ by (48)
Compute $\sum_x / \sum_{\varphi(x)}$, $\sum_{xy} / \sum_{\varphi(x)y}$
Compute
$u_1^i / u_2^i$,by
(39)/(40)
$\| u_1^{i+1} - u_1^i \| <$ or
$\| u_2^{i+1} - u_2^i \| <$
No
Yes
Return $u_1 / u_2$

**Fig. 4** Summary of steps performed by PWWTSVR algorithm

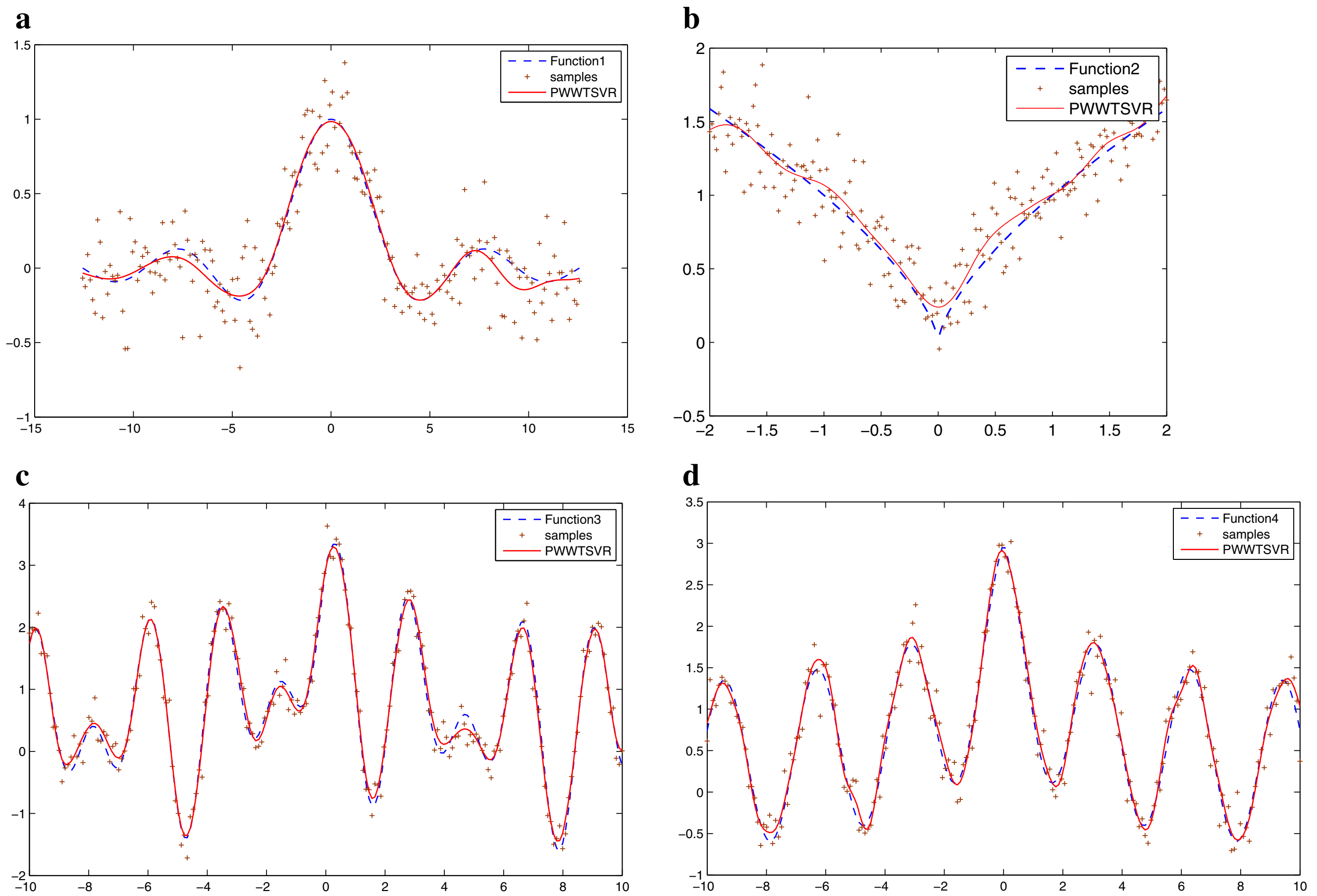


**Fig. 5** Regression performance of PWWTSVR for noise variance $\varsigma^2 = 0.2^2$ on **a** Function 1 dataset; **b** Function 2 dataset; **c** Function 3 dataset; **d** Function 4 dataset

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=$\sum_{i=1}^{m}(y_i - \hat{y}_i)^2/(y_i - \bar{y}_i)^2$.

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

In most cases, a small SSE/SST represents good agreement between estimates and real values, but too small a value also probably means overfitting of the regressor. SSR/SST shows the variance ratio of the estimated data over the sample data. SSR/SST=1 means the estimated data have the same variance as training samples.

**Table 1** Function definition and definition domain of artificial datasets

| Name | Function definition | Domain of definition |
|---|---|---|
| Function 1 | $f(x) = \frac{\sin x}{x}$ | $x \in [-4\pi, 4\pi]$ |
| Function 2 | $f(x) = x^{2/3}$ | $x \in [-2, 2]$ |
| Function 3 | $f(x) = \frac{4}{\lvert x\rvert+2} + \cos(2x) + \sin(3x)$ | $x \in [-10, 10]$ |
| Function 4 | $f(x) = \frac{4}{\lvert x\rvert+2} + \cos(2x)$ | $x \in [-10, 10]$ |

## 4.1 Experiments on artificial datasets

To demonstrate the performance of the proposed algorithm on artificial datasets, consider four noised functions for approximation, denoted as $y = f(x) + n$, where the noise $n\tilde{}N(0, \varsigma^2)$ is Gaussian additive noise with mean zero and variance $\varsigma^2$ being $0.1^2$ and $0.2^2$, respectively. Set the number of training points to 200. Db-3 wavelets used in PWWTSVR are selected to denoise the training data. We compare the performance of seven algorithms (TSVR, $\varepsilon$-TSVR, TPSVR, $\nu$-TWSVR, Asy-$\nu$-TSVR, KNNUPWTSVR, and WL-$\varepsilon$-TSVR) to that of PWWTSVR. Figure 5 shows the one-run approximation functions obtained by PWWTSVR for four artificial functions with different variance noise. The adopted functions for regression are defined in Table 1. The selection of parameters can be seen in Table 2.

**Table 2** Performance and selection of parameters of eight algorithms on artificial datasets with different variance of Gaussian noises

| Datasets | PWWTSVR | TSVR | $\varepsilon$-TSVR | TPTSVR | $\nu$-TWSVR | Asy-$\nu$-TSVR | KNNUPWTSVR | WL-$\varepsilon$-TSVR |
|---|---|---|---|---|---|---|---|---|
| | $c_1, c_2, c_3, E, \sigma^2$ | $C, \varepsilon$ | $c_1, c_3$ | $c, \nu, \lambda$ | $c_1, c_2, \nu$ | $C, \nu, p$ | $c_1, c_3, \varepsilon$ | $c, \nu$ |
| | SSE±STD | SSE±STD | SSE±STD | SSE±STD | SSE±STD | SSE±STD | SSE±STD | SSE±STD |
| | SSE/SST±STD | SSE/SST±STD | SSE/SST±STD | SSE/SST±STD | SSE/SST±STD | SSE/SST±STD | SSE/SST±STD | SSE/SST±STD |
| | SSR/SST±STD | SSR/SST±STD | SSR/SST±STD | SSR/SST±STD | SSR/SST±STD | SSR/SST±STD | SSR/SST±STD | SSR/SST±STD |
| | CPU sec.±STD | CPU sec.±STD | CPU sec.±STD | CPU sec.±STD | CPU sec.±STD | CPU sec.±STD | CPU sec.±STD | CPU sec.±STD |
| Function 1 | 1,1,0.1,0.1,1 | 0.001,0.1 | 0.001,10 | 1000,100,100 | 10,1,10 | 10,0.1,0.5 | 1,0.01,1 | 1,1 |
| $N(0, 0.1^2)$ | **0.1458±0.0356** | 0.3046±0.0752 | 0.1466±0.0361 | 0.3893±0.0951 | 0.1462±0.0360 | 0.3057±0.0764 | 0.2466±0.0636 | 0.1831±0.0445 |
| | **0.0086±0.0021** | 0.0179±0.0044 | **0.0086±0.0021** | 0.0229±0.0056 | **0.0086±0.0021** | 0.0180±0.0045 | 0.0145±0.0037 | 0.0108±0.0026 |
| | 0.9328± 0.0186 | 1.0121±0.0189 | 0.9299±0.0185 | 1.0100±0.0265 | 0.9310±0.0183 | 1.0106±0.0183 | 1.0094±0.0184 | **0.9981±0.0190** |
| | **0.0301±0.0103** | 0.0863±0.0582 | 0.0880±0.0190 | 0.1483±0.0167 | 0.0831±0.0126 | 0.1184±0.0048 | 0.0428±0.0152 | 0.0586±0.0038 |
| Function 1 | 1,1,0.1,0.1,10 | 0.001,0.1 | 0.1,10 | 1000,100,100 | 10,10,10 | 10,0.1,0.5 | 1,0.01,1 | 1,1 |
| $N(0, 0.2^2)$ | **0.5297±0.1706** | 1.2478±0.2333 | 0.5395±0.1795 | 1.4612±0.3537 | 0.5427±0.1847 | 1.2557±0.2269 | 1.0614±0.2352 | 0.8026±0.2198 |
| | **0.0215±0.0069** | 0.0506±0.0095 | 0.0219±0.0073 | 0.0592±0.0143 | 0.0220±0.0075 | 0.0509±0.0092 | 0.0430±0.0095 | 0.0325±0.0089 |
| | 0.9626±0.0943 | 1.0522±0.1127 | **0.9699±0.0966** | 1.0629±0.1257 | 0.9665±0.0965 | 1.0531±0.1129 | 1.0505±0.1087 | 1.0340±0.1018 |
| | **0.0266±0.0032** | 0.0759±0.0081 | 0.0889±0.0024 | 0.1571±0.0095 | 0.0920±0.0034 | 0.1115±0.0033 | 0.0396±0.0068 | 0.0627±0.0070 |
| Function 2 | 0.01,0.001,1,1,0.1 | 0.01,0.1 | 0.1,0.01 | 1000,100,100 | 0.01,100,1 | 100,0.1,0.5 | 10,0.1,10 | 1000,0.001 |
| $N(0, 0.1^2)$ | **0.1731±0.0391** | 0.1857±0.0484 | 0.1740±0.0384 | 0.2889±0.0809 | 0.1903±0.0638 | 0.2095±0.0662 | 0.2277±0.0415 | 0.1849±0.0485 |
| | **0.0055±0.0012** | 0.0059±0.0015 | **0.0055±0.0012** | 0.0091±0.0026 | 0.0060±0.0020 | 0.0066±0.0021 | 0.0072±0.0013 | 0.0058±0.0015 |
| | 1.0078±0.0265 | 1.0163±0.0274 | 1.0169±0.0274 | 1.0202±0.0444 | 1.0178±0.0306 | 1.0189±0.0359 | **1.0070±0.0271** | 1.0160±0.0275 |
| | **0.0302±0.0031** | 0.0664±0.0034 | 0.0629±0.0069 | 0.1554±0.0075 | 0.1177±0.0061 | 0.1210±0.0095 | 0.0331±0.0026 | 0.0584±0.0031 |
| Function 2 | 0.01,0.001,1,1,1 | 0.01,0.1 | 0.1,0.01 | 1000,100,100 | 0.01,100,1 | 100,0.1,0.5 | 10,0.1,10 | 1000,0.001 |
| $N(0, 0.2^2)$ | **0.5500±0.1776** | 0.6102±0.1810 | 0.5509±0.1714 | 0.7574±0.2187 | 0.6298±0.1913 | 0.6352±0.1951 | 0.5604±0.1830 | 0.6096±0.1815 |
| | 0.0194±0.0063 | 0.0215±0.0064 | **0.0194±0.0060** | 0.0267±0.0077 | 0.0222±0.0067 | 0.0224±0.0069 | 0.0197±0.0064 | 0.0215±0.0064 |
| | 1.0248±0.0666 | 1.0345±0.0671 | 1.0319±0.0684 | **0.9916±0.1009** | 1.0311±0.0718 | 1.0263±0.0673 | 1.0266±0.0691 | 1.0340±0.0673 |
| | **0.0299±0.0013** | 0.0675±0.0044 | 0.0595±0.0034 | 0.1670±0.0088 | 0.1248±0.0161 | 0.1312±0.0057 | 0.0321±0.0009 | 0.0560±0.0021 |
| Function 3 | 0.1,0.01,10,1,1 | 0.01,0.1 | 0.1,0.01 | 1000,100,100 | 0.01,100,1 | 100,0.1,0.5 | 0.1,0.1,10 | 1000.0.01 |
| $N(0, 0.1^2)$ | **0.4414±0.0738** | 0.5807±0.0650 | 0.5090±0.0515 | 0.6644±0.0709 | 0.6044±0.0717 | 0.6807±0.0893 | 0.4437±0.0575 | 0.5811±0.0655 |
| | 0.0019±0.0003 | 0.0024±0.0003 | 0.0021±0.0002 | 0.0028±0.0003 | 0.0025±0.0003 | 0.0029±0.0004 | **0.0019±0.0002** | 0.0024±0.0003 |
| | 0.9829±0.0157 | 1.0011±0.0169 | **1.0002±0.0164** | 0.9900±0.0163 | 1.0028±0.0164 | 1.0022±0.0165 | 0.9945±0.0157 | 1.0012±0.0169 |
| | 0.0457±0.0021 | 0.0662±0.0027 | 0.0561±0.0022 | 0.1312±0.0077 | 0.1316±0.0530 | 0.1069±0.0051 | **0.0328±0.0007** | 0.0588±0.0098 |
| Function 3 | 1,0.01,100,1,1 | 0.01,0.1 | 0.1,0.01 | 1000,100,100 | 0.01,100,1 | 100,0.1,0.5 | 0.001,0.1,10 | 1000,0.01 |
| $N(0, 0.2^2)$ | **1.6688±0.3877** | 2.6141±0.3280 | 2.2318±0.2865 | 2.5502±0.5591 | 2.4431±0.3310 | 2.8262±0.3788 | 1.7578±0.3113 | 2.2318±0.2865 |
| | **0.0086±0.0020** | 0.0135±0.0017 | 0.0116±0.0015 | 0.0132±0.0029 | 0.0127±0.0017 | 0.0146±0.0020 | 0.0091±0.0016 | 0.0116±0.0015 |
| | 0.9165±0.0246 | 0.9940±0.0313 | 0.9912±0.0297 | 0.9738±0.0304 | 0.9908±0.0337 | **0.9940±0.0301** | 0.9802±0.0281 | 0.9912±0.0297 |
| | 0.0534±0.0066 | 0.0733±0.0061 | 0.0590±0.0042 | 0.1496±0.0388 | 0.1270±0.0090 | 0.1149±0.0052 | **0.0236±0.0014** | 0.0578±0.0016 |

**Table 2** (continued)

| Datasets | PWWTSVR | TSVR | $\varepsilon$-TSVR | TPTSVR | $\nu$-TWSVR | Asy-$\nu$-TSVR | KNNUPWTSVR | WL-$\varepsilon$-TSVR |
|---|---|---|---|---|---|---|---|---|
| Function 4 | 1,0.01,100,10,0.1 | 0.01,0.1 | 0.1,0.01 | 1000,100,100 | 0.01,100,1 | 100,0.1,0.5 | 0.0010 0.1000 10 | 1000,0.01 |
| $N(0, 0.1^2)$ | **0.4716±0.1442** | 0.7097±0.1401 | 0.6204±0.1381 | 0.6929±0.1738 | 0.6762±0.1732 | 0.7922±0.1809 | 0.5609±0.1444 | 0.6204±0.1381 |
| | **0.0043±0.0013** | 0.0064±0.0013 | 0.0056±0.0013 | 0.0063±0.0016 | 0.0061±0.0016 | 0.0072±0.0016 | 0.0051±0.0013 | 0.0056±0.0013 |
| | 0.9954±0.0225 | 1.0045±0.0224 | 1.0024±0.0223 | 0.9998±0.0175 | 1.0052±0.0227 | 1.0072±0.0213 | **1.0023±0.0230** | 1.0024±0.0223 |
| | 0.0539±0.0068 | 0.0657±0.0025 | 0.0558±0.0020 | 0.1460±0.0456 | 0.1117±0.0070 | 0.1110±0.0084 | **0.0233±0.0005** | 0.0579±0.0046 |
| Function 4 | 1,0.01,100,10,0.1 | 0.01,0.1 | 0.1,0.01 | 1000,100,100 | 0.01,100,1 | 100,0.1,0.5 | 0.001,0.1,10 | 1000,0.01 |
| $N(0, 0.2^2)$ | **1.8274±0.4318** | 2.6136±0.4249 | 2.3451±0.4143 | 2.5709±0.4456 | 2.5466±0.4436 | 2.8086±0.4496 | 2.0370±0.4263 | 2.3451±0.4143 |
| | **0.0115±0.0027** | 0.0164±0.0027 | 0.0147±0.0026 | 0.0161±0.0028 | 0.0160±0.0028 | 0.0176±0.0028 | 0.0128±0.0027 | 0.0147±0.0026 |
| | 0.9849±0.0426 | 1.0076±0.0490 | **1.0028±0.0477** | 0.9929±0.0593 | 1.0056±0.0486 | 1.0104±0.0542 | 0.9945±0.0446 | 1.0028±0.0477 |
| | 0.0506±0.0008 | 0.0714±0.0077 | 0.0578±0.0019 | 0.1496±0.0352 | 0.1235±0.0080 | 0.1185±0.0086 | **0.0235±0.0007** | 0.0580±0.0021 |

The number of training points and testing points were all set to 200. Testing data were selected randomly under a uniform distribution and assumed to be noise-free. Other settings can be seen in Table 2.

The average results are summarized in Table 2. In the value items, the first item shows the mean value of 10-times testing results and the second represents plus or minus the standard deviation. The values with the lowest SSE, SSE/SST, and CPU time, and the closest SSR/SST to 1 are typeset in bold. Obviously, PWWTSVR obtains the lowest testing errors, and it has the smallest SSE and SSE/SST values among these eight cases. For the value of SSR/SST, PWWTSVR gets a value near to 1 and is close to that of other methods. For calculation time, PWWTSVR gets the smallest value for function 1 and function 2, and the second smallest value for function 3 and function 4. It is slightly higher than that of KNNUPWTSVR, which is also solved in primal space rather than dual space. From the comparisons of the eight algorithms, it can be seen that PWWTSVR has better performance in most criteria.

To further verify the validity of the proposed algorithm, an iterative regression experiment was carried out. The procedure was as follows.

Step 1: Calculate regression function $f(x)$ using Algorithm 1.

Step 2: Replace the $\hat{Y}$ in (48) by $f(x)$ and renew the weighting parameter $D$ by (48).

Step 3: Repeat steps 1 and 2 several times.

Figure 6 shows a one-shot SSE value over iteration times for the regression of function 1. The training number and test number are both set to 500, the variance of noise is 0.2, and other parameters can be seen in Table 2. The STD in Table 3 means the standard deviation of the performance. It shows that the SSE value decreases as the iterations increase, which can verify the effectiveness of adding the weighting parameter $D$. The quality of pre-regression can determine the regression precision of the proposed method; that is why the iterative process can play a positive role.

To test the sensitivity of parameter selection on PWWTSVR regression performance, we studied the influence of parameters $c_1, c_2, c_3, E, \sigma^2$ on SSE, SSE/SST, and SSR/SST for artificial datasets on function 1 and function 2 with noise variance $\sigma^2 = 0.1^2$. From Fig. 7, we can see that most SSE and SSE/SST curves show convexity, i.e., with the increase of a parameter, the performance value decreases first and then increases, which can ensure that the optimal value can be found. However, their sensitivities to the change of parameter values are different. The performance is sensitive to $c_1, c_2, E, \sigma^2$, but not to $c_3$.

The selection of a kernel function is important to the performance of regression algorithms. To test the effect of kernel functions, we studied the performance of six kernel

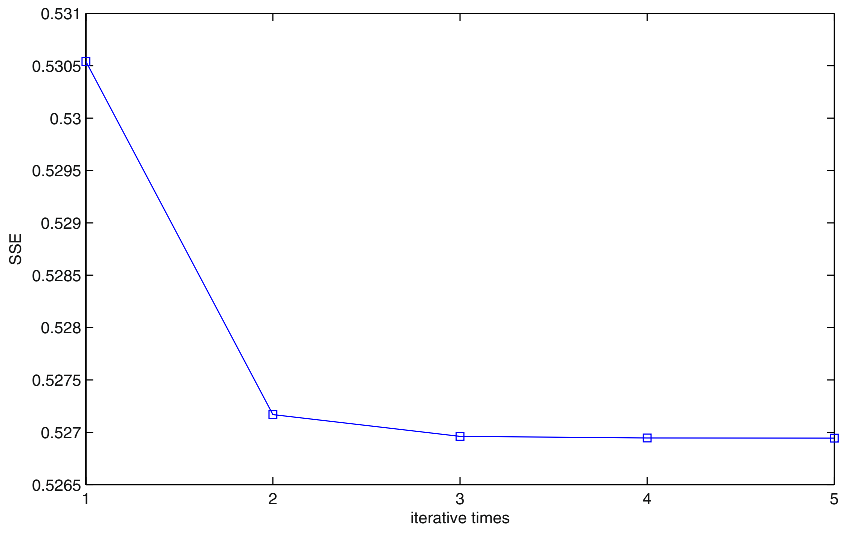


**Fig. 6** The influence of iterative times on SSE value for Function 1 dataset

functions for artificial datasets on function 1, including 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. For linear kernel or polynomial kernel to have high SSE values means that it is not suitable for regression in this kind of problem.

## 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 [29] were conducted. Due to the properties of wavelet transforms, the selected datasets are all time series. Although there are many regression applications of time-series signals, few publicly available datasets are available to use. Information on the selected 14 datasets is summarized in Table 4, including the adopted number of training samples and attributes. Note that the Istanbul stock dataset is used as four datasets by adopting 1/2/4/8 attributes as inputs of algorithms. In addition, one dataset, Concrete, which is not a time series, is used for performance comparison.

| | |
|---|---|
| **Energy** | Energy used in a low-energy building. |
| **Beijing2.5** | PM2.5 data of U.S. Embassy in Beijing. |
| **AirQuality** | Gas concentrations on the field in an Italian city. |
| **DowJones** | Weekly data for the Dow Jones Industrial Index. |
| **Istanbul** | Returns of the Istanbul Stock Exchange with seven other international indices. |
| **PowerConsume** | Measurements of electric power consumption in one household. |
| **GHG** | Emissions of greenhouse gas in California. |
| **Electricity** | Electricity consumption of 370 points/clients. |
| **DailyDemand** | Demand data collected from a large Brazilian logistics company. |
| **SML** | Indoor environmental data in a house, including temperature, humidity, and carbon dioxide in ppm. |
| **Concrete** | Concrete compressive strength, which is a highly nonlinear function of age and ingredients (not a time-series dataset). |

For all the real-world examples considered in this paper, the original data are normalized as: $\tilde{s}_i = (s_i - s_{\min})/(s_{\max} - s_{\min})$, where $s_i$ is the input value of the

**Table 3** Comparisons of PWWTSVR on Function 1 with different kernel functions

| Kernel name | Function definition | SSE | SSE/SST | SSR/SST |
|---|---|---|---|---|
| Gaussian | $\exp(-\|x\text{-}x_i\|^2/e)$ | **0.1458±0.0356** | **0.0086±0.0021** | **0.9328± 0.0186** |
| Linear | $xx_i^T$ | 21.171±0.0963 | 1.0087±0.0045 | 0.0091±0.0043 |
| Polynomial | $(axx_i^T+c)^d$ | 13.602±0.0855 | 0.6825± 0.0042 | 0.3056±0.0218 |
| Sigmoid | $\tanh(axx_i^T+c)$ | 0.4815±0.0682 | 0.0311±0.0044 | 0.9171±0.0340 |
| Multiquadric | $(\|x\text{-}x_i\|^2+\sigma^2)^{0.5}$ | 0.2754±0.0747 | 0.0121±0.0032 | 0.9154±0.0313 |
| Log | $-\log(1+\|x\text{-}x_i\|^{\sigma})$ | 0.2146±0.0649 | 0.0121±0.0037 | 0.9282±0.0367 |

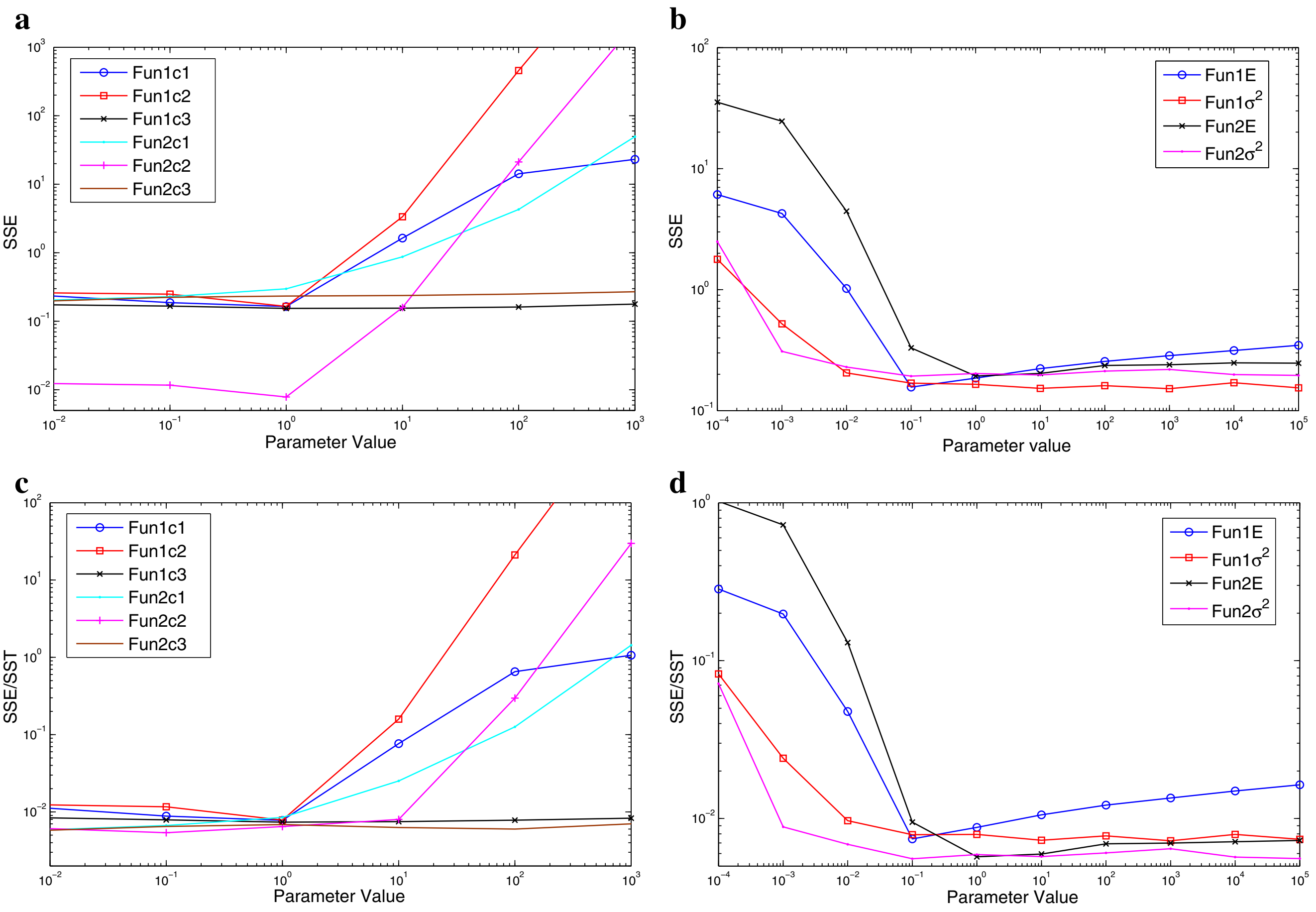


**Fig. 7** Influence of parameter $c_1, c_2, c_3, E, \sigma^2$ on SSE, SSE/SST, and SSR/SST for artificial datasets Function 1 and Function 2 with noise variance $\sigma^2 = 0.1^2$. The curve of SSE over $c_1$ for Function 1 is marked as Fun1c1, for example

$i$th sample, $\tilde{s}_i$ is its corresponding normalized value, and $s_{\min}$ and $s_{\max}$ denote the minimum and maximum values, respectively. The parameters are shown in Table 5, which also show the performance of the eight algorithms for the benchmark datasets. The odd-numbered examples are selected as training data, and even-numbered examples as test data, so the amounts of training data and test data are the same. The numbers ($m$) and dimensions ($n$) of samples are listed in Table 5. So as not to destroy the character of the time series of the data, one-shot experiments rather than cross-validation experiments are adopted. The values with the lowest SSE, SSE/SST, and CPU time, and the closest SSR/SST to 1 are typeset in bold.

**Table 4** Benchmark datasets and their number of samples and attributes

| Datasets | Samples($m$) | Attributes($n$) |
|---|---|---|
| Energy | 500 | 22 |
| Beijing2.5 | 500 | 4 |
| AirQuality | 411 | 9 |
| DowJones | 360 | 4 |
| Istanbul | 268 | 1/2/4/8 |
| PowerConsume | 298 | 4 |
| GHG | 162 | 18 |
| Electricitye | 250 | 9 |
| DailyDemand | 30 | 4 |
| SML | 212 | 8 |
| Concrete | 500 | 8 |

From Table 5, it can be seen that PWWTSVR outperforms the other algorithms. On most datasets, PWWTSVR obtains the smallest or the second-smallest SSE and SSE/SST value. Only on PowerConsume, GHG, Electricity, and Concrete does PWWTSVR obtain poor values. This is mainly because these datasets have too much noise, or the regularity of change with time is not strong, which will result in weakening or failure of the wavelet-filtering effect. In particular, the contrast dataset Concrete is not a time series, which leads to very poor performance for PWWTSVR. This also illustrates the scope of application of our algorithm. For SSR/SST criteria, PWWTSVR achieves nine of the first or second values which is close to 1 among 14 datasets. It means that the proposed algorithm has nearly the same variance as the

**Table 5** Performance and selection of parameters of eight algorithms on benchmark datasets

| Datasets | PWWTSVR | TSVR | $\varepsilon$-TSVR | TPTSVR | $\nu$-TWSVR | Asy-$\nu$-TSVR | KNNUPWTSVR | WL-$\varepsilon$-TSVR |
|---|---|---|---|---|---|---|---|---|
| (Data Size m*n) | $c_1, c_2, c_3, E, \sigma^2$ | $C, \varepsilon$ | $c_1, c_3$ | $c, \nu, \lambda$ | $c_1, c_2, \nu$ | $C, \nu, p$ | $c_1, c_3, \varepsilon$ | $c, \nu$ |
|  | SSE | SSE | SSE | SSE | SSE | SSE | SSE | SSE |
|  | SSE/SST | SSE/SST | SSE/SST | SSE/SST | SSE/SST | SSE/SST | SSE/SST | SSE/SST |
|  | SSR/SST | SSR/SST | SSR/SST | SSR/SST | SSR/SST | SSR/SST | SSR/SST | SSR/SST |
|  | CPU sec. | CPU sec. | CPU sec. | CPU sec. | CPU sec. | CPU sec. | CPU sec. | CPU sec. |
| Energy | 0.01,10,1,10,0.1 | 0.01,10 | 0.1,0.01 | 1000,100,10000 | 0.01,100,0.01 | 0.01,0.01,0.5 | 10,0.01,10 | 10,0.01 |
| (500*22) | **3.6767** | 3.8750 | 3.7077 | 3.6904 | 3.7306 | 3.8628 | 3.7101 | 3.7077 |
|  | **0.4854** | 0.5116 | 0.4895 | 0.4872 | 0.4925 | 0.5099 | 0.4898 | 0.4895 |
|  | 1.1517 | 1.2348 | 1.1190 | 1.1523 | 1.1420 | 1.2280 | **1.0769** | 1.1190 |
|  | **0.1444** | 0.2385 | 0.3769 | 0.8941 | 0.7838 | 0.4515 | 0.9863 | 0.2912 |
| Beijing2.5 | 0.01,10,10,100,1 | 0.01,10 | 0.1,0.01 | 1000,100,10000 | 0.0001,1,0.1 | 0.01,0.01,0.5 | 0.0001,1,10 | 1,0.0001 |
| (500*4) | **6.1700** | 6.3487 | 6.5399 | 6.4861 | 6.1765 | 6.3433 | 6.5809 | 6.1935 |
|  | **0.5503** | 0.5663 | 0.5833 | 0.5786 | 0.5509 | 0.5658 | 0.5870 | 0.5524 |
|  | **0.3922** | 0.3706 | 0.3607 | 0.3672 | 0.3863 | 0.3688 | 0.3600 | 0.3898 |
|  | 0.2177 | 0.3421 | 0.3868 | 1.4580 | 0.8646 | 0.3977 | **0.1349** | 0.2642 |
| AirQuality | 0.01,100,100,10,0.1 | 0.001,0.1 | 0.1,0.01 | 1000,100,100000 | 0.001,1,0.1 | 0.001,0.1,0.4 | 1,0.0001,10 | 1.0000,0.001 |
| (411*9) | **1.0239** | 1.0703 | 1.1934 | 1.2625 | 1.0708 | 1.0705 | 1.0627 | 1.0705 |
|  | **0.0636** | 0.0665 | 0.0742 | 0.0785 | 0.0666 | 0.0665 | 0.0660 | 0.0665 |
|  | 0.8593 | 0.8314 | 0.8082 | 0.8546 | 0.8316 | 0.8313 | **0.8817** | 0.8313 |
|  | **0.1645** | 0.2774 | 0.2078 | 0.9048 | 0.5012 | 0.1769 | 0.5086 | 0.1867 |
| DowJones | 0.1,100,10,10.0000,0.1 | 0.1,10 | 0.1,0.01 | 1000,100,100000 | 0.001,100,0.01 | 0.1,0.01,0.4 | 10,0.01,10 | 100,0.01 |
| (360*4) | **0.0386** | 0.0509 | 0.0501 | 0.0971 | 0.0488 | 0.0488 | 0.0492 | 0.0501 |
|  | **0.0026** | 0.0034 | 0.0033 | 0.0065 | 0.0033 | 0.0032 | 0.0033 | 0.0033 |
|  | **0.9897** | 0.9452 | 0.9399 | 0.9257 | 0.9423 | 0.9407 | 0.9377 | 0.9399 |
|  | **0.0537** | 0.2809 | 0.1759 | 0.5272 | 0.2616 | 0.4592 | 0.2179 | 0.2168 |
| Istanbul1 | 1,100,100,10,0.1 | 0.01,10 | 0.1,0.01 | 1000,100,100000 | 0.1,100,0.001 | 0.01,0.001,0.5 | 1000,0.01,10 | 100,0.01 |
| (268*1) | 5.5213 | 6.1766 | 5.7984 | 6.4490 | **5.5018** | 6.1760 | 5.7806 | 5.7984 |
|  | 1.3602 | 1.5216 | 1.4284 | 1.5887 | **1.3554** | 1.5214 | 1.4240 | 1.4284 |
|  | **0.9827** | 1.5286 | 1.4017 | 1.5935 | 1.1387 | 1.5294 | 1.3200 | 1.4017 |
|  | **0.0300** | 0.1755 | 0.0934 | 0.5047 | 0.2865 | 0.1164 | 0.2168 | 0.1580 |
| Istanbul2 | 1,1,10,10,0.1 | 0.01,10 | 0.1,0.01 | 1000,100,100000 | 0.1,10,0.1 | 0.01,0.1,0.6 | 100,0.01,10 | 10,0.1 |
| (268*2) | **5.2382** | 5.4127 | 5.4153 | 5.8178 | 5.2832 | 5.4099 | 5.6069 | 5.2745 |
|  | **1.2904** | 1.3334 | 1.3340 | 1.4332 | 1.3015 | 1.3327 | 1.3813 | 1.2994 |
|  | **1.1122** | 1.4831 | 1.4667 | 1.5813 | 1.3316 | 1.4820 | 1.4223 | 1.3205 |
|  | **0.0348** | 0.1164 | 0.0904 | 0.4923 | 0.2979 | 0.1151 | 0.2909 | 0.0937 |

**Table 5** (continued)

| Datasets | PWWTSVR | TSVR | $\varepsilon$-TSVR | TPTSVR | $\nu$-TWSVR | Asy-$\nu$-TSVR | KNNUPWTSVR | WL-$\varepsilon$-TSVR |
|---|---|---|---|---|---|---|---|---|
| (Data Size m*n) | $c_1, c_2, c_3, E, \sigma^2$ | $C, \varepsilon$ | $c_1, c_3$ | $c, \nu, \lambda$ | $c_1, c_2, \nu$ | $C, \nu, p$ | $c_1, c_3, \varepsilon$ | $c, \nu$ |
| | SSE | SSE | SSE | SSE | SSE | SSE | SSE | SSE |
| | SSE/SST | SSE/SST | SSE/SST | SSE/SST | SSE/SST | SSE/SST | SSE/SST | SSE/SST |
| | SSR/SST | SSR/SST | SSR/SST | SSR/SST | SSR/SST | SSR/SST | SSR/SST | SSR/SST |
| | CPU sec. | CPU sec. | CPU sec. | CPU sec. | CPU sec. | CPU sec. | CPU sec. | CPU sec. |
| Istanbul4 | 1,10,100,10,0.1 | 0.01,10 | 0.1,0.01 | 1000,100,100000 | 0.1,10,0.1 | 0.01,0.1,0.5 | 100,0.01,10 | 10,0.1 |
| (268*4) | 4.6503 | 4.6725 | 4.7778 | **4.6435** | 4.7190 | 4.6687 | 5.1710 | 4.7152 |
| | 1.1456 | 1.1511 | 1.1770 | **1.1439** | 1.1625 | 1.1501 | 1.2739 | 1.1616 |
| | **1.3074** | 1.6666 | 1.6407 | 1.6739 | 1.5161 | 1.6664 | 1.6685 | 1.5043 |
| | **0.0385** | 0.1474 | 0.0830 | 0.4875 | 0.2933 | 0.1054 | 0.2141 | 0.0891 |
| Istanbul8 | 0.001,10,10000,1,0.01 | 0.01,10 | 0.1,0.01 | 1000,100,100000 | 0.1,1000,0.1 | 0.01,0.1,0.5 | 100,0.1,10 | 100,0.1 |
| (268*8) | **3.9904** | 4.0682 | 4.2395 | 4.1344 | 4.0270 | 4.0575 | 3.9942 | 4.3194 |
| | **0.9830** | 1.0022 | 1.0444 | 1.0185 | 0.9920 | 0.9996 | 0.9840 | 1.0641 |
| | **1.4573** | 2.0271 | 2.0009 | 2.0381 | 1.8629 | 2.0253 | 1.4690 | 1.8141 |
| | **0.0938** | 0.1637 | 0.1051 | 0.5855 | 0.2306 | 0.0984 | 0.2126 | 0.2580 |
| PowerConsume | 0.01,0.1,0.1,0.1,0.1 | 1,10 | 10,1 | 1000,100,1 | 0.01,100,1 | 1000,1,0.01 | 1,0.01,100 | 1,0.1 |
| (298*4) | 0.2966 | 0.3234 | 0.2868 | 0.3315 | 0.3399 | **0.2813** | 0.3478 | 0.3364 |
| | 0.0284 | 0.0274 | 0.0243 | 0.0281 | 0.0288 | **0.0238** | 0.0295 | 0.0285 |
| | **1.1236** | 1.2258 | 1.2737 | 1.2087 | 1.2148 | 1.2510 | 1.2152 | 1.1987 |
| | **0.0295** | 0.3098 | 0.2897 | 0.4959 | 0.3110 | 0.2292 | 0.1636 | 0.1386 |
| GHG | 0.001,10000,0.1,1000,1 | 1,100 | 1,0.0001 | 1000,100,10000 | 0.0001,100,1 | 10,10,0.1 | 100,0.001,1000 | 0.001,0.0001 |
| (162*18) | 6.9723 | 6.9010 | 6.7685 | 7.0160 | 6.8230 | 6.9789 | **6.6322** | 6.8572 |
| | 0.8986 | 0.8973 | 0.8801 | 0.9123 | 0.8872 | 0.9075 | **0.8624** | 0.8916 |
| | 0.2554 | 0.1262 | 0.1228 | 0.1211 | 0.1329 | 0.1614 | **0.2629** | 0.1245 |
| | **0.0296** | 0.0679 | 0.0958 | 0.1349 | 0.1238 | 0.0826 | 0.1452 | 0.0712 |
| Electricity | 0.1,10,1000,1000,1000 | 1,1 | 1,0.1 | 1000,100,0.1 | 0.01,10,0.001 | 10,0.01,0.01 | 0.001,0.0001,1 | 10,0.0001 |
| (250*9) | 3.0986 | **3.0985** | 3.1007 | 3.3627 | **3.0985** | 3.0994 | **3.0985** | 3.0985 |
| | 1.0168 | **1.0167** | 1.0175 | 1.1034 | **1.0167** | 1.0170 | **1.0167** | 1.0167 |
| | 0.0168 | 0.0167 | 0.0175 | **0.1034** | 0.0167 | 0.0170 | 0.0167 | 0.0167 |
| | 0.0434 | 0.1060 | 0.1303 | 0.1312 | 0.0952 | 0.0859 | **0.0318** | 0.0711 |
| DailyDemand | 100,1000,10000,10000,100 | 1,0.01 | 1,1 | 1000,100,1 | 0.001,100,1 | 1,0.01,0.1 | 1,1,1 | 100,0.001 |
| (30*4) | **1.0362** | 1.1107 | 1.2894 | 1.1224 | 1.1107 | 1.0490 | 1.1364 | 1.0489 |
| | **0.5693** | 0.6103 | 0.7085 | 0.6167 | 0.6102 | 0.5764 | 0.6244 | 0.5763 |
| | 0.3259 | 0.3289 | 0.2480 | 0.3027 | 0.3289 | 0.3337 | **0.3409** | 0.3325 |
| | 0.0032 | 0.0096 | 0.0095 | 0.0139 | 0.0113 | 0.0089 | **0.0016** | 0.0072 |

**Table 5** (continued)

| Datasets | PWWTSVR | TSVR | $\varepsilon$-TSVR | TPTSVR | $\nu$-TWSVR | Asy-$\nu$-TSVR | KNNUPWTSVR | WL-$\varepsilon$-TSVR |
|---|---|---|---|---|---|---|---|---|
| (Data Size m*n) | $c_1, c_2, c_3, E, \sigma^2$ | $C, \varepsilon$ | $c_1, c_3$ | $c, \nu, \lambda$ | $c_1, c_2, \nu$ | $C, \nu, p$ | $c_1, c_3, \varepsilon$ | $c, \nu$ |
| | SSE | SSE | SSE | SSE | SSE | SSE | SSE | SSE |
| | SSE/SST | SSE/SST | SSE/SST | SSE/SST | SSE/SST | SSE/SST | SSE/SST | SSE/SST |
| | SSR/SST | SSR/SST | SSR/SST | SSR/SST | SSR/SST | SSR/SST | SSR/SST | SSR/SST |
| | CPU sec. | CPU sec. | CPU sec. | CPU sec. | CPU sec. | CPU sec. | CPU sec. | CPU sec. |
| SML | 0.01,10000,0.01,1000,0.1 | 0.1,0.1 | 100,0.001 | 1000,100,10000 | 0.001,100,0.01 | 1000,0.01,0.001 | 10000,10,1 | 100,0.0001 |
| (212*8) | **3.6762** | 4.6467 | 4.7980 | 3.7499 | 4.7910 | 4.6279 | 5.2826 | 3.8864 |
| | **0.4183** | 0.5288 | 0.5460 | 0.4267 | 0.5452 | 0.5266 | 0.6012 | 0.4423 |
| | 1.4863 | 1.3319 | 1.4300 | 1.2247 | 1.3275 | 1.3921 | **1.1501** | 1.3886 |
| | **0.0118** | 0.1631 | 0.1454 | 0.1799 | 0.1299 | 0.1414 | 0.0777 | 0.1118 |
| Concrete | 0.01,1,0.1,0.1,0.01 | 0.01,1 | 0.01,0.1 | 100,100,10000 | 0.1,0.1,0.1 | 0.1,0.1,0.5 | 10,0.01,1000 | 1,0.1 |
| (500*8) | 8.5898 | **7.3894** | 7.4087 | 8.4739 | 7.4087 | 7.4087 | 11.9396 | 7.4087 |
| | 0.4525 | **0.3859** | 0.3869 | 0.4426 | 0.3869 | 0.3869 | 0.6236 | 0.3869 |
| | 0.4074 | 0.7586 | 0.7590 | 0.9369 | 0.7590 | 0.7590 | **0.9724** | 0.7590 |
| | **0.1020** | 0.4094 | 0.4619 | 2.4266 | 0.8005 | 0.7978 | 1.2716 | 2.0519 |

**Table 6** Algorithms' average rank for SSE, SSE/SST, and CPU time

| Datasets | PWWTSVR | TSVR | $\varepsilon$-TSVR | TPTSVR | $\nu$-TWSVR | Asy-$\nu$-TSVR | KNNUPWTSVR | WL-$\varepsilon$-TSVR |
|---|---|---|---|---|---|---|---|---|
| SSE | 1.8639 | 5.0909 | 4.5227 | 6.3182 | 4.2273 | 5.3864 | 4.6136 | 3.9773 |
| SSE/SST | 2.0862 | 5.1136 | 4.3864 | 6.2727 | 4.2955 | 5.3409 | 4.6136 | 3.9091 |
| CPU sec. | 1.3182 | 4.7273 | 3.9545 | 7.8636 | 6.1818 | 5.0000 | 3.4545 | 3.5000 |

original data. From the experimental results of Istanbul1, Istanbul2, Istanbul4, and Istanbul8 (Istanbul datasets with 1/2/4/8 attributes), it can be seen that SSE decreases as the number of input attributes increases, i.e., more input information leads to more accurate estimation. This shows the importance of collecting more information for learning accuracy. For the training time, due to the adoption of the Newton optimization approach in primal space, the PWWTSVR takes the minimum amount of CPU time except for Beijing2.5, Electricity, and DailyDemand. For these datasets, PWWTSVR uses the second shortest CPU time and KNNUPWTSVR the first, and its objective function is also solved in terms of the unconstrained primal problem.

## 4.3 Statistical analysis

Non-parametric statistical tests were carried out to validate the experimental performance results of the proposed algorithm. The Friedman test [16] was run to rank the algorithms over 22 datasets, including eight artificial datasets and 14 real datasets. The Friedman test had alpha=0.05 and was distributed according to chi-square with seven degrees of freedom. The Friedman statistic was 43.78, 39.74, and 98.85, with $p$ -values of 2.3595e-7, 1.4112e-6, and 1.8650e-18 for SSE, SSE/SST, and training time, respectively. The average ranks obtained by the eight methods for the criteria SSE, SSE/SST, and training time in the Friedman test are shown in Table 6, from which we can see that the average SSE, SSE/SST, and training time ranks of PWWTSVR are lower than those of other methods. This shows the superiority of the proposed algorithm, which achieves better generalization performance and shorter running times. To check the significant differences between the methods, the $p$-value was calculated through the Bonferroni-Dunn non-parametric test by computing multiple pairwise comparisons among the proposed algorithm and the other methods. The test assumes that the performances of two algorithms are significantly different if their average ranks differ by at least some critical value [16]. Table 7 lists the $p$-value of the Bonferroni-Dunn test on the criteria SSE, SSE/SST, and training time ranking results obtained by the Friedman procedure. The null hypothesis of these tests is that there is no difference between the results. From Table 7, we see that the $p$-value is very small, which means that PWWTSVR is quite different from other methods for SSE, SSE/SST, and training time, and this confirms its superiority.

From the above analysis, we can conclude that PWWTSVR can improve the performance of the model and reduce the computation time. The empirical risk term in the objective function is the sum of weighted squared distances from training points to the bound function. Minimizing it causes the function $f(x)$ to fit the training samples and avoiding under-fitting. The regularization term in the objective function is adopted to solve the overfitting problem. The structural risk minimization is implemented by minimizing the regularization term, and solving the problems in the primal space can reduce the computational costs. The introduction of the projection item and 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

**Table 7** $p$-value of Bonferroni-Dunn test for SSE, SSE/SST and CPU time comparing our PWWTSVR with other methods

| Algorithms | SSE | SSE/SST | CPU sec. |
|---|---|---|---|
| TSVR | $1.0606e^{-5}$ | $2.9385e^{-5}$ | $3.9134e^{-6}$ |
| $\varepsilon$-TSVR | $2.8451e^{-4}$ | $1.4706e^{-3}$ | $3.5745e^{-4}$ |
| TPTSVR | $1.2065e^{-9}$ | $8.0033e^{-9}$ | $7.8179e^{-19}$ |
| $\nu$-TWSVR | $1.2563e^{-3}$ | $2.2455e^{-3}$ | $4.5368e^{-11}$ |
| Asy-$\nu$-TSVR | $1.5270e^{-6}$ | $7.1227e^{-6}$ | $6.1896e^{-7}$ |
| KNNUPWTSVR | $1.7467e^{-4}$ | $4.7904e^{-4}$ | $3.8201e^{-3}$ |
| WL-$\varepsilon$-TSVR | $3.9191e^{-3}$ | $1.1549e^{-2}$ | $3.1349e^{-3}$ |

prior information of samples. It should be noted that wavelet theory is a powerful denoising tool for time-series signals. So, the proposed method is suitable for dealing with time-series datasets. If we deal with non-series samples with the proposed algorithm, performance may suffer. Additionally, the proposed model is suitable for small datasets, and a large number of training samples will incur a tremendous computational cost.

## 5 Conclusions

In this paper, a projection wavelet weighted twin support vector regression algorithm is proposed. The PWWTSVR calculates the regression function by the mean of up- and down-bound regression functions, which are solved by two small-sized optimization problems in primal space. Unlike the cases in $\varepsilon$-TSVR, each optimization problem of the proposed algorithm aims to seek a suitable projection axis such that the variance of projected points can be minimized. Moreover, samples in different positions in the proposed model are given different weights according to the distance between samples and preprocessed results by wavelet transform. Computational comparisons between PWWTSVR and several existing methods were performed on artificial and benchmark datasets. The experimental results show better generalization performance and demonstrate the effectiveness of the proposed method. The solution of the proposed objective functions based on the Newton iterative approach is deduced in this paper, and the experimental results manifest the advantage in calculation time. However, one can notice that a matrix-inversion process is involved in the algorithm, so it is not suitable for large-scale nonlinear problems. The selection of parameters can obviously affect the performance of PWWTSVR, so for future work, the study of optimal parameter selection and a method involving less calculation could be carried out.

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

## References

1. Vapnik VN (1995) The natural of statistical learning theroy. Springer, New York
2. Vapnik VN (1998) Statistical learning theroy. Wiley, New York
3. Khemchandani JR, Chandra S (2007) Twin support vector machines for pattern classification. IEEE Trans Pattern Anal 29(5):905–910
4. Peng X (2010) TSVR: an efficient twin support vector machine for regression. Neural Netw 23(3):356–372
5. 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
6. Suykens JAK, Vandewalle J (1999) Least squares support vector machine classifiers. Neural Process Lett 9(3):293–300
7. Scholkopf B, Smola AJ, Williamson RC, Bartlett PL (2000) New support vector algorithms. Neurocomputing 12(5):1207–1245
8. Huang XL, Shi L, Pelckmans K, Suykens JAK (2014) Asymmetric $\nu$-tube support vector regression. Comput Stat Data Anal 77:371–382
9. Huang XL, Shi L, Suykens JAK (2014) Support vector machine classifier with pinball loss. IEEE Trans Pattern Anal 36:984–997
10. 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
11. 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
12. Xu Y, Li X, Pan X, Yang Z (2017) Asymmetric $\nu$-twin support vector regression. Neural Comput Appl 2:1–16
13. Shao Y, Zhang C, Yang Z, Jing L, Deng N (2013) An $\nu$-twin support vector machine for regression. Neural Comput Appl 23:175–185
14. Rastogi R, Anand P, Chandra S (2017) A $\nu$-twin support vector machine based regression with automatic accuracy control. Appl Intell 46:670–683
15. Peng X, Xu D, Shen J (2014) A twin projection support vector machine for data regression. Neuro Comput 138:131–141
16. Melki G, Cano A, Kecman V, Ventura S (2017) Multi-target support vector regression via correlation regressor chains. Info Sci s415–416:53–69
17. Ding S, Wu F, Shi Z (2014) Wavelet twin support vector machine. Neural Comput Appl 25(6):1241–1247
18. Melki G, Cano A, Ventura S (2018) MIRSVM: multi-instance support vector machine with bag representatives. Pattern Recogn 79:228–241
19. Melki G, Kecman V, Ventura S, Cano A (2018) OLLAWV: OnLine learning algorithm using worst-violators. Appl Soft Comput 66:384–393
20. Xu Y, Wang L (2014) K-nearest neighbor-based weighted twin support vector regression. Appl Intell 41:299–309
21. Gupta D (2017) Training primal K-nearest neighbor based weighted twin support vector regression via unconstrained convex minimization. Appl Intell 47:962–991
22. Chapelle O (2007) Training a support vector machine in the primal. Neurocomputing 19(5):1155–1178
23. Ye Y, Bai L, Hua X, Shao Y, Wang Z, Deng N (2016) Weighted Lagrange $\nu$-twin support vector regression. Neurocomputing 197:53–68
24. Shevade S, Keerthi S, Bhattacharyya C (2000) Improvements to the SMO algorithm for SVM regression. IEEE Trans Neural Netw 11(5):1188–1193
25. 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
26. Peng X, Chen D (2018) PTSVRs: regression models via projection twin support vector machine. Info Sci 435:1–14
27. Horn RA, Johnson CR (2013) Matrix analysis, 2nd edn. Cambridge University Press, New York
28. Zhang F (2005) The Schur complement and its applications. Springer, New York
29. Blake C, Merz C (1998) UCI repository for machine learning databases. http://www.ics.uci.edu/mlearn/MLRepository.html

**Lidong Wang** was born in Liaoning province, China in 1971, received the B.S. and M.S. degrees from Harbin Institute of Technology and University of Science and Technology Liaoning, China in 1993 and 2004 respectively, and he received Ph.D. degree from Gyeongsang National University, Korea in 2008.

Since 1999 he has been with University of Science and Technology Liaoning, and he is currently an associate professor in Communication Engineering with University of Science and Technology Liaoning. He has authored over 30 research papers. His research interests include pattern recognition, machine learning and digital signal processing.

**Chuang Gao** was born in Liaoning Province, China, in 1982, received the B.S. and M.S. degrees from Warwick University and King's College London, UK in 2005 and 2007 respectively.

He is currently a Ph.D. candidate in University of Science and Technology Liaoning, China. He has authored over 10 research papers indexed by SCI and EI. His research interests include nonlinear system control, machine learning and intelligent control.

**Nannan Zhao** was born in Liaoning Province, China, in 1975, received the B.S. and M.S. degrees from Northeastern University and University of Science and Technology Liaoning, China in 1998 and 2003 respectively, and she received Ph.D. degree in intelligent control and expert system from Northeastern University in 2006.

She is currently an Associate Professor in the Communication Engineering with University of Science and Technology Liaoning. She has authored over 20 research papers. Her research interests include pattern recognition, machine learning and image processing.

**Xuebo Chen** Doctor of engineering, Professor. Graduated from University of Belgrade, Yugoslavia in 1994. Now he works in University of Science and Technology Liaoning. His research interests include theory of large scale system and control theory.