Training an Improved TSVR Based on Wavelet Transform Weight Via Unconstrained Convex Minimization

Nannan Zhao, Xinyu Ouyang, Chuang Gao and Zihui Zang

***Abstract*—An improved wavelet transform based weighted *ε*-twin support vector regression (WW-*ε*-TSVR) is proposed in this paper. In our WW-*ε*-TSVR, to reduce the impact of outliers, the wavelet weight matrix is introduced to give different penalties for the samples located in different places. Further, by using the 'plus' function, a pair of unconstrained minimization problems is solved in primal space rather than dual space, in which three smooth functions are introduced to replace the non-differentiable non-smooth 'plus' function. To speed up the training procedure, the generalized derivative iterative approach and Newton iterative approach are used to obtain the approximate solution, and five more detailed iterative algorithms are given. At last, the experimental results on several artificial and UCI datasets indicate that the proposed method is of effectiveness and applicability, it not only gives similar or better generalization performance with other popular methods such as TSVR and *ε*-TSVR, but also requires less computational time.**

***Index Terms*—Twin support vector regression, Smooth approximation, Unconstrained convex minimization, Wavelet transform, Iterative approach**

I. Introduction

In recent years, support vector machine (SVM) [1-4] has gained great attention as a powerful method because it shows improved generalization performance in comparison with other machine learning techniques. SVM has been successfully applied in various aspects ranging from pattern recognition, text categorization, and financial regression. As for support vector regression (SVR) model [5], which uses SVM technique to solve the regression estimation, there are many significant methods, such as *ε*-support vector regression (*ε*-SVR) [6], *ν*-support vector regression (*ν*-SVR) [7] and so on.

Manuscript received October 28, 2018; revised Novermber 20, 2018. This work was supported by the National Natural Science Foundation of China under Grants 61304002 and 61403177.

Nannan Zhao is with School of Electronics and Information Engineering, University of Science and Technology Liaoning, Anshan, CO 114051 China. (corresponding author: e-mail: zhaonannan@ustl.edu.cn).

Xinyu Oyang is with School of International Finance & Banking, University of Science and Technology Liaoning, Anshan, CO 114051 China. (Corresponding author, phone: 86-0412-2598139; e-mail: ouyangxinyu@ustl.edu.cn).

Chuang Gao is with School of Electronics and Information Engineering, University of Science and Technology Liaoning, Anshan, CO 114051 China. (e-mail: gaoyouwen@ustl.edu.cn).

Zihui Zang is a postgraduate student in the School of Electronics and Information Engineering, University of Science and Technology Liaoning, Anshan, CO 114051 China. (e-mail: 1003941417@qq.com).

However, the main drawback of SVR is its high learning cost, i.e. $O((2l)^3)$, where l is the number of training samples. In order to improve the computational speed of SVR, Peng [8-10] proposed a twin support vector regression (TSVR) in the sprint of twin support vector machine (TSVM) [11]. Different from SVR, TSVR generates two nonparallel up- and down-bound functions by solving a pair of smaller sized quadratic programming problems (QPPs). It has been proved that learning cost of TSVM is $2O(l^3)$, that is approximately four times faster than the standard SVR in theory. However, Khemchandani *et al.* [12] observed that the model of Peng [8] is not the true spirit of twin methodology, and only the empirical risk minimization principle is considered in TSVR. To overcome these problems, Shao *et al.* [13] presented another twin model for regression termed as *ε*-TSVR, which can be justified on the basis of structural risk minimization principle. Later, Rastogi *et al.* [14] extended *ε*-TSVR and presented *ν*-TSVR, which can automatically optimize the parameters ε_1 and ε_2 according to the sample data. By employing the pinball loss function, Xu *et al.* [15] further developed asymmetric *ν*-twin support vector regression termed as Asy-*ν*-TSVR, which can effectively reduce the disturbance of the noise and improve the generalization performance. Consequently, twin-type SVR has been studied extensively.

Nevertheless, it should be pointed that, in all these twin-type SVR models, the distribution of the training data wasn't considered in solving regression problems. This implies that all samples play the same role on the bound function whether they are important or not, so it will lead to a decline in the performance of the regression. It is more reasonable to give the data samples different penalties depending on their importance. To this end, various methods [16-20] were developed to study this kind of fault. For example, Xu *et al.* [18] presented the K-nearest neighbor-based weighted twin support vector regression by using the local information existing on the samples, and improved the prediction accuracy. By clustering the training data based on their degree of similarity, the authors in [19] presented the modified twin support vector regression. Ye [20] proposed an efficient weighted Lagrangian *ε*-twin support vector regression with quadratic loss functions (WL-*ε*-TSVR), in which the samples are given different penalties by introducing a weight matrix *D* to reduce the impact of the outliers on the regressor to a certain extent.

Traditionally, the down and up bound regressors of twin-type SVR are obtain by their approximate dual solutions. However, Chapelle [21] observed that the approximate dual solution may not result in good primal approximate solution by comparing approximate efficiencies of SVR in primal and dual spaces. There are some related works [22-25] that have been solved directly in the primal space. For example, motivated by the works of twin-type SVR and the Newton approach, Balasundaram *et al.* [23] proposed a new unconstrained Lagrangian TSVR (ULTSVR) to improve the computational speed by solving a pair unconstrained minimization problems. Gupta [24] and Balasundaram [25] used a generalized derivative approach to get the solution of QPPs, although their works were effective and fast, it took only empirical risk minimization into account, not structural risk minimization at the same time.

Motivated by the above researches, this paper proposes an improved wavelet transform based weighted ε-twin support vector regression (WW-ε-TSVR). The main contributions compared with the existing researches of this paper are summarized as follows:

1) Inspired by the K-nearest neighbor-based weight idea, the wavelet transform-based weight is introduced for regressing the time series signal. In the other words, the profile of time series is obtained first by wavelet transform (WT) [26], and then the distance between the training data and the wavelet transformed coefficient is calculated. By taking these distances as weights instead of K-nearest neighbor-based weights, each data is given a different penalty. It is noticed that the proposed WT-based weight TSVR is only suitable for time series signals. In addition, because the ε-TSVR model considers structural risk minimization, combing the ε-TSVR model and the WT-based weight idea makes the proposed algorithm consider both structural risk minimization and empirical risk minimization.

2) After finding a pair of QPPs with constraints, they are first converted into a pair of unconstraint QPPs by introducing smooth functions [27-28]. Then, the unconstrained problem is solved directly in primal space by using generalized derivative iterative approach and Newton iterative approach [29-30] which can reduce the complexities of the model and speed up the training. Further, five more detailed iterative algorithms are given to solve the proposed WW-ε-TSVR.

3) At last, experimental results on both artificial and real datasets show that, compared with the popular TSVR and ε-TSVR, our WW-ε-TSVR significantly shorts the training time with similar or better generalization performance.

The reminder of this paper is organized as follows: Section Ⅱ describes the background including linear TSVR and ε-TSVR. The proposed WW-ε-TSVR is given in section Ⅲ, and five iterative approaches are discussed to solve these unconstrained problems. Experimental results and analysis for both artificial and real datasets are carried out in Section Ⅳ. Section Ⅴ is the conclusion of the proposed work.

II. Background

In this section, we give a brief description of twin support vector regression (TSVR) and ε-twin support vector regression (ε-TSVR). Given a training set $T=\{(x_i,y_i)\}$, $i=1,2,\ldots,l$, where $x_i\in R^n$ and $y_i\in R$. Also let $A=(A_1,A_2,\ldots,A_l)^T$ be the input training sample, and $Y=(y_1,y_2,\ldots,y_l)^T$ be the response of the training samples, where A is an $l\times n$ matrix, A_i, the i-th row of A, represents the i-th training sample, Y is an $l\times 1$ vector, and y_i represents the i-th response.

A. Twin Support Vector Regression (TSVR)

The TSVR model [8] inspired by TSVM [11] finds two nonparallel functions $f_1(x)=w_1^Tx+b_1$ and $f_2(x)=w_2^Tx+b_2$, each one determines the down- and up-bound ε-insensitive regressors, respectively. In the linear case, it can be obtained by solving the following QPPs:

$$\begin{aligned}&\min_{w_1,b_1,\xi}\frac{1}{2}\|Y-\varepsilon_1 e-(Aw_1+eb_1)\|^2+c_1e^T\xi,\\ &s.t.\quad Y-(Aw_1+eb_1)\ge\varepsilon_1 e-\xi\ ,\ \ \xi\ge 0.\end{aligned}\tag{1}$$

and

$$\begin{aligned}&\min_{w_2,b_2,\eta}\frac{1}{2}\|Y+\varepsilon_2 e-(Aw_2+eb_2)\|^2+c_2e^T\eta,\\ &s.t.\quad (Aw_2+eb_2)-Y\ge\varepsilon_2 e-\eta\ ,\ \ \eta\ge 0.\end{aligned}\tag{2}$$

where $c_1, c_2>0$ are the pre-specified penalty factors, $\varepsilon_1,\varepsilon_2>0$ are constants, ξ and η are slack vectors, and e is column vector of 'ones' of appropriate dimension. Their dual problems are

$$\begin{aligned}&\min_{\alpha}\frac{1}{2}\alpha^TG(G^TG)^{-1}G^T\alpha-f^TG(G^TG)^{-1}G^T\alpha+f^T\alpha,\\ &s.t.\quad 0\le\alpha\le c_1e.\end{aligned}\tag{3}$$

and

$$\begin{aligned}&\min_{\gamma}\frac{1}{2}\gamma^TG(G^TG)^{-1}G^T\gamma+h^TG(G^TG)^{-1}G^T\gamma-h^T\gamma,\\ &s.t.\quad 0\le\gamma\le c_2e.\end{aligned}\tag{4}$$

where $f=Y-\varepsilon_1e$, $h=Y+\varepsilon_2e$, and $G=[A\ e]$. The augmented vectors $u_1=[w_1^T\ b_1]^T$ *a*nd $u_2=[w_2^T\ b_2]^T$ are determined by $u_1=(G^TG)^{-1}G^T(f-\alpha)$ and $u_2=(G^TG)^{-1}G^T(h+\gamma)$, respectively. The final estimated regressor is constructed as

$$f(x)=\frac{1}{2}\big(f_1(x)+f_2(x)\big)=\frac{1}{2}(w_1+w_2)^Tx+\frac{1}{2}(b_1+b_2)\tag{5}$$

B. ε-Twin Support Vector Regression (ε-TSVR)

Following the idea of TWSVM and TSVR, Shao *et al.* [13] proposed another twin type SVR termed as ε-twin support vector regression (ε-TSVR). It also finds two insensitive proximal linear function: $f_1(x)=w_1^Tx+b_1$ and $f_2(x)=w_2^Tx+b_2$. Introducing the regularization terms $\frac{1}{2}(w_1^Tw_1+b_1^2)$, $\frac{1}{2}(w_2^Tw_2+b_2^2)$, and the slack variables ξ,ξ^*,η,η^*, the primal problems can be expressed as

$$\begin{aligned}&\min_{w_1,b_1,\xi,\xi^*}\frac{1}{2}c_3(w_1^Tw_1+b_1^2)+\frac{1}{2}\xi^{*T}\xi^*+c_1e^T\xi,\\ &s.t.\quad Y-(Aw_1+eb_1)=\xi^*,\\ &\qquad\ \ Y-(Aw_1+eb_1)\ge-\varepsilon_1e-\xi\ ,\ \ \xi\ge 0.\end{aligned}\tag{6}$$

and

$$\begin{aligned}&\min_{w_2,b_2,\eta,\eta^*}\frac{1}{2}c_4(w_2^Tw_2+b_2^2)+\frac{1}{2}\eta^{*T}\eta^*+c_2e^T\eta,\\ &s.t.\quad (Aw_2+eb_2)-Y=\eta^*,\\ &\qquad\ \ (Aw_2+eb_2)-Y\ge-\varepsilon_2e-\eta\ ,\ \ \eta\ge 0.\end{aligned}\tag{7}$$

where c_1, c_2, c_3, c_4, ε_1 and ε_2 are positive parameters. The main difference between TSVR and ε-TSVR is an extra regularization term $\frac{1}{2}c_3(w_1^Tw_1+b_1^2)$ in (6) (respectively $\frac{1}{2}c_4(w_2^Tw_2+b_2^2)$ in (7)) so that the structural risk minimization principle is implemented.

In order to get the solutions of (6) and (7), their dual problems need to be derived. By introducing the Lagrangian multiplies α and γ, the dual problems of ε-TSVR can be derived as follows,

$$\begin{aligned}&\min_{\alpha}\frac{1}{2}\alpha^TG(G^TG+c_3I)^{-1}G^T\alpha-Y^TG(G^TG+c_3I)^{-1}G^T\alpha+(e^T\varepsilon_1+Y^T)\alpha,\\ &s.t.\quad 0\le\alpha\le c_1e.\end{aligned}\tag{8}$$

and

$$\min_{\gamma} \frac{1}{2}\gamma^T G(G^T G + c_4 I)^{-1} G^T \gamma + Y^T G(G^T G + c_4 I)^{-1} G^T \gamma - (Y^T - e^T \varepsilon_2)\gamma, \quad s.t. \quad 0 \le \gamma \le c_2 e. \tag{9}$$

The final regressor $f(x)$ is in the same form as (5), in which u_1, u_2 are as follows:

$$u_1 = [w_1^T \quad b_1^T]^T = (G^T G + c_3 I)^{-1} G^T (Y - \alpha) \tag{10}$$

$$u_2 = [w_2^T \quad b_2^T]^T = (G^T G + c_4 I)^{-1} G^T (Y + \gamma) \tag{11}$$

III. Proposed WW-ε-TSVR method

Motivated by the above studies, wavelet transform based weighted ε-twin support vector regression (WW-ε-TSVR) is proposed in this section. Further, a pair of unconstrained convex minimizations in primal space will be given to obtain the solutions of the proposed regression model.

A. Wavelet Transform Based Weighted Method

In ε-TSVR, all samples are given the same weight, which is not suitable for many practical situations, and it also reduces the regression performance due to the impact of outliers. Therefore, it is necessary to give different weight to different sampling points in the training of regression model. Next, a method of choosing weights based on wavelet transform [26] is introduced. Assuming $Y = (y_1, y_2, \cdots, y_l)^T$ is a time series, the wavelet transform is a good mathematical tool that can be used to extract its low-frequency component because of the time frequency localization properties of wavelets. First, perform *k* levels of the wavelet transform on sample signal *Y*, and then reconstruct only the low-frequency components to get $Y_w = (y_{w1}, y_{w2}, \cdots, y_{wl})^T$, which is actually the profile of sample single *Y*. It can be assumed that the closer the sample point is to y_{wi}, the greater the weight that should be given. Defining a variable$d_i = \lambda - |y_{wi} - y_i|$ represents the weight, where λ is a constant value, for example, $\lambda = max(d_i)$. In addition, it should be noted that the method based on wavelet transform to determine weights is only suitable for time series signals, but for random sequences, $d_i = 1$.

B. Proposed WW-ε-TSVR Method

A novel WW-ε-TSVR method is proposed in this subsection. By introducing matrix *D*, that represents the penalty for each sampling point, the proposed algorithm in linear form is obtained by solving the following pair of QPPs:

$$\min_{w_1, b_1, \xi} \frac{1}{2}\big(Y - (Aw_1 + eb_1)\big)^T D\big(Y - (Aw_1 + eb_1)\big) + \frac{1}{2}c_3(w_1^T w_1 + b_1^2) + c_1 e^T \xi, \quad s.t. \quad Y - (Aw_1 + eb_1) \ge -\varepsilon_1 e - \xi, \ \xi \ge 0. \tag{12}$$

and

$$\min_{w_2, b_2, \eta} \frac{1}{2}\big(Y - (Aw_2 + eb_2)\big)^T D\big(Y - (Aw_2 + eb_2)\big) + \frac{1}{2}c_4(w_2^T w_2 + b_2^2) + c_2 e^T \eta, \quad s.t. \quad (Aw_2 + eb_2) - Y \ge -\varepsilon_2 e - \eta, \ \eta \ge 0. \tag{13}$$

where c_1, c_2, c_3, c_4, ε_1 and ε_2 are user specified positive parameters, e is column vector of 'ones' of appropriate dimension, and $D = diag(d_1, d_2, \cdots, d_l)$ is a wavelet weight diagonal matrix. If $D \equiv I$, the proposed method degrades into the original ε-TSVR.

WW-ε-TSVR finds two proximal functions $f_1(x) = w_1^T x + b_1$ and $f_2(x) = w_2^T x + b_2$, and the final regressor $f(x)$, which is in the same form as (5), is determined by the mean of these two proximal functions. Note that there are three terms in the objective function (12). The first term is the sum of squared distances from the shifted function $f_1(x)$ to the training points, and different weights are also given to each training sample base on the matric *D*. Therefore, minimizing it can cause the function $f_1(x)$ to be suitable for the training samples. The constraint requires the estimated function $f_1(x)$ to be at a distance of at lease ε_1 from the training samples. The slack vector ξ is introduced to measure the error wherever the distance is closer to ε_1. The second term is an extra regularization term, the structural risk is minimized in (12) due to this term. The third term of the objective function minimizes the sum of error variables, which attempts to over-fit the training points. For the optimization problem (13), we have the similar illustrations.

Next, by introducing the addition function $(\cdot)_+$, the constraint problems (12) and (13) are converted into a pair of unconstrained problems and rewritten as

$$\min_{u_1} L_1(u_1) = \frac{1}{2}(Y - Gu_1)^T D(Y - Gu_1) + \frac{1}{2}c_3 u_1^T u_1 + c_1 e^T (Gu_1 - f_1)_+ \tag{14}$$

and

$$\min_{u_2} L_2(u_2) = \frac{1}{2}(Y - Gu_2)^T D(Y - Gu_2) + \frac{1}{2}c_4 u_2^T u_2 + c_2 e^T (f_2 - Gu_2)_+ \tag{15}$$

where $f_1 = Y + \varepsilon_1 e$, $f_2 = Y - \varepsilon_2$, $u_1 = [w_1^T \ b_1]^T$, $u_2 = [w_2^T \ b_2]^T$, $G = [A \ e]$ for the linear case and $G = [K(A, A^T) \ e]$ for the nonlinear case. It can be seen that (14) and (15) solve two smaller sized unconstrained minimization problems. However, it is worth mention that the 'plus' functions that exist on (14) and (15) are not differentiable. In order to solve this problem, these non-smooth 'plus' functions should be replaced by smooth approximate functions. Denoting $p(\cdot)$ as a smooth function, (14) and (15) are modified as

$$\min_{u_1} L_1(u_1) = \frac{1}{2}(Y - Gu_1)^T D(Y - Gu_1) + \frac{1}{2}c_3 u_1^T u_1 + c_1 e^T p(Gu_1 - f_1) \tag{16}$$

and

$$\min_{u_2} L_2(u_2) = \frac{1}{2}(Y - Gu_2)^T D(Y - Gu_2) + \frac{1}{2}c_4 u_2^T u_2 + c_2 e^T p(f_2 - Gu_2) \tag{17}$$

The solution of unconstrained in (16) and (17) can be sovled by computing their critical point i.e. $\nabla L_1(u_1) = 0$ and $\nabla L_2(u_2) = 0$. Next, the generalized derivative iterative approach and Newton iterative approach are applied to solve the solution directly. Before that, three popular smoothing functions are first introduced, each of which will be used to substitute into (16) and (17) in this work. The definitions of them are as follows:

$$p_1(x, x_0) = \frac{1}{4}\frac{x^2}{|x_0|} + \frac{1}{2}x + \frac{1}{4}|x_0| \tag{18}$$

$$p_2(x, \alpha) = x + \frac{1}{\alpha}\log(1 + e^{-\alpha x}) \tag{19}$$

$$p_3(x, \alpha) = \frac{x + \sqrt{x^2 + 4\alpha^2}}{2} \tag{20}$$

C. Generalized Derivative Iterative Approach

In this subsection, the generalized derivative iterative approach is used to solve the proposed primal unconstrained problems of (16) and (17). In fact, the generalized gradient of problems are as follows:

$$\nabla L_1(u_1) = (G^T DG + c_3 I)u_1 - G^T DY + c_1 G^T p^*(Gu_1 - f_1)$$

and

$$\nabla L_2(u_2) = (G^T DG + c_4 I)u_2 - G^T DY - c_2 G^T p^*(f_1 - Gu_2)$$

where $p^*(\cdot)$ is the sub-gradient of $p(\cdot)$. Computing $\nabla L_1(u_1) = 0$, $\nabla L_2(u_2) = 0$, defining $Q_1 = (G^T DG + c_3 I)$, $Q_2 = (G^T DG + c_4 I)$ and denoting k as the *k*-th iteration, the following iterative scheme can be gotten:

$$Q_1 u_1^{k+1} = G^T DY - c_1 G^T p^*(Gu_1^k - f_1) \tag{21}$$

and

$$Q_2 u_2^{k+1} = G^T DY + c_2 G^T p^*(f_2 - Gu_2^k) \tag{22}$$

Next, three popular smoothing functions will be substituted into (21) and (22), respectively.

The first smooth function $p_1(x, x_0)$ that is differential is first considered here, where x_0 is a non-zero real number. This function will be a better approximation of 'plus' function as long as $|x_0|$ is very close to $|x|$. From (21) and (22), we can get

$$Q_1 u_1^{k+1} = G^T DY - c_1 G^T \left(\frac{Gu_1^k - f_1}{2|x_0|} + \frac{e}{2}\right) \tag{23}$$

and

$$Q_2 u_2^{k+1} = G^T DY + c_2 G^T \left(\frac{f_2 - Gu_2^k}{2|x_0|} + \frac{e}{2}\right) \tag{24}$$

Obviously, the solutions u_1 and u_2 depend on the choice of x_0, so that the vector $|\mathrm{x}_0|$ can be adjusted till it is close to $\left|Gu_1^k - f_1\right|$ for (23) and close to $\left|f_2 - Gu_2^k\right|$ for (24). The further iterative procedures are obtained as follows:

$$u_1^{k+1} = \left(Q_1 + \frac{c_1}{2} G^T d_1^{p_1} G\right)^{-1} G^T \left(DY + \frac{c_1}{2} d_1^{p_1} f_1 - \frac{c_1}{2} e\right) \tag{25}$$

and

$$u_2^{k+1} = \left(Q_2 + \frac{c_2}{2} G^T d_2^{p_1} G\right)^{-1} G^T \left(DY + \frac{c_2}{2} d_2^{p_1} f_2 + \frac{c_2}{2} e\right) \tag{26}$$

where $d_1^{p_1} = diag(\left|Gu_1^k - f_1\right|^{-1})$, $d_2^{p_1} = diag(\left|f_2 - Gu_2^k\right|^{-1})$ to simplify our expression.

Now let's summarize the proposed smooth WW-ε-TSVR model as **Algorithm 1:** GWW-ε-TSVR1, where only the iterative algorithm for (25) is given here, and the iterative algorithm for (26) is similar.

```
Algorithm 1: GWW-ε-TSVR1
Input:
    · tol: the error tolerance for learning accuracy.
    · itmax: the maximum number of iterations.
    · $Q_1, G, D, c_1, \varepsilon_1$
    · $u_1^{old} = e$: initial approximation of $u_1$.
    · $f_1 = Y + \varepsilon_1 e, z_1 = DY - \frac{c_1}{2} e$
    · $k = 0$
Output:
    The optimal solution $u_1$.
Process:
    do {
    · $d_1^{p_1} = diag\left(\left(\left|Gu_1^{old} - f_1\right| + \rho e\right)^{-1}\right)$, where $\rho > 0$ is a small positive number.
    · Solve $u_1^{new} = \left(Q_1 + \frac{c_1}{2} G^T d_1^{p_1} G\right)^{-1} G^T (z_1 + \frac{c_1}{2} d_1^{p_1} f_1)$
    · $err = u_1^{new} - u_1^{old}$
    · $u_1^{old} = u_1^{new}$
    · $k = k + 1$
    } while ($|err| > tol$ and $k < itmax$)
```

The second smooth function introduced by this paper is $p_2(x, \alpha) = x + \frac{1}{\alpha} log(1 + exp(-\alpha x))$, where the parameter $\alpha > 0$. Using the generalized derivative iterative approach, we can obtain

$$u_1^{k+1} = Q_1^{-1} G^T \left[DY - \frac{c_1}{1 + \exp\left(-\alpha(Gu_1^k - f_1)\right)}\right] \tag{27}$$

and

$$u_2^{k+1} = Q_2^{-1} G^T \left[DY + \frac{c_2}{1 + \exp\left(-\alpha(f_2 - Gu_2^k)\right)}\right] \tag{28}$$

The proposed smooth WW-ε-TSVR model of (27) is summarized **Algorithm 2:** GWW-ε-TSVR2 given in Appendix A.

The third smooth function introduced by this paper is $p_3(x, \alpha) = \frac{x + \sqrt{x^2 + 4\alpha^2}}{2}$. To simplify our expression, $d_1^{p_3} = diag(\frac{1}{\sqrt{(Gu_1 - f_1).\hat{}2 + 4\alpha^2}})$, $d_2^{p_3} = diag(\frac{1}{\sqrt{(f_2 - Gu_2).\hat{}2 + 4\alpha^2}})$, and its iterative formulation can be obtained as follows:

$$u_1^{k+1} = Q_1^{-1} G^T \left[DY - \frac{c_1}{2}\left(e + d_1^{p_3}(Gu_1^k - f_1)\right)\right] \tag{29}$$

and

$$u_2^{k+1} = Q_2^{-1} G^T \left[DY + \frac{c_2}{2}\left(e + d_2^{p_3}(f_2 - Gu_2^k)\right)\right] \tag{30}$$

The proposed smooth WW-ε-TSVR model of (29) is **Algorithm 3:** GWW-ε-TSVR3 shown in Appendix A.

D. Newton Iterative Approach

In this subsection, Newton iterative method, which is a good mathematical tool commonly used to solve extreme problems, is used to solve the proposed algorithm. $p_2(x, \alpha)$ and $p_3(x, \alpha)$ mentioned above are discussed here.

For the second smooth function $p_2(x, \alpha)$, the minimization problems (16) and (17) are modified as

$$\min_{u_1} L_1(u_1) = \frac{1}{2}(Y - Gu_1)^T D(Y - Gu_1) + \frac{1}{2} c_3 u_1^T u_1 + c_1 e^T p_2(Gu_1 - f_1, \alpha) \tag{31}$$

and

$$\min_{u_2} L_2(u_2) = \frac{1}{2}(Y - Gu_2)^T D(Y - Gu_2) + \frac{1}{2} c_4 u_2^T u_2 + c_2 e^T p_2(f_2 - Gu_2, \alpha) \tag{32}$$

The gradient vector $\nabla L_1(u_1)$ and $\nabla L_2(u_2)$ of the (31) and (32) can be obtained as

$$\nabla L_1(u_1) = Q_1 u_1 - G^T \left[DY - \frac{c_1}{1 + \exp\left(-\alpha(Gu_1 - f_1)\right)}\right] \tag{33}$$

and

$$\nabla L_2(u_2) = Q_2 u_2 - G^T \left[DY + \frac{c_2}{1 + \exp\left(-\alpha(f_2 - Gu_2)\right)}\right] \tag{34}$$

Defining $d_1^{p_2} = diag((1 + exp(-\alpha(Gu_1 - f_1)))^{-1})$, and $d_2^{p_2} = diag((1 + exp(-\alpha(f_2 - Gu_2)))^{-1})$, (33) and (34) can be modified as follows:

$$\nabla L_1(u_1) = Q_1 u_1 - G^T DY + c_1 G^T d_1^{p_2} e \tag{35}$$

and

$$\nabla L_2(u_2) = Q_2 u_2 - G^T DY - c_2 G^T d_2^{p_2} e \tag{36}$$

Then, $\nabla^2 L_1(u_1)$ and $\nabla^2 L_2(u_2)$ are as follows

$$\nabla^2 L_1(u_1) = Q_1 + \alpha c_1 G^T diag\left(\frac{\exp\left(-\alpha(Gu_1 - f_1)\right)}{\left(1 + \exp\left(-\alpha(Gu_1 - f_1)\right)\right)^2}\right) G \tag{37}$$

and

$$\nabla^2 L_2(u_2) = Q_2 + \alpha c_2 G^T diag\left(\frac{\exp\left(-\alpha(f_2 - Gu_2)\right)}{\left(1 + \exp\left(-\alpha(f_2 - Gu_2)\right)\right)^2}\right) G \tag{38}$$

Let's define

$$d_3^{p_2} = diag\left(\frac{\exp\left(-\alpha(Gu_1 - f_1)\right)}{\left(1 + \exp\left(-\alpha(Gu_1 - f_1)\right)\right)^2}\right)$$

and

$$d_4^{p_2} = diag\left(\frac{exp(-\alpha(f_2 - Gu_2))}{(1 + exp(-\alpha(f_2 - Gu_2)))^2}\right)$$

then (37) and (38) can be modified as

$$\nabla^2 L_1(u_1) = Q_1 + \alpha c_1 G^T d_3^{p_2} G\,, \tag{39}$$

and

$$\nabla^2 L_2(u_2) = Q_2 + \alpha c_2 G^T d_4^{p_2} G\,. \tag{40}$$

Next, Newton method can be applied to solve (31) and (32) by combining (33) to (40). This algorithm is termed as **Algorithm 4**: NWW-ε-TSVR2 in Appendix A, and its iterative procedure is as follows:

$$(Q_1+\alpha c_1 G^T d_3^{p_2} G)(u_1^{k+1}-u_1^k)=-\left(Q_1 u_1-G^T DY+c_1 G^T d_1^{p_2} e\right) \quad (41)$$

and

$$(Q_2+\alpha c_2 G^T d_4^{p_2} G)(u_2^{k+1}-u_2^k)=-\left(Q_2 u_2-G^T DY-c_2 G^T d_2^{p_2} e\right) \quad (42)$$

For the third smooth function $p_3(x,\alpha)=\frac{x+\sqrt{x^2+4\alpha^2}}{2}$, the minimization problems (16) and (17) are modified as

$$\min_{u_1} L_1(u_1)=\frac{1}{2}(Y-Gu_1)^T D(Y-Gu_1)+\frac{1}{2}c_3 u_1^T u_1 \\ +c_1 e^T p_3(Gu_1-f_1,\alpha) \quad (43)$$

and

$$\min_{u_2} L_2(u_2)=\frac{1}{2}(Y-Gu_2)^T D(Y-Gu_2)+\frac{1}{2}c_4 u_2^T u_2 \\ +c_2 e^T p_4(f_2-Gu_2,\alpha) \quad (44)$$

So the gradient vector $\nabla L_1(u_1)$ and $\nabla L_2(u_2)$ can be obtained for the (43) and (44) as

$$\nabla L_1(u_1)=Q_1 u_1-G^T DY+\frac{1}{2}c_1 G^T(e+d_1^{p_3}(Gu_1-f_1)) \quad (45)$$

and

$$\nabla L_2(u_2)=Q_2 u_2-G^T DY-\frac{1}{2}c_2 G^T(e+d_2^{p_3}(f_2-Gu_2)) \quad (46)$$

where the definition of $d_1^{p_3}$ and $d_2^{p_3}$ is the same as the **Algorithm 3**, then we get

$$\nabla^2 L_1(u_1)=Q_1+2\alpha^2 c_1 G^T d_3^{p_3} G \quad (47)$$

and

$$\nabla^2 L_2(u_2)=Q_2+2\alpha^2 c_2 G^T d_4^{p_3} G \quad (48)$$

where

$$d_3^{p_3}=diag\left(\frac{1}{(\sqrt{(Gu_1-f_1).^2+4\alpha})^3}\right)$$

and

$$d_4^{p_3}=diag\left(\frac{1}{(\sqrt{(f_2-Gu_2).^2+4\alpha})^3}\right)$$

Combining (45) with (48), we have

$$(Q_1+2\alpha^2 c_1 G^T d_3^{p_3} G)(u_1^{k+1}-u_1^k)= \\ -(Q_1 u_1-G^T DY+\frac{1}{2}c_1 G^T(e+d_1^{p_3}(Gu_1-f_1))) \quad (49)$$

and

$$(Q_2+2\alpha c_2 G^T d_4^{p_3} G)(u_2^{k+1}-u_2^k)= \\ -(Q_2 u_2-G^T DY-\frac{1}{2}c_2 G^T(e+d_2^{p_3}(f_2-Gu_2))) \quad (50)$$

This algorithm is summarized as **Algorithm 5**: NWW-ε-TSVR3 given in Appendix A.

IV. Experimental results

To demonstrate the performance of the proposed WW-ε-TSVR, we compare it with the popular TSVR and ε-TSVR in several datasets, including 5 types of artificial datasets and 5 benchmark datasets [31]. All computations are carried out on Windows 7 OS Intel Core i5-4210U CPU (2.4GHz) with 4GB RAM and MATLAB R2014a environment. In order to decrease the computational complexity in parameter selection, let $c_1=c_2$, $c_3=c_4$, and $\varepsilon_1=\varepsilon_2$. Gaussian kernel function defined by $k(a,b)=exp(-\frac{\|a-b\|^2}{\rho})$ is used to deal with nonlinear situations, where the vectors $a,b\in R^n$, and the parameter $\rho>0$. 3-level discrete wavelet transform (DWT), and db2 wavelet are choose for the wavelet transform algorithm of all the following experiments.

Some commonly used evaluation criterions [8] as shown in Table I are introduced before evaluating the performances of these methods. Without loss generality, let m be the number of test samples, y_i be the real-value of sample x_i, $\hat{y}_i$ be the prediction of y_i, and $\bar{y}=(\sum_i y_i)/m$ be the mean of $y_1,y_2,\cdots,y_m$. SSE represents the fitting precision, namely, the smaller is SSE, the fitter the estimation is, and small SSE/SST means good agreement between estimations and real-values, while the smaller SSE/SST is usually accompanied by an increase of SSR/SST [8].

TABLE I
PERFORMANCE METRICS AND THEIR CALCULATION

Metric	Calculation
SSE	$\sum_{i=1}^m (y_i-\hat{y}_i)^2$
SSR	$\sum_{i=1}^m (\hat{y}_i-\bar{y})^2$
SST	$\sum_{i=1}^m (y_i-\bar{y})^2$
SSE/SST	$\sum_{i=1}^m (y_i-\hat{y}_i)^2/\sum_{i=1}^m (y_i-\bar{y})^2$
SSR/SST	$\sum_{i=1}^m (\hat{y}_i-\bar{y})^2/\sum_{i=1}^m (y_i-\bar{y})^2$

A. *Artificial Datasets*

To test the regression performance of our proposed WW-ε-TSVR, five functions shown in Table II are used to generate all artificial datasets. The training data's observed values are polluted by the form $y=f(x)+err$, where err is an additive noise by adding $U[a,b]$ or $N(\mu,\sigma^2)$. $U[a,b]$ represents the uniformly random variable in $[a,b]$ and $N(\mu,\sigma^2)$ represents the Gaussian random variable with means μ and variance σ^2, respectively. To avoid biased comparisons, twenty independent groups of noisy samples for each kind of noise are generated randomly by Matlab toolbox, each of which includes 260 samples for training and 500 samples for testing for each function. Besides, testing data points are uniformly sampled from the objective function without any noise.

TABLE II
FUNCTIONS DEFINITION USED FOR GENERATING ARTIFICIAL DATASETS

Name	Function definition	Domain of definition
$f_1(x)$	$sin(x)/x$	$x\in[-4\pi,4\pi]$
$f_2(x)$	$\left\|x^{2/3}\right\|$	$x\in[-2,1]$
$f_3(x)$	$0.2sin(2\pi x)+0.2x^2+0.3$	$x\in[0,2]$
$f_4(x)$	$\frac{4}{\|x\|+2}+\cos(2x)+\sin(3x)$	$x\in[-10,10]$
$f_5(x)$	$1.9[1.35+e^{x_1}\sin(13(x_1-0.6)^2)$ $+e^{3x_2-1.5}\sin(4\pi(x_2-0.9)^2)]$	$x_1\in[0,1]$, $x_2\in[0,1]$

For simplicity and clarity, one run results of ε-TSVR and NWW-ε-TSVR2 on $f_1(x)$ to $f_4(x)$ with $N(0,0.1^2)$ are shown in Fig. 1, and their SSE values are 0.4504 and 0.4100 for $f_1(x)$, 0.9813 and 0.8127 for $f_2(x)$, 0.3464 and 0.0435 for $f_3(x)$ and 0.8765 and 0.7829 for $f_4(x)$, respectively. It can be observed that our proposed algorithm has better approximates than ε-TSVR. Table III shows the performance comparisons of our proposed GWW-ε-TSVR1, GWW-ε-TSVR2, GWW-ε-TSVR3, NWW-ε-TSVR2 and NWW-ε-TSVR3 with TSVR and ε-TSVR on artificial datasets for uniformly distributed noise over the interval [-0.2, 0.2] with Gaussian kernel. Table IV shows the performance comparisons of our proposed methods for Gaussian noise with mean zero and standard deviation 0.1. It can be seen that our proposed five algorithms have similar generalization performance, but they all have better approximates than TSVR and ε-TSVR. Besides, Table III and Table IV also compare the training CPU time for all methods, and it can be seen that our proposed methods are the fastest learning method. This indicates the statistical information in the training datasets is well expressed by our WW-ε-TSVR with the lowest average regression errors and the fastest learning speed.

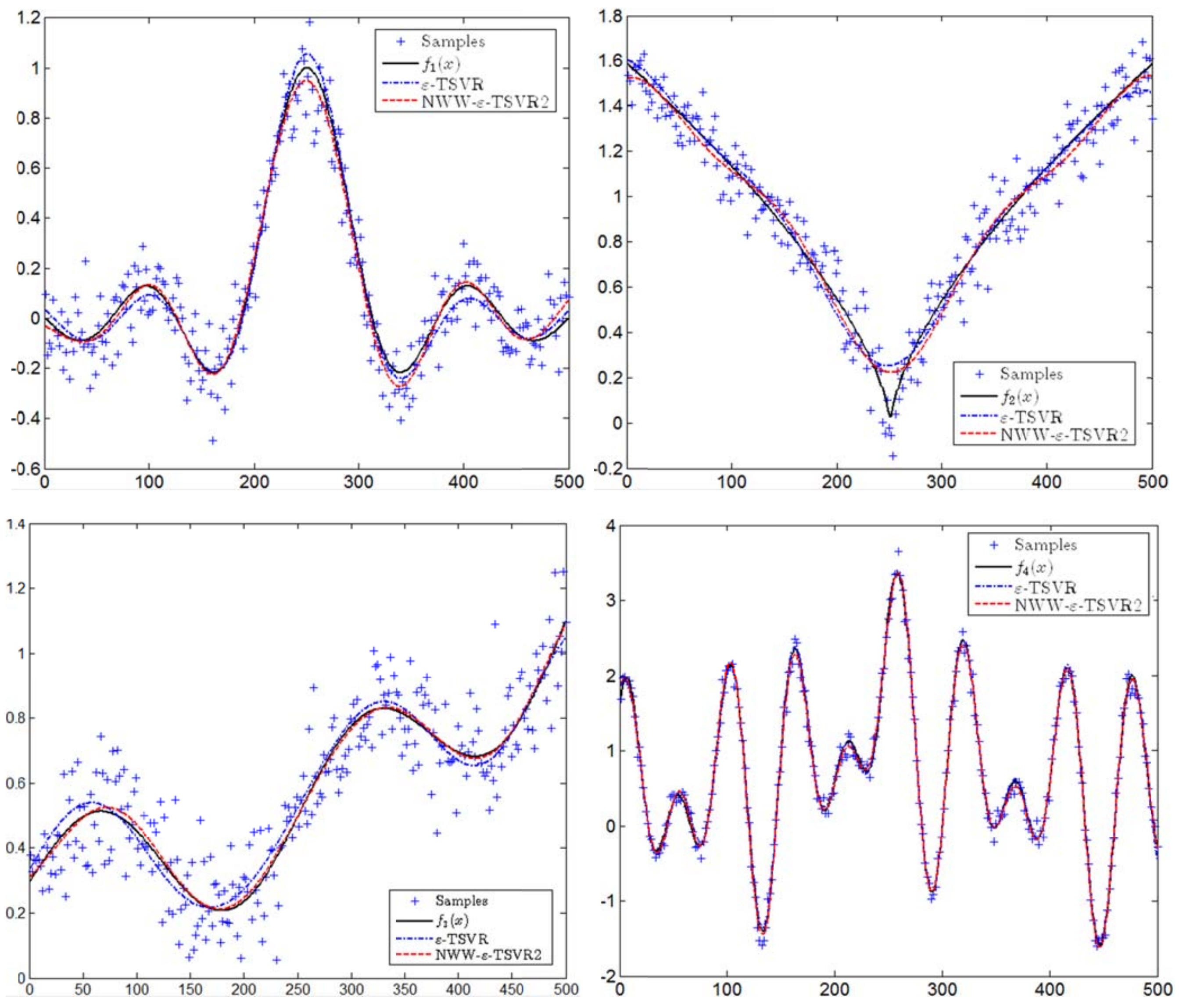

Fig. 1 Performance Comparison of ε-TSVR and NWW-ε-TSVR2 on $f_1(x)$ to $f_4(x)$ with $N(0, 0.1^2)$.

TABLE III
PERFORMANCE COMPARISON ON ARTIFICIAL DATASETS FOR U[-0.2, 0.2]

Dataset	Regressor	SSE	SSE/SST	SSR/SST	CPU sec.
	TSVR	0.4859 ± 0.2246	0.0090 ± 0.0042	0.9787 ± 0.0452	0.25
	ε-TSVR	0.3332 ± 0.1667	0.0062 ± 0.0031	0.9760 ± 0.0390	0.25
	GWW-ε-TSVR1	0.2874 ± 0.1499	0.0053 ± 0.0028	1.0129 ± 0.0463	0.06
f_1(x)	GWW-ε-TSVR2	0.2802 ± 0.1535	0.0052 ± 0.0029	1.0159 ± 0.0451	0.02
	GWW-ε-TSVR3	0.3355 ± 0.1847	0.0062 ± 0.0034	1.0166 ± 0.0453	0.02
	NWW-ε-TSVR1	0.2800 ± 0.1533	0.0052 ± 0.0029	1.0159 ± 0.0451	0.04
	NWW-ε-TSVR2	0.2776 ± 0.1544	0.0052 ± 0.0029	1.0155 ± 0.0443	0.05
	TSVR	0.9692 ± 0.1653	0.0112 ± 0.0019	0.9919 ± 0.0340	0.24
	ε-TSVR	0.9568 ± 0.1319	0.0110 ± 0.0015	0.9893 ± 0.0341	0.24
	GWW-ε-TSVR1	0.8765 ± 0.0462	0.0101 ± 0.0005	0.9843 ± 0.0394	0.09
f_2(x)	GWW-ε-TSVR2	0.8767 ± 0.0510	0.0101 ± 0.0006	0.9874 ± 0.0394	0.02
	GWW-ε-TSVR3	0.9162 ± 0.0528	0.0106 ± 0.0006	0.9887 ± 0.0388	0.03
	NWW-ε-TSVR1	0.8767 ± 0.0510	0.0101 ± 0.0006	0.9874 ± 0.0394	0.04
	NWW-ε-TSVR2	0.8713 ± 0.0494	0.0101 ± 0.0006	0.9859 ± 0.0388	0.06
	TSVR	0.4063 ± 0.1998	0.0157 ± 0.0077	1.0224 ± 0.0619	0.24
	ε-TSVR	0.3223 ± 0.1533	0.0125 ± 0.0059	1.0083 ± 0.0536	0.24
f_3(x)	GWW-ε-TSVR1	0.1746 ± 0.0714	0.0068 ± 0.0028	0.9930 ± 0.0354	0.10
	GWW-ε-TSVR2	0.1710 ± 0.0665	0.0066 ± 0.0026	0.9936 ± 0.0377	0.02
	GWW-ε-TSVR3	0.2073 ± 0.0962	0.0080 ± 0.0037	0.9946 ± 0.0355	0.03

	NWW-ε-TSVR1	0.1710 ± 0.0664	0.0066 ± 0.0026	0.9936 ± 0.0377	0.04
	NWW-ε-TSVR2	0.1646 ± 0.0722	0.0064 ± 0.0028	0.9951 ± 0.0328	0.06
f_4(x)	TSVR	1.0306 ± 0.2667	0.0018 ± 0.0005	1.0006 ± 0.0172	0.23
	ε-TSVR	0.9282 ± 0.1762	0.0016 ± 0.0003	0.9966 ± 0.0147	0.24
	GWW-ε-TSVR1	0.7722 ± 0.0863	0.0013 ± 0.0001	0.9968 ± 0.0114	0.11
	GWW-ε-TSVR2	0.8535 ± 0.0997	0.0015 ± 0.0002	0.9955 ± 0.0129	0.02
	GWW-ε-TSVR3	0.7865 ± 0.0938	0.0013 ± 0.0002	0.9965 ± 0.0120	0.04
	NWW-ε-TSVR1	0.8535 ± 0.0997	0.0015 ± 0.0002	0.9955 ± 0.0129	0.04
	NWW-ε-TSVR2	0.7857 ± 0.0900	0.0013 ± 0.0002	0.9964 ± 0.0119	0.08
f_5(x)	TSVR	0.7041 ± 0.1352	0.0006 ± 0.0001	1.0000 ± 0.0102	0.24
	ε-TSVR	0.6288 ± 0.1222	0.0005 ± 0.0001	0.9997 ± 0.0092	0.24
	GWW-ε-TSVR1	0.5677 ± 0.1021	0.0005 ± 0.0001	1.0027 ± 0.0064	0.10
	GWW-ε-TSVR2	0.5748 ± 0.1058	0.0005 ± 0.0001	1.0026 ± 0.0062	0.02
	GWW-ε-TSVR3	0.5468 ± 0.1015	0.0005 ± 0.0001	1.0026 ± 0.0063	0.03
	NWW-ε-TSVR1	0.5748 ± 0.1058	0.0005 ± 0.0001	1.0026 ± 0.0062	0.04
	NWW-ε-TSVR2	0.5467 ± 0.1016	0.0005 ± 0.0001	1.0026 ± 0.0063	0.07

TABLE IV
PERFORMANCE COMPARISON ON ARTIFICIAL DATASETS FOR $N[0, 0.1^2]$

Dataset	Regressor	SSE	SSE/SST	SSR/SST	CPU sec.
f_1(x)	TSVR	0.2825 ± 0.1060	0.0053 ± 0.0020	1.0107 ± 0.0390	0.23
	ε-TSVR	0.2383 ± 0.0952	0.0044 ± 0.0018	1.0089 ± 0.0411	0.22
	GWW-ε-TSVR1	0.2243 ± 0.1102	0.0042 ± 0.0020	0.9914 ± 0.0264	0.06
	GWW-ε-TSVR2	0.2174 ± 0.1027	0.0040 ± 0.0019	0.9879 ± 0.0211	0.02
	GWW-ε-TSVR3	0.2314 ± 0.0828	0.0043 ± 0.0015	0.9883 ± 0.0212	0.02
	NWW-ε-TSVR1	0.2175 ± 0.1029	0.0040 ± 0.0019	0.9879 ± 0.0211	0.03
	NWW-ε-TSVR2	0.2174 ± 0.0979	0.0040 ± 0.0018	0.9880 ± 0.0215	0.04
f_2(x)	TSVR	0.9222 ± 0.0812	0.0115 ± 0.0009	0.9864 ± 0.0297	0.21
	ε-TSVR	0.8814 ± 0.0604	0.0102 ± 0.0007	0.9930 ± 0.0299	0.22
	GWW-ε-TSVR1	0.8274 ± 0.0584	0.0096 ± 0.0007	0.9846 ± 0.0234	0.09
	GWW-ε-TSVR2	0.8253 ± 0.0524	0.0095 ± 0.0006	0.9839 ± 0.0249	0.02
	GWW-ε-TSVR3	0.8649 ± 0.0717	0.0100 ± 0.0008	0.9844 ± 0.0245	0.02
	NWW-ε-TSVR1	0.8251 ± 0.0522	0.0095 ± 0.0006	0.9839 ± 0.0249	0.03
	NWW-ε-TSVR2	0.8269 ± 0.0548	0.0095 ± 0.0006	0.9850 ± 0.0250	0.04
f_3(x)	TSVR	0.2000 ± 0.0464	0.0077 ± 0.0018	0.9916 ± 0.0475	0.23
	ε-TSVR	0.1830 ± 0.0440	0.0071 ± 0.0017	0.9907 ± 0.0570	0.23
	GWW-ε-TSVR1	0.1263 ± 0.0487	0.0049 ± 0.0019	1.0046 ± 0.0521	0.10
	GWW-ε-TSVR2	0.1221 ± 0.0492	0.0047 ± 0.0019	1.0047 ± 0.0489	0.02
	GWW-ε-TSVR3	0.1201 ± 0.0477	0.0046 ± 0.0018	1.0045 ± 0.0513	0.02
	NWW-ε-TSVR1	0.1221 ± 0.0492	0.0047 ± 0.0019	1.0047 ± 0.0489	0.04
	NWW-ε-TSVR2	0.1217 ± 0.0462	0.0047 ± 0.0018	1.0057 ± 0.0510	0.06
f_4(x)	TSVR	0.8332 ± 0.1269	0.0014 ± 0.0002	1.0046 ± 0.0111	0.21
	ε-TSVR	0.8148 ± 0.1311	0.0014 ± 0.0002	1.0011 ± 0.0122	0.23
	GWW-ε-TSVR1	0.7778 ± 0.1462	0.0013 ± 0.0003	1.0020 ± 0.0100	0.09
	GWW-ε-TSVR2	0.7811 ± 0.1427	0.0013 ± 0.0002	1.0006 ± 0.0095	0.02
	GWW-ε-TSVR3	0.7732 ± 0.1384	0.0013 ± 0.0002	1.0018 ± 0.0098	0.03
	NWW-ε-TSVR1	0.7811 ± 0.1427	0.0013 ± 0.0002	1.0006 ± 0.0095	0.03
	NWW-ε-TSVR2	0.7753 ± 0.1395	0.0013 ± 0.0002	1.0018 ± 0.0097	0.07
f_5(x)	TSVR	0.4367 ± 0.1106	0.0003 ± 0.0001	0.9990 ± 0.0079	0.22
	ε-TSVR	0.4117 ± 0.0929	0.0003 ± 0.0001	0.9997 ± 0.0085	0.22
	GWW-ε-TSVR1	0.3778 ± 0.0591	0.0003 ± 0.0000	1.0007 ± 0.0063	0.09
	GWW-ε-TSVR2	0.3835 ± 0.0673	0.0003 ± 0.0001	1.0006 ± 0.0064	0.02
	GWW-ε-TSVR3	0.3768 ± 0.0647	0.0003 ± 0.0001	1.0006 ± 0.0063	0.03
	NWW-ε-TSVR1	0.3835 ± 0.0673	0.0003 ± 0.0001	1.0006 ± 0.0064	0.04
	NWW-ε-TSVR2	0.3768 ± 0.0637	0.0003 ± 0.0001	1.0006 ± 0.0063	0.07

B. Real Work Benchmark Datasets

For further evaluation, two time series UCI datasets and five non-time series UCI datasets, which are all commonly used in testing machine learning algorithms, are tested. Being PM2.5 Data Dataset is a kind of time series dataset, in which the hourly data set contains the PM2.5 data of US Embassy in Beijing, meanwhile, meteorological data from Beijing Capital International Airport are also included. The number of attributes is seven including dew point, temperature, pressure, combined wind direction, cumulated wind speed (m/s), cumulated hours of snow and cumulated hours of rain. The time period of this dataset is from Jan. 1st, 2010 to Dec 31st, 2014. To test the proposed algorithm, three groups of data were selected from Being PM2.5 Data Dataset. Each group has 2,000 data from about 84 days in different years (from 2011 to 2014). Istanbul stock exchange dataset is another time series UCI datasets, which includes returns of Istanbul stock exchange with seven other international indexes from Jun. 5, 2009 to Feb. 22, 2011. Note that cross validation experiments cannot be adopted in time series, otherwise the feature of time series will be destroyed.

Therefore, the odd number examples are selected for training, and the even number examples are selected for testing. As for non-time series UCI datasets, five UCI datasets: Servo, Wisconsin breast cancer datasets, Auto-Mpg, Concrete compressive strength and Combined cycle power plant are tested. Due to the large number of samples in combined cycle power plant dataset, 1 sample out of every 12 is taken. A more detailed description of these non-time series UCI datasets is shown in Table V. To avoid biased comparisons, the optimal values are computed by using standard ten-fold cross-validation on the training data. It's important to note that these datasets are not time-sequence, so $D \equiv I$. The performance of time series datasets and non-time series datasets listed in Table VI and Table VII respectively show the superiority of the proposed algorithms based on generalized derivative iterative approach and Newton iterative approach. In Table VI, our proposed five iterative WW-ε-TSVR algorithms all derive the smallest SSE and SSE/SST among the popular TSVR and ε-TSVR because of introducing the wavelet transform based weight D. It has been seen that in Table VII our WW-ε-TSVR gives the better performance than TSVR, the similar performance to ε-TSVR because both our method and the ε-TSVR all consider minimizing structural risks. As for the computation time, it is obviously that our method needs less CPU time than others, indicating that our proposed iterative methods are the efficient algorithm for regression.

TABLE V
THE DESCRIPTION FOR UCI DATASETS

Datasets	No. of samples	No. of features
Servo	167	1
Wisconsin B.C.	194	32
Auto-Mpg	398	7
Concrete CS	1030	8
CombinedCPP	798	4

TABLE VI
PERFORMANCE COMPARISON ON TIME SERIES UCI DATASETS

Dataset	Regressor	SSE	SSE/SST	SSR/SST	CPU sec.
Beijing PM2.5 (2011)	TSVR	7.2172	0.4365	0.9117	0.66
	ε-TSVR	6.7495	0.4083	0.8459	0.69
	GWW-ε-TSVR1	6.6496	0.3983	0.8659	0.44
	GWW-ε-TSVR2	6.6592	0.3988	0.8664	0.30
	GWW-ε-TSVR3	6.6499	0.3983	0.8656	0.11
	NWW-ε-TSVR1	6.6592	0.3988	0.8664	0.20
	NWW-ε-TSVR2	6.6504	0.3983	0.8660	0.64
Beijing PM2.5 (2012)	TSVR	7.8362	0.4151	0.8084	0.60
	ε-TSVR	7.5562	0.4003	0.7428	0.57
	GWW-ε-TSVR1	7.4564	0.3923	0.7628	0.37
	GWW-ε-TSVR2	7.4569	0.3923	0.7622	0.07
	GWW-ε-TSVR3	7.4565	0.3923	0.7626	0.10
	NWW-ε-TSVR1	7.4569	0.3923	0.7622	0.20
	NWW-ε-TSVR2	7.4564	0.3923	0.7628	0.22
Beijing PM2.5 (2013)	TSVR	24.7274	0.3797	0.7595	0.71
	ε-TSVR	24.1335	0.3706	0.7117	0.64
	GWW-ε-TSVR1	24.0334	0.3626	0.7217	0.30
	GWW-ε-TSVR2	24.0457	0.3628	0.7220	0.07
	GWW-ε-TSVR3	24.0321	0.3626	0.7216	0.10
	NWW-ε-TSVR1	24.0457	0.3628	0.7220	0.18
	NWW-ε-TSVR2	24.0323	0.3626	0.7217	0.23
Beijing PM2.5 (2014)	TSVR	24.6978	0.5347	0.7271	0.59
	ε-TSVR	23.3219	0.5049	0.6645	0.53
	GWW-ε-TSVR1	23.2226	0.4949	0.6845	0.32
	GWW-ε-TSVR2	23.2614	0.4957	0.6857	0.25
	GWW-ε-TSVR3	23.2232	0.4949	0.6844	0.16
	NWW-ε-TSVR1	23.2614	0.4957	0.6857	0.20
	NWW-ε-TSVR2	23.2236	0.4949	0.6845	0.24
Istanbul Stock Exchange	TSVR	9.4095	0.7771	0.7113	0.19
	ε-TSVR	8.5706	0.7079	0.6400	0.16
	GWW-ε-TSVR1	8.4709	0.6979	0.6600	0.06
	GWW-ε-TSVR2	8.4618	0.6971	0.6582	0.01
	GWW-ε-TSVR3	8.4728	0.6980	0.6602	0.02
	NWW-ε-TSVR1	8.4618	0.6971	0.6582	0.03
	NWW-ε-TSVR2	8.4714	0.6979	0.6601	0.04

TABLE VII
PERFORMANCE COMPARISON ON NON-TIME SERIES UCI DATASETS

Dataset	Regressor	SSE/SST	SSR/SST	CPU sec.
Servo	TSVR	0.1778 ± 0.1069	1.0310 ± 0.3332	0.09
	ε-TSVR	0.1722 ± 0.0914	0.9987 ± 0.3160	0.09
	GWW-ε-TSVR1	0.1722 ± 0.0914	0.9987 ± 0.3161	0.02
	GWW-ε-TSVR2	0.1723 ± 0.0914	0.9988 ± 0.3162	0.01
	GWW-ε-TSVR3	0.1722 ± 0.0914	0.9987 ± 0.3160	0.01
	NWW-ε-TSVR1	0.1723 ± 0.0914	0.9988 ± 0.3162	0.01
	NWW-ε-TSVR2	0.1722 ± 0.0914	0.9987 ± 0.3161	0.02
Wisconsin B.C.	TSVR	1.0208 ± 0.3496	0.4764 ± 0.3204	0.07
	ε-TSVR	0.9634 ± 0.2953	0.4217 ± 0.2731	0.07
	GWW-ε-TSVR1	0.9634 ± 0.2953	0.4217 ± 0.2731	0.02

	GWW-ε-TSVR2	0.9631 ± 0.2948	0.4209 ± 0.2725	0.01
	GWW-ε-TSVR3	0.9634 ± 0.2953	0.4217 ± 0.2731	0.01
	NWW-ε-TSVR1	0.9631 ± 0.2948	0.4209 ± 0.2725	0.01
	NWW-ε-TSVR2	0.9634 ± 0.2953	0.4217 ± 0.2731	0.01
Auto-Mpg	TSVR	0.2640 ± 0.0813	0.8471 ± 0.1540	0.43
	ε-TSVR	0.2490 ± 0.0551	0.8370 ± 0.1473	0.41
	GWW-ε-TSVR1	0.2490 ± 0.0551	0.8370 ± 0.1473	0.14
	GWW-ε-TSVR2	0.2491 ± 0.0551	0.8370 ± 0.1474	0.04
	GWW-ε-TSVR3	0.2490 ± 0.0551	0.8369 ± 0.1472	0.05
	NWW-ε-TSVR1	0.2491 ± 0.0551	0.8370 ± 0.1474	0.08
	NWW-ε-TSVR2	0.2490 ± 0.0551	0.8370 ± 0.1472	0.10
Concrete CS	TSVR	0.1362 ± 0.0342	0.9280 ± 0.0742	3.26
	ε-TSVR	0.1393 ± 0.0313	0.9081 ± 0.0748	3.18
	GWW-ε-TSVR1	0.1393 ± 0.0313	0.9081 ± 0.0748	1.93
	GWW-ε-TSVR2	0.1394 ± 0.0313	0.9080 ± 0.0748	0.40
	GWW-ε-TSVR3	0.1393 ± 0.0313	0.9081 ± 0.0748	0.55
	NWW-ε-TSVR1	0.1394 ± 0.0313	0.9080 ± 0.0748	1.05
	NWW-ε-TSVR2	0.1393 ± 0.0313	0.9081 ± 0.0748	1.32
CombinedCPP	TSVR	0.0712 ± 0.0166	0.9484 ± 0.0489	1.25
	ε-TSVR	0.0668 ± 0.0143	0.9417 ± 0.0493	1.46
	GWW-ε-TSVR1	0.0668 ± 0.0143	0.9417 ± 0.0493	0.76
	GWW-ε-TSVR2	0.0667 ± 0.0143	0.9413 ± 0.0493	0.17
	GWW-ε-TSVR3	0.0668 ± 0.0143	0.9417 ± 0.0493	0.24
	NWW-ε-TSVR1	0.0667 ± 0.0143	0.9413 ± 0.0493	0.43
	NWW-ε-TSVR2	0.0668 ± 0.0143	0.9417 ± 0.0493	0.55

V. Conclusion

In this paper, an improved wavelet transform based weighted ε-TSVR formulation (WW-ε-TSVR) in primal space for the regression of time series is proposed, where the wavelet weight matrix is introduced to give different penalties for the samples located in different places. Although the proposed algorithm is based on ε-TSVR, unlike ε-TSVR, it not only adds a weighted matrix D into QPPs, but also introduces a 'plus' function to convert the algorithm into a pair of unconstrained minimization problems. To further solve these unconstrained minimization problems in primal space, generalized derivative iterative approach and Newton iterative approach depending on differential smooth functions are proposed. Unlike solving the quadratic programming problem by using external optimization toolbox, five iterative algorithms including GWW-ε-TSVR1, GWW-ε-TSVR2, GWW-ε-TSVR3, NWW-ε-TSVR2 and NWW-ε-TSVR3 can be easily coded in MATLAB. The experimental results on several artificial and UCI datasets show that our proposed method gives similar or better generalization performance with TSVR and ε-TSVR, but what is important is that it has greatly reduced the computation time. Moreover, how to select the optimal hyperparameters is a difficult problem and should be addressed in the future.

Appendix A

Algorithm 2: GWW-ε-TSVR2

```
Input:
    · $tol$: the error tolerance for learning accuracy.
    · $itmax$: the maximum number of iterations.
    · $Q_1, G, D, c_1, \varepsilon_1, \alpha$
    · $u_1^{old} = e$: initial approximation of $u_1$.
    · $f_1 = Y + \varepsilon_1 e, r_1 = Q_1^{-1} G^T$
    · $k = 0$
Output:
    The optimal solution $u_1$.
Process:
    do {
    · $d_1^{p_2} = diag((1 + \exp(-\alpha(G u_1^{old} - f_1)))^{-1})$
    · Solve $u_1^{new} = r_1(DY - c_1 d_1^{p_2} e)$
    · $err = u_1^{new} - u_1^{old}$
    · $u_1^{old} = u_1^{new}$
    · $k = k + 1$
    } while ($|err| > tol$ and $k < itmax$)
```

Algorithm 3: GWW-ε-TSVR3

```
Input:
    · $tol$: the error tolerance for learning accuracy.
    · $itmax$: the maximum number of iterations.
    · $Q_1, G, D, c_1, \varepsilon_1, \alpha$
    · $u_1^{old} = e$: initial approximation of $u_1$.
    · $f_1 = Y + \varepsilon_1 e, r_1 = Q_1^{-1} G^T, z_1 = DY - \frac{c_1}{2} e$
    · $k = 0$
Output:
    The optimal solution $u_1$.
Process:
    do {
    · $d_1^{p_3} = diag(((G u_1^{old} - f_1).\hat{}2 + 4\alpha^2)^{-\frac{1}{2}})$
    · Solve $u_1^{new} = r_1[z_1 - \frac{1}{2} c_1 d_1^{p_3}(G u_1^{old} - f_1)]$
    · $err = u_1^{new} - u_1^{old}$
    · $u_1^{old} = u_1^{new}$
    · $k = k + 1$
    } while ($|err| > tol$ and $k < itmax$)
```

Algorithm 4: NWW-ε-TSVR2

```
Input:
    . $tol$: the error tolerance for learning accuracy.
    · $itmax$: the maximum number of iterations.
    · $Q_1, G, D, c_1, \varepsilon_1, \alpha$
    · $u_1^{old} = e$: initial approximation of $u_1$.
    · $f_1 = Y + \varepsilon_1 e$
    · $k = 0$
Output:
    The optimal solution $u_1$.
Process:
    do {
    · $d_1^{p_2} = diag((1+\exp(-\alpha(Gu_1^{old} - f_1)))^{-1})$
    · $d_3^{p_2} = diag\left(\frac{exp(-\alpha(Gu_1^{old}-f_1))}{(1+exp(-\alpha(Gu_1^{old}-f_1)))^2}\right)$
    · $r = Q_1 + \alpha c_1 G^T d_3^{p_2} e$
    · $s = Q_1 u_1^{old} - G^T DY + c_1 G^T d_1^{p_2} e$
    · Solve $u_1^{new} = u_1^{old} - r^{-1} s$
    · $err = u_1^{new} - u_1^{old}$
    · $u_1^{old} = u_1^{new}$
    · $k = k + 1$
    } while ($|err| > tol$ and $k < itmax$)
```

```
Algorithm 5: NWW-ε-TSVR3
Input:
    . tol: the error tolerance for learning accuracy.
    · itmax: the maximum number of iterations.
    · $Q_1, G, D, c_1, \varepsilon_1, \alpha$
    · $u_1^{old} = e$: initial approximation of $u_1$.
    · $f_1 = Y + \varepsilon_1 e$
    · $k = 0$
Output:
    The optimal solution $u_1$.
Process:
    · do {
    · $d_1^{p_3} = diag(\frac{1}{\sqrt{(Gu_1^{old}-f_1).\hat{}2+4\alpha^2}})$
    · $d_3^{p_3} = diag(\frac{1}{\left(\sqrt{(Gu_1^{old}-f_1).^2+4\alpha^2}\right)^3})$
    · $r = Q_1 + 2\alpha^2 c_1 G^T d_3^{p_3} G$
    · $s = Q_1 u_1^{old} - G^T DY + \frac{1}{2} c_1 G^T (e + d_1^{p_3}(Gu_1^{old} - f_1))$
    · Solve $u_1^{new} = u_1^{old} - r^{-1} s$
    · $err = u_1^{new} - u_1^{old}$
    · $u_1^{old} = u_1^{new}$
    · $k = k + 1$
    } while ($|err| > tol$ and $k < itmax$)
```

REFERENCES

[1] V. Vapnik, *The Natural of Statistical Learning Theory*. New York: Springer, 1995, 521-576.

[2] R. Debnath, M. Muramatsu, and H. Takahashi, "An efficient support vector machine learning method with second-order cone programming for large-scale problems", *Applied Intelligence*, vol.23, no.3, pp. 219-239, 2005.

[3] F. Nie, H. Huang, X. Cai, et al. "Efficient and robust feature selection via joint ℓ2, 1-norms minimization", in *International Conference on Neural Information Processing Systems,* Curran Associates Inc., 2010, pp.1813-1821.

[4] M. Claesen, F. De Smet, J. Suykens, et al., "A robust ensemble approach to learn from positive and unlabeled data using SVM base models", *Neurocomputing*, vol. 160, pp.73-74, 2015.

[5] X Cai, X Nan, B Gao, "Oxygen supply prediction model based on IWO-SVR in bio-oxidation pretreatment", *Engineering Letters*, vol.23, no.3, pp.173-179, 2015.

[6] C. Burges, "A tutorial on support vector machines for pattern recognition", *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121-167, 1998.

[7] B. Scholkopf, A. Smola, R. Williamson, et al., "New support vector algorithms", *Neural Computation*, vol.12, no. 5, pp. 1207-1245, 2000.

[8] X.J. Peng, "TSVR: An efficient twin support vector machine for regression", *Neural Networks*, vol. 23, no. 3, pp. 365-372, 2010.

[9] X.J. Peng, "Efficient twin parametric insensitive support vector regression model", *Neurocomputing,* vol. 79, pp.26-38, 2012.

[10] X.J. Peng, D. Xu, J.D. Shen, "A twin projection support vector machine for data regression", *Neurocomputing*, vol.138, pp. 131-141, 2014.

[11] Jayadeva, R. Khemchandani and S. Chandra, "Twin support vector machines for pattern classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 5, pp. 905-910, 2007.

[12] R. Khemchandani, K. Goyal and S. Chandra, "TWSVR: Regression via twin support vector machine", *Neural Networks*, vol.74, pp.14-21, 2016.

[13] Y.H. Shao, C.H. Zhang, Z. M. Yang, et al., "An ε-twin support vector machine for regression," *Neural Comput. Appl.*, vol. 23, no.1, pp.175-185, 2013.

[14] R. Rastogi, P. Anand and S. Chandra, "A ν-twin support vector machine based regression with automatic accuracy control," *Applied Intelligence,* vol. 46, no.3, pp.670-683, 2017.

[15] Y. Xu, X. Li, X. Pan, et al., "Asymmetric ν-twin support vector regression", *Neural Computing & Applications*, vol. 2, pp.1-16, 2017.

[16] Y. Xu; L. Wang, "A weighted twin support vector regression", *Knowl. -Based Syst.*, vol. 33, pp. 92-101,2012.

[17] O. Matei; P.C. Pop; H. Vălean, "Optical character recognition in real environments using neural networks and k-nearest neighbor", *Applied Intelligence*, vol. 39, no. 4, pp. 739-748, 2013.

[18] Y. Xu, L. Wang, "K-nearest neighbor-based weighted twin support vector regression", *Applied Intelligence,* vol.41, no.1, pp.299-309, 2014.

[19] N. Parastalooi, A. Amiri, P. Aliherdari, "Modified twin support vector regression", *Neurocomputing*, vol. 211, pp. 84-97, 2016.

[20] Y. Ye, L. Bai, X. Hua, et al., "Weighted Lagrange ε-twin support vector regression", *Neurocomputing*, vol. 197, pp. 53-68 , 2016.

[21] Chapelle O, "Training a support vector machine in primal", *Neural Computation*, vol.19, no.5, pp.1155–1178, 2007.

[22] X.J. Peng, "Primal twin support vector regression and its sparse approximation", *Neurocomputing*, vol. 73, no.16-18, pp. 2846-2858, 2010.

[23] S. Balasundaram, D. Gupta, "Training Lagrangian twin support vector regression via uncontrained convex minimization", *Knowl. -Based Syst.*, vol. 59, pp. 85-96, 2014.

[24] D. Gupta, "Training primal K-nearest neighbor based weighted twin support vector regression via unconstrained convex minimization", *Applied Intelligence,* vol. 47, no. 3, pp.962-991, 2017.

[25] S. Balasundaram, Y. Meena, "Training primal twin support vector regression via unconstrained convex minimization", *Applied Intelligence,* vol. 44, no. 4, pp. 931-955, 2016.

[26] I. Daubechies, *Ten lectures on wavelets*. Philapdelphia: Society for Industrial and Applied Mathematics, 1992.

[27] Y.J. Lee, O.L. Mangasarian, "SSVM: A smooth support vector machine for classification". *Computational Optimization and Applications*, vol. 20, no.1, pp. 5-22, 2001.

[28] X. Chen, J. Yang, J. Liang, Q. Ye, "Smooth twin support vector regression", *Neural Computing & Applications*, vol. 21, no. 3, pp.505-513, 2012.

[29] O.L. Mangasarian, "A finite Newton method for classification", *Optimization Methods & Software*, vol. 17, no. 5, pp. 913-929, 2002.

[30] G. Fung, O.L. Mangasarian, "Finite Newton method for Lagrangian support vector machine", *Neurocomputing,* vol. 55, no. 1, pp. 39-55, 2003.

[31] C. L. Blake and C. J. Merz. (1998). UCI Repository for Machine Learning Databases, Department of Information and Computer Sciences, University of California, Irvine. [Online]. Available: http://www.ics.uci.edu/mlearn/MLRepository. html

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.

XINYU OUYANG was born in Hunan Province, China, in 1974. He 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 he received Ph.D. degree in Control Theory and Control Engineering from Dalian University of Technology in 2014.

Since 1998 he has been with University of Science and Technology Liaoning, and he is currently a Professor of Automatic Control. He published over 30 scientific papers in diverse fields, some of which were indexed by SCI and EI. His research interests include artificial intelligence, machine learning and digital image 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.

ZIHUI ZANG was born in China in 1994, received her B.S. degree from University of Science and Technology Liaoning, China in 2016.

She is currently a graduate student of University of Science and Technology Liaoning. Her research interests include pattern recognition, machine learning.