International Journal of Innovative
Computing, Information and Control
Volume **15**, Number **6**, December **2019**

 ISSN 1349-4198
pp. **2275–2288**

LEAST SQUARES TWIN PROJECTION SUPPORT VECTOR REGRESSION

Binjie Gu*, Geliang Shen, Feng Pan and Hao Chen

Key Laboratory of Advanced Process Control for Light Industry, Ministry of Education
Jiangnan University
No. 1800, Lihu Road, Wuxi 214122, P. R. China
*Corresponding author: gubinjie1980@126.com

Received February 2019; revised July 2019

Abstract. *In order to further promote the predictive performance of least squares twin support vector regression (LSTSVR), we proposed a least squares twin projection support vector regression (LSTPSVR). The variance of the projected data is integrated into the objective function of the primal problems as a new term. Minimizing the variance ensures that we can obtain a suitable projection axis, and the empirical covariance and correlation coefficients of the input and output data are embedded into the solution of linear equations automatically. Extensive experimental results on several benchmark datasets and artificial test function show that LSTPSVR can obtain better predictive performance than other state-of-the-art algorithms. In addition, the proposed algorithm is stable.*
Keywords: Twin support vector regression (TSVR), Least squares, Feature space, Projection, Variance

1. **Introduction.** Support vector machine (SVM) is a machine learning algorithm based on the principle of structural risk minimization [1-6], and SVM has been successfully applied in solving the high dimensional and local minimum problems of regression.

The ε-support vector regression (ε-SVR) is one of the most commonly used regression algorithms. The aim of ε-SVR is to find a regression function that makes the error acceptable within allowable range. This regression function can be obtained by solving large-size quadratic programming problems (QPPs) [7-11]. Unfortunately, it is difficult to select a suitable parameter for ε-SVR. Hence, Schölkopf et al. proposed a novel ν-SVR. In ν-SVR, a new proportion parameter ν with $0 \leq \nu \leq 1$ is introduced into the primal optimization problem, and ν can be utilized to control the number of support vectors and errors [12,13]. Therefore, it is easier to tune parameter ν than ε-SVR [14]. However, the ν-SVR works on the premise of data with uniformly-distributed noise, and this assumption is not always true in reality. In order to address this problem, Hao proposed a parametric-insensitive ν-SVR (par-ν-SVR) algorithm, whose aim is to find an adjustable parameter insensitive zone [15]. The error of data in the parameter insensitive zone can be ignored, and only the points outside this zone contribute to the cost function. Subsequently, Peng and Xu proposed a projection support vector regression, which further solves the over-fitting problem of par-ν-SVR for many real-world applications [16].

Recently, Jayadeva et al. proposed a novel machine learning algorithm, named twin support vector machine (TSVM), which converts a large-size QPPs into two smaller-size QPPs [17]. Therefore, the training time cost by TSVM is approximately a quarter of corresponding SVM. To date, many extensions of TSVM have been developed. Peng

DOI: 10.24507/ijicic.15.06.2275

extended the idea of TSVM to regression and he proposed an efficient twin support vector regression (TSVR) [18]. The experimental results demonstrate that the prediction performance of TSVR is better than SVR. Then, in order to improve the generalization ability of TSVR, Shao et al. proposed an ε-TSVR which implements the structural risk minimization principle instead of the empirical risk minimization principle by introducing a regularization term into the primal problems of their ε-TSVR [19]. Balasundaram and Tanveer designed a novel iterative Lagrangian TSVR. The main advantage of the iterative Lagrangian TSVR is that the unknown regressor can be obtained using an extremely simple and linearly convergent iterative algorithm rather than solving QPPs as in SVR and TSVR [20]. Considering that the local information among samples are not exploited in TSVR, Xu and Wang proposed a K-nearest neighbour-based weighted TSVR (KNNWTSVR). Experimental results show that KNNWTSVR not only yields lower prediction error but also costs lower running time in comparison with other algorithms [21]. Later, Peng proposed an efficient twin parametric insensitive SVR (TPISVR), and the results of simulation on several benchmark datasets show that TPISVR is better than TSVR [22]. To successfully address the over-fitting problem of TPISVR, Peng et al. employed projection method and they proposed a twin projection support vector regression (TPSVR) [23]. The embedded data prior structural information by projection in TPSVR improves the prediction performance. However, the disadvantages of existing TSVR should not be ignored. Firstly, TSVR loses sparsity [19-23]. Secondly, the implementation of the empirical risk minimization principle in TSVR declines the generalization ability and lowers the prediction performance [24]. Finally, the same penalties are given to the data samples in TSVR. In fact, samples at different positions have different influences on the bound functions [25].

Although the training time of TSVR and its extensions are much shorter than SVR, it is still time-consuming when solving large-scale optimization problems. Kumar and Gopal exploited least squares TSVM (LSTSVM) for pattern classification; the least squares method is adopted to solve the primal problems [26]. LSTSVM only needs to solve two linear equations in primal space. Hence, the training time of LSTSVM is greatly shortened. Later, Huang et al. generalized the idea of LSTSVM to regression and they proposed a least squares TSVR (LSTSVR) [27]. Then, Ding and Huang proposed least squares twin parametric insensitive support vector regression (LSTPISVR) which further improves the prediction performance [28]. However, due to the introduction of parametric insensitive term, the resulting linear equation is more complicated than LSTSVR, and the training time will increase slightly. In addition, LSTPISVR may also suffer over-fitting problem, and it is easily affected by noise. Huang et al. proposed a sparse method for LSTSVR. The proposed method yields sparse solutions and costs fewer training time without sacrificing regression accuracy compared with TSVR and LSTSVR [29]. Zhang designed a novel algorithm named p-norm LSTSVR (PLSTSVR). By adjusting the parameter p, PLSTSVR can improve regression accuracy [30]. Experiments carried out on several standard UCI datasets and synthetic datasets show the feasibility and effectiveness of PLSTSVR.

Based on the work of Peng et al. [16,18,23], we investigate a least squares twin projection support vector regression (LSTPSVR). In LSTPSVR, we attempt to integrate the variance of the projected data into the objective function of the primal problems as a new term. During the process of solving linear equations, the empirical covariance and correlation coefficients of input and output data are embedded into the final linear matrices. This means that the data prior structural information is considered automatically. In order to evaluate the performance of LSTPSVR, we conduct extensive experiments

on benchmark datasets and artificial test function. Compared with LSTSVR and its extensions, LSTPSVR can obtain better prediction performance. Moreover, the proposed LSTPSVR algorithm is stable when suitable parameters are selected.

The remainder of this paper is organized as follows. Section 2 briefly introduces the least squares twin support vector regression in linear and nonlinear cases. Section 3 elaborates our LSTPSVR algorithm. Section 4 presents the results and analysis of simulation experiments. Section 5 gives the conclusion of this paper.

2. **Least Squares Twin Support Vector Regression (LSTSVR).** Given a training sample set $S = \{S_1, \ldots, S_n\} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, where $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ represent the input data of m attributions and $y_1, y_2, \ldots, y_n$ is the corresponding output data, n is the size of the training sample set. For simplicity, we denote $\boldsymbol{A} = [\boldsymbol{x}_1; \boldsymbol{x}_2; \ldots; \boldsymbol{x}_n] \in R^{n \times m}$ and $\boldsymbol{Y} = [y_1, y_2, \ldots, y_n]^{\mathrm{T}} \in R^n$ as the input sample matrix and output vector, respectively.

In linear case, the primal problems of LSTSVR are expressed as follows:

$$\begin{aligned} \min_{\boldsymbol{\omega}_1, b_1, \boldsymbol{\xi}} \quad & \frac{1}{2} \left\| \boldsymbol{Y} - \varepsilon_1 \boldsymbol{e} - (\boldsymbol{A}\boldsymbol{\omega}_1 + b_1 \boldsymbol{e}) \right\|^2 + \frac{C_1}{2} \boldsymbol{\xi}^{\mathrm{T}} \boldsymbol{\xi} \\ \text{s.t.} \quad & \boldsymbol{Y} - (\boldsymbol{A}\boldsymbol{\omega}_1 + b_1 \boldsymbol{e}) = \varepsilon_1 \boldsymbol{e} - \boldsymbol{\xi}, \quad \boldsymbol{\xi} \geq \boldsymbol{0} \end{aligned} \tag{1}$$

and

$$\begin{aligned} \min_{\boldsymbol{\omega}_2, b_2, \boldsymbol{\eta}} \quad & \frac{1}{2} \left\| \boldsymbol{Y} + \varepsilon_2 \boldsymbol{e} - (\boldsymbol{A}\boldsymbol{\omega}_2 + b_2 \boldsymbol{e}) \right\|^2 + \frac{C_2}{2} \boldsymbol{\eta}^{\mathrm{T}} \boldsymbol{\eta} \\ \text{s.t.} \quad & (\boldsymbol{A}\boldsymbol{\omega}_2 + b_2 \boldsymbol{e}) - \boldsymbol{Y} = \varepsilon_2 \boldsymbol{e} - \boldsymbol{\eta}, \quad \boldsymbol{\eta} \geq \boldsymbol{0} \end{aligned} \tag{2}$$

where, $\boldsymbol{\omega}_1, \boldsymbol{\omega}_2 \subseteq \boldsymbol{R}^m$, $b_1, b_2 \subseteq R$, $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ are non-negative slack vectors, $C_1, C_2 > 0$ are penalty factors, $\varepsilon_1, \varepsilon_2 > 0$ are insensitive loss parameters, $\boldsymbol{e}$ is a unit column vector of proper dimensions.

Equations (1) and (2) indicate that the inequality constraints can be replaced by equality constraints in LSTSVR compared with conventional TSVR. Therefore, the training time cost by LSTSVR is much shorter than TSVR.

By substituting equality constraints into the objective function in Equation (1), we can obtain the following Lagrangian function:

$$L(\boldsymbol{\omega}_1, b_1, \boldsymbol{\xi}_1) = \frac{1}{2} ||\boldsymbol{Y} - \varepsilon_1 \boldsymbol{e} - (\boldsymbol{A}\boldsymbol{\omega}_1 + b_1 \boldsymbol{e})||^2 + \frac{C_1}{2} ||(\boldsymbol{A}\boldsymbol{\omega}_1 + b_1 \boldsymbol{e}) - \boldsymbol{Y} + \varepsilon_1 \boldsymbol{e}||^2 \tag{3}$$

Then, setting the partial derivatives of $\boldsymbol{\omega}_1$ and b_1 in Equation (3) to zeros, we have:

$$\frac{\partial L(\boldsymbol{\omega}_1, b_1, \boldsymbol{\xi}_1)}{\partial \boldsymbol{\omega}_1} = -\boldsymbol{A}^{\mathrm{T}} [\boldsymbol{Y} - \varepsilon_1 \boldsymbol{e} - (\boldsymbol{A}\boldsymbol{\omega}_1 + b_1 \boldsymbol{e})] + C_1 \boldsymbol{A}^{\mathrm{T}} [(\boldsymbol{A}\boldsymbol{\omega}_1 + b_1 \boldsymbol{e}) - \boldsymbol{Y} + \varepsilon_1 \boldsymbol{e}] = \boldsymbol{0} \tag{4}$$

$$\frac{\partial L(\boldsymbol{\omega}_1, b_1, \boldsymbol{\xi}_1)}{\partial b_1} = -\boldsymbol{e}^{\mathrm{T}} [\boldsymbol{Y} - \varepsilon_1 \boldsymbol{e} - (\boldsymbol{A}\boldsymbol{\omega}_1 + b_1 \boldsymbol{e})] + C_1 \boldsymbol{e}^{\mathrm{T}} [(\boldsymbol{A}\boldsymbol{\omega}_1 + b_1 \boldsymbol{e}) - \boldsymbol{Y} + \varepsilon_1 \boldsymbol{e}] = 0 \tag{5}$$

Further, Equations (4) and (5) can be written in the following matrix form:

$$\begin{aligned} & - \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{e} \end{bmatrix}^{\mathrm{T}} \left((\boldsymbol{Y} - \varepsilon_1 \boldsymbol{e}) - \begin{bmatrix} \boldsymbol{A} & \boldsymbol{e} \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_1 \\ b_1 \end{bmatrix} \right) \\ & + C_1 \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{e} \end{bmatrix}^{\mathrm{T}} \left(\begin{bmatrix} \boldsymbol{A} & \boldsymbol{e} \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_1 \\ b_1 \end{bmatrix} - (\boldsymbol{Y} - \varepsilon_1 \boldsymbol{e}) \right) = \boldsymbol{0} \end{aligned} \tag{6}$$

Defining $\boldsymbol{G} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{e} \end{bmatrix}$, $\boldsymbol{f} = \boldsymbol{Y} - \varepsilon_1 \boldsymbol{e}$ and $\boldsymbol{u}_1 = \begin{bmatrix} \boldsymbol{\omega}_1^{\mathrm{T}} & b_1 \end{bmatrix}^{\mathrm{T}}$, Equation (6) changes into:

$$-\boldsymbol{G}^{\mathrm{T}} (\boldsymbol{f} - \boldsymbol{G}\boldsymbol{u}_1) + C_1 \boldsymbol{G}^{\mathrm{T}} (\boldsymbol{G}\boldsymbol{u}_1 - \boldsymbol{f}) = \boldsymbol{0} \tag{7}$$

Solving Equation (7), we can obtain:

$$\boldsymbol{u}_1 = \left(\boldsymbol{G}^{\mathrm{T}} \boldsymbol{G} \right)^{-1} \boldsymbol{G}^{\mathrm{T}} \boldsymbol{f} \tag{8}$$

In practice, a regularization term $C_3\boldsymbol{I}$ is introduced to avoid matrix singularity, where $C_3 = 0.01$ and $\boldsymbol{I}$ is an identity matrix which is of the same size as $\boldsymbol{G}^{\mathrm{T}}\boldsymbol{G}$. Then, Equation (8) can be rewritten as:

$$\boldsymbol{u}_1 = \left(\boldsymbol{G}^{\mathrm{T}}\boldsymbol{G} + C_3\boldsymbol{I}\right)^{-1}\mathbf{G}^{\mathrm{T}}\boldsymbol{f} \tag{9}$$

Similarly, we can obtain:

$$\boldsymbol{u}_2 = \left(\boldsymbol{G}^{\mathrm{T}}\boldsymbol{G} + C_4\boldsymbol{I}\right)^{-1}\mathbf{G}^{\mathrm{T}}\boldsymbol{h} \tag{10}$$

where $C_4 = 0.01$, $\boldsymbol{h} = \boldsymbol{Y} + \varepsilon_2\boldsymbol{e}$ and $\boldsymbol{u}_2 = \left[\begin{array}{cc} \boldsymbol{\omega}_2^{\mathrm{T}} & b_2 \end{array}\right]^{\mathrm{T}}$.

Finally, the regression function in linear case can be constructed as follows:

$$f(\boldsymbol{x}) = \frac{1}{2}(\boldsymbol{\omega}_1 + \boldsymbol{\omega}_2)^{\mathrm{T}}\boldsymbol{x} + \frac{1}{2}(b_1 + b_2) \tag{11}$$

As for nonlinear case, by introducing the kernel function $\boldsymbol{K}\left(\boldsymbol{A}, \boldsymbol{A}^{\mathrm{T}}\right)$, we can map the training samples $\boldsymbol{x}_i$ into a high-dimensional reproducing kernel Hilbert space (RKHS), the primal problems change into:

$$\begin{array}{rl} \min\limits_{\boldsymbol{\omega}_1, b_1, \boldsymbol{\xi}} & \dfrac{1}{2}\left\|\boldsymbol{Y} - \varepsilon_1\boldsymbol{e} - \left(\boldsymbol{K}\left(\boldsymbol{A}, \boldsymbol{A}^{\mathrm{T}}\right)\boldsymbol{\omega}_1 + b_1\boldsymbol{e}\right)\right\|^2 + \dfrac{C_1}{2}\boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{\xi} \\ \text{s.t.} & \boldsymbol{Y} - \left(\boldsymbol{K}\left(\boldsymbol{A}, \boldsymbol{A}^{\mathrm{T}}\right)\boldsymbol{\omega}_1 + b_1\boldsymbol{e}\right) = \varepsilon_1\boldsymbol{e} - \boldsymbol{\xi}, \quad \boldsymbol{\xi} \geq \boldsymbol{0} \end{array} \tag{12}$$

and

$$\begin{array}{rl} \min\limits_{\boldsymbol{\omega}_2, b_2, \boldsymbol{\eta}} & \dfrac{1}{2}\left\|\boldsymbol{Y} + \varepsilon_2\boldsymbol{e} - \left(\boldsymbol{K}\left(\boldsymbol{A}, \boldsymbol{A}^{\mathrm{T}}\right)\boldsymbol{\omega}_2 + b_2\boldsymbol{e}\right)\right\|^2 + \dfrac{C_2}{2}\boldsymbol{\eta}^{\mathrm{T}}\boldsymbol{\eta} \\ \text{s.t.} & \left(\boldsymbol{K}\left(\boldsymbol{A}, \boldsymbol{A}^{\mathrm{T}}\right)\boldsymbol{\omega}_2 + b_2\boldsymbol{e}\right) - \boldsymbol{Y} = \varepsilon_2\boldsymbol{e} - \boldsymbol{\eta}, \quad \boldsymbol{\eta} \geq \boldsymbol{0} \end{array} \tag{13}$$

The optimal solutions of Equations (12) and (13) are:

$$\boldsymbol{\theta}_1 = \left(\boldsymbol{H}^{\mathrm{T}}\boldsymbol{H} + C_3\boldsymbol{I}\right)^{-1}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{f} \tag{14}$$

$$\boldsymbol{\theta}_2 = \left(\boldsymbol{H}^{\mathrm{T}}\boldsymbol{H} + C_4\boldsymbol{I}\right)^{-1}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{h} \tag{15}$$

where $\boldsymbol{\theta}_1 = \left[\begin{array}{cc} \boldsymbol{\omega}_1^{\mathrm{T}} & b_1 \end{array}\right]^{\mathrm{T}}$, $\boldsymbol{\theta}_2 = \left[\begin{array}{cc} \boldsymbol{\omega}_2^{\mathrm{T}} & b_2 \end{array}\right]^{\mathrm{T}}$ and $\boldsymbol{H} = \left[\begin{array}{cc} \boldsymbol{K}\left(\boldsymbol{A}, \boldsymbol{A}^{\mathrm{T}}\right) & \boldsymbol{e} \end{array}\right]$.

Then, the regression function in nonlinear case can be constructed as follows:

$$f(\boldsymbol{x}) = \frac{1}{2}(\boldsymbol{\omega}_1 + \boldsymbol{\omega}_2)^{\mathrm{T}}\boldsymbol{K}\left(\boldsymbol{x}, \boldsymbol{A}^{\mathrm{T}}\right) + \frac{1}{2}(b_1 + b_2) \tag{16}$$

Unfortunately, the prediction performance of existing LSTSVR algorithm is not always the best due to the fact that the data prior structural information is ignored during the learning process. Hence, in the next section, we will attempt to use projection method to further promote the prediction performance of LSTSVR.

3. Least Squares Twin Projection Support Vector Regression (LSTPSVR).

In this section, we first discuss the purpose of introducing projection, and then we present the LSTPSVR algorithm in linear and nonlinear cases.

3.1. Projection.

The purpose of introducing projection is to find a projection axis for the up- and down- bound respectively. Minimizing the variance of the projected data can make the projected zone as small as possible. This means that the original data can be well-fitted by regression function.

Denote $\hat{\boldsymbol{\omega}}_k$, $k = 1, 2$ as the projection axis, the projected data can be expressed as $\hat{\boldsymbol{\omega}}_k^{\mathrm{T}}\boldsymbol{S}_i$, where $\boldsymbol{S}_i = (\boldsymbol{x}_i; y_i)$, $i = 1, \ldots, n$. As for regression, minimizing the variance implies that the predicted value will be more accurate.

The variance of the projected data is equivalent to optimize the following problem:

$$\begin{aligned}\min \quad & \frac{1}{2n}\sum_{i=1}^{n}\left(\hat{\boldsymbol{\omega}}_k^{\mathrm{T}}\boldsymbol{S}_i-\hat{\boldsymbol{\omega}}_k^{\mathrm{T}}\boldsymbol{\mu}_S\right)^2 \\ &=\frac{1}{2n}\sum_{i=1}^{n}\hat{\boldsymbol{\omega}}_k^{\mathrm{T}}(\boldsymbol{S}_i-\boldsymbol{\mu}_S)(\boldsymbol{S}_i-\boldsymbol{\mu}_S)^{\mathrm{T}}\hat{\boldsymbol{\omega}}_k \\ &=\frac{1}{2}\hat{\boldsymbol{\omega}}_k^{\mathrm{T}}\Sigma_S\hat{\boldsymbol{\omega}}_k,\quad k=1,2\end{aligned} \tag{17}$$

where $\boldsymbol{\mu}_S$ is the centroid point of $\boldsymbol{S}_i$ and $\Sigma_S=\frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{S}_i-\boldsymbol{\mu}_S)(\boldsymbol{S}_i-\boldsymbol{\mu}_S)^{\mathrm{T}}$ is the covariance matrix of $\boldsymbol{S}_i$.

Considering that the projection axis $\hat{\boldsymbol{\omega}}_k$ can be treated as normal vector of the insensitive up- and down- bound functions, $\hat{\boldsymbol{\omega}}_k$ can be denoted as $\hat{\boldsymbol{\omega}}_k=\begin{bmatrix}\boldsymbol{\omega}_k^{\mathrm{T}} & w_y\end{bmatrix}^{\mathrm{T}}$, where $w_y=-1$. Therefore, Equation (17) can be rewritten as:

$$\begin{aligned}\min \quad & \frac{1}{2}\hat{\boldsymbol{\omega}}_k^{\mathrm{T}}\Sigma_S\hat{\boldsymbol{\omega}}_k \\ &=\frac{1}{2}\begin{bmatrix}\boldsymbol{\omega}_k^{\mathrm{T}} & w_y\end{bmatrix}\begin{bmatrix}\Sigma_x & \Sigma_{xy}\\ \Sigma_{yx} & \Sigma_y\end{bmatrix}\begin{bmatrix}\boldsymbol{\omega}_k^{\mathrm{T}}\\ w_y\end{bmatrix} \\ &=\frac{1}{2}\left(\boldsymbol{\omega}_k^{\mathrm{T}}\Sigma_x\boldsymbol{\omega}_k+\Sigma_y\right)-\boldsymbol{\omega}_k^{\mathrm{T}}\Sigma_{xy}\end{aligned} \tag{18}$$

where, Σ_x and Σ_y are the empirical covariance matrices of input/output data, Σ_{xy} and Σ_{yx} are the empirical correlation coefficient matrices between input and output data, respectively. They can be calculated by the following equations:

$$\Sigma_x=\frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i-\boldsymbol{\mu}_x)(\boldsymbol{x}_i-\boldsymbol{\mu}_x)^{\mathrm{T}}=\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i\boldsymbol{x}_i^{\mathrm{T}}-\boldsymbol{\mu}_x\boldsymbol{\mu}_x^{\mathrm{T}} \tag{19}$$

$$\Sigma_y=\frac{1}{n}\sum_{i=1}^{n}(y_i-\mu_y)(y_i-\mu_y)^{\mathrm{T}}=\frac{1}{n}\sum_{i=1}^{n}y_i^2-\mu_y^2 \tag{20}$$

$$\Sigma_{xy}=\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_iy_i-\boldsymbol{\mu}_x\mu_y \tag{21}$$

$$\Sigma_{yx}=\Sigma_{xy}^{\mathrm{T}} \tag{22}$$

where $\boldsymbol{\mu}_x$ and μ_y are the centroid points of input and output data, respectively.

3.2. **Linear LSTPSVR.** The aim of LSTPSVR is to determine suitable insensitive up- and down- bound functions by solving two QPPs with equality constraints, and searching for a proper projection axis for each QPPs.

Similar to the work of Peng [16], we introduce the variance of projected data for each bound function. The objective functions of LSTSVR in Equations (1) and (2) change into the following forms:

$$\begin{aligned}\min_{\boldsymbol{\omega}_1,b_1,\boldsymbol{\xi}} \quad & \frac{1}{2}||\boldsymbol{Y}-\varepsilon_1\boldsymbol{e}-(\boldsymbol{A}\boldsymbol{\omega}_1+b_1\boldsymbol{e})||^2+\frac{\lambda_1}{2}\hat{\boldsymbol{\omega}}_1^{\mathrm{T}}\Sigma_S\hat{\boldsymbol{\omega}}_1+\frac{C_1}{2}\boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{\xi} \\ \text{s.t.} \quad & \boldsymbol{Y}-(\boldsymbol{A}\boldsymbol{\omega}_1+b_1\boldsymbol{e})=\varepsilon_1\boldsymbol{e}-\boldsymbol{\xi},\quad \boldsymbol{\xi}\geq\boldsymbol{0}\end{aligned} \tag{23}$$

and

$$\begin{aligned}\min_{\boldsymbol{\omega}_2,b_2,\boldsymbol{\eta}} \quad & \frac{1}{2}||\boldsymbol{Y}+\varepsilon_2\boldsymbol{e}-(\boldsymbol{A}\boldsymbol{\omega}_2+b_2\boldsymbol{e})||^2+\frac{\lambda_2}{2}\hat{\boldsymbol{\omega}}_2^{\mathrm{T}}\Sigma_S\hat{\boldsymbol{\omega}}_2+\frac{C_2}{2}\boldsymbol{\eta}^{\mathrm{T}}\boldsymbol{\eta} \\ \text{s.t.} \quad & (\boldsymbol{A}\boldsymbol{\omega}_2+b_2\boldsymbol{e})-\boldsymbol{Y}=\varepsilon_2\boldsymbol{e}-\boldsymbol{\eta},\quad \boldsymbol{\eta}\geq\boldsymbol{0}\end{aligned} \tag{24}$$

where $\lambda_1,\lambda_2>0$ are penalty factors, which are utilized to control the weight of the second term.

Now, we explain the influence of each term on regression in Equation (23). The first term is to minimize the sum of squared distances from the shifted function $y = \boldsymbol{\omega}_1\boldsymbol{x}+b_1+\varepsilon_1$ to the training data points. This means that minimizing the first term will make the regression function $f_1(\boldsymbol{x}) = \boldsymbol{\omega}_1^{\mathrm{T}}\boldsymbol{x} + b_1$ fit ε_1-insensitive down-bound function well; the second term is the variance of projected data which represents the data prior structural information, it is utilized to depict data distributions and reduce prediction error; the third term is the square of 2-norm of slack vectors $\boldsymbol{\xi}$ with penalty parameter C_1, which ensures that the inequality constraints $\boldsymbol{\xi} \geq 0$ in the primal twin support vector regression problem are redundant. Furthermore, we can easily introduce least squares method by this term. The three terms in Equation (24) have the same explanation.

Substituting the equality constraints into the objective function, we can obtain the following Lagrangian function of Equation (23):

$$\begin{aligned} L(\boldsymbol{\omega}_1, b_1, \boldsymbol{\xi}) = \frac{1}{2}||\boldsymbol{Y} - \varepsilon_1\boldsymbol{e} - (\boldsymbol{A}\boldsymbol{\omega}_1 + b_1\boldsymbol{e})||^2 + \frac{\lambda_1}{2}\hat{\boldsymbol{\omega}}_1^{\mathrm{T}}\Sigma_S\hat{\boldsymbol{\omega}}_1 \\ + \frac{C_1}{2}||(\boldsymbol{A}\boldsymbol{\omega}_1 + b_1\boldsymbol{e}) - \boldsymbol{Y} + \varepsilon_1\boldsymbol{e}||^2 \end{aligned} \tag{25}$$

Then, setting the partial derivatives of $\boldsymbol{\omega}_1$ and b_1 in Equation (25) to zeros, we have:

$$\begin{aligned} \frac{\partial L(\boldsymbol{\omega}_1, b_1, \boldsymbol{\xi})}{\partial \boldsymbol{\omega}_1} = -\boldsymbol{A}^{\mathrm{T}}[\boldsymbol{Y} - \varepsilon_1\boldsymbol{e} - (\boldsymbol{A}\boldsymbol{\omega}_1 + b_1\boldsymbol{e})] + \lambda_1\left(\Sigma_x\boldsymbol{\omega}_1 - \Sigma_{xy}\right) \\ + C_1\boldsymbol{A}^{\mathrm{T}}[(\boldsymbol{A}\boldsymbol{\omega}_1 + b_1\boldsymbol{e}) - \boldsymbol{Y} + \varepsilon_1\boldsymbol{e}] = \boldsymbol{0} \end{aligned} \tag{26}$$

$$\frac{\partial L(\boldsymbol{\omega}_1, b_1, \boldsymbol{\xi})}{\partial b_1} = -\boldsymbol{e}^{\mathrm{T}}[\boldsymbol{Y} - \varepsilon_1\boldsymbol{e} - (\boldsymbol{A}\boldsymbol{\omega}_1 + b_1\boldsymbol{e})] + C_1\boldsymbol{e}^{\mathrm{T}}[(\boldsymbol{A}\boldsymbol{\omega}_1 + b_1\boldsymbol{e}) - \boldsymbol{Y} + \varepsilon_1\boldsymbol{e}] = 0 \tag{27}$$

Further, Equations (25) and (26) can be written in the following matrix form:

$$\begin{aligned} &\begin{bmatrix} (1+C_1)\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} + \lambda_1\Sigma_x & (1+C_1)\boldsymbol{A}^{\mathrm{T}}\boldsymbol{e} \\ (1+C_1)\boldsymbol{e}^{\mathrm{T}}\boldsymbol{A} & (1+C_1)\boldsymbol{e}^{\mathrm{T}}\boldsymbol{e} \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_1 \\ b_1 \end{bmatrix} \\ &= \begin{bmatrix} (1+C_1)\boldsymbol{A}^{\mathrm{T}}\boldsymbol{Y} + \lambda_1\Sigma_{xy} - (1+C_1)\mathbf{A}^{\mathrm{T}}\varepsilon_1\boldsymbol{e} \\ (1+C_1)\boldsymbol{e}^{\mathrm{T}}\boldsymbol{Y} - (1+C_1)\boldsymbol{e}^{\mathrm{T}}\varepsilon_1\boldsymbol{e} \end{bmatrix} \end{aligned} \tag{28}$$

Solving Equation (28), we can obtain:

$$\begin{aligned} &\begin{bmatrix} \boldsymbol{\omega}_1 \\ b_1 \end{bmatrix} \\ &= \begin{bmatrix} (1+C_1)\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} + \lambda_1\Sigma_x & (1+C_1)\boldsymbol{A}^{\mathrm{T}}\boldsymbol{e} \\ (1+C_1)\boldsymbol{e}^{\mathrm{T}}\boldsymbol{A} & (1+C_1)\boldsymbol{e}^{\mathrm{T}}\boldsymbol{e} \end{bmatrix}^{-1} \begin{bmatrix} (1+C_1)\boldsymbol{A}^{\mathrm{T}}\boldsymbol{Y} + \lambda_1\Sigma_{xy} - (1+C_1)\mathbf{A}^{\mathrm{T}}\varepsilon_1\boldsymbol{e} \\ (1+C_1)\boldsymbol{e}^{\mathrm{T}}\boldsymbol{Y} - (1+C_1)\boldsymbol{e}^{\mathrm{T}}\varepsilon_1\boldsymbol{e} \end{bmatrix} \end{aligned} \tag{29}$$

Therefore, the down-bound function can be constructed as:

$$f_1(\boldsymbol{x}) = \boldsymbol{\omega}_1\boldsymbol{x} + b_1 \tag{30}$$

Similarly, we can construct the up-bound function as:

$$f_2(\boldsymbol{x}) = \boldsymbol{\omega}_2\boldsymbol{x} + b_2 \tag{31}$$

The variables $\boldsymbol{\omega}_2$ and b_2 are determined by the following equation:

$$\begin{aligned} &\begin{bmatrix} \boldsymbol{\omega}_2 \\ b_2 \end{bmatrix} \\ &= \begin{bmatrix} (1+C_2)\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} + \lambda_2\Sigma_x & (1+C_2)\boldsymbol{A}^{\mathrm{T}}\boldsymbol{e} \\ (1+C_2)\boldsymbol{e}^{\mathrm{T}}\boldsymbol{A} & (1+C_2)\boldsymbol{e}^{\mathrm{T}}\boldsymbol{e} \end{bmatrix}^{-1} \begin{bmatrix} (1+C_2)\boldsymbol{A}^{\mathrm{T}}\boldsymbol{Y} + \lambda_2\Sigma_{xy} - (1+C_2)\mathbf{A}^{\mathrm{T}}\varepsilon_2\boldsymbol{e} \\ (1+C_2)\boldsymbol{e}^{\mathrm{T}}\boldsymbol{Y} - (1+C_2)\boldsymbol{e}^{\mathrm{T}}\varepsilon_2\boldsymbol{e} \end{bmatrix} \end{aligned} \tag{32}$$

Finally, the regression function of linear LSTPSVR can be constructed as follows:

$$f(\boldsymbol{x}) = \frac{1}{2}[f_1(\boldsymbol{x}) + f_2(\boldsymbol{x})] = \frac{1}{2}(\boldsymbol{\omega}_1 + \boldsymbol{\omega}_2)^{\mathrm{T}}\boldsymbol{x} + \frac{1}{2}(b_1 + b_2) \tag{33}$$

3.3. **Nonlinear LSTPSVR.** In order to apply LSTPSVR to nonlinear case, we can map the input data into RKHS by kernel trick. Unfortunately, due to the higher and infinite dimensions, it is rather difficult to express the kernel mapping explicitly during the process of solving $\begin{bmatrix} \boldsymbol{\omega}_k^{\mathrm{T}} & b_k \end{bmatrix}^{\mathrm{T}}$, $k = 1, 2$. Therefore, in this paper, to solve this difficulty, we attempt to map the input data into the empirical feature space [31-33].

Denote $\boldsymbol{K} = [k_{ij}]_{n\times n}$ as the kernel matrix, where $k_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $(\cdot, \cdot)$ represents inner product. Since $\boldsymbol{K}$ is a positive semi-definite matrix, it can be decomposed into the following form:

$$\boldsymbol{K} = \boldsymbol{P}\boldsymbol{\Lambda}\boldsymbol{P}^{\mathrm{T}} \tag{34}$$

where, $\boldsymbol{\Lambda}$ is an $r \times r$ dimensional diagonal matrix which consists of r positive eigenvalues of $\boldsymbol{K}$ in decreasing order, and $\boldsymbol{P}$ is an $n\times r$ dimensional eigenvector matrix which corresponds to the eigenvalues of $\boldsymbol{\Lambda}$.

In our work, we choose the mapping rules from the input data to empirical kernel space as follows:

$$\boldsymbol{x} \to \varphi(\boldsymbol{x}) = \boldsymbol{\Lambda}^{1/2}\boldsymbol{P}^{\mathrm{T}}(k(\boldsymbol{x}, \boldsymbol{x}_1), \ldots, k(\boldsymbol{x}, \boldsymbol{x}_n))^{\mathrm{T}} \tag{35}$$

We use the radial basis function (RBF), i.e., $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\big(-(\boldsymbol{x}_i, \boldsymbol{x}_j)^2/2\sigma^2\big)$, where σ is the predefined kernel width parameter.

After mapping the input data into the empirical kernel space by Equation (35), we can obtain a new training data set $\varphi(\boldsymbol{x})$. Note that the variance will change into $\sum_{\varphi(\boldsymbol{x})}$, which represents the empirical variance matrix of new data set $\varphi(\boldsymbol{x})$. Finally, we can obtain the regression function similar with linear LSTPSVR. The detailed derivative process is omitted here.

In a word, the aim of both linear and nonlinear LSTPSVR is to find a linear hyperplane for regression, and the key difference between them is that the kernel trick is utilized to map the input data into empirical kernel space in nonlinear LSTPSVR. In practice, nonlinear LSTPSVR is more useful than linear LSTPSVR; it can be applied in scenarios such as financial analysis, weather forecast, and biological fermentation process.

4. Simulation Experiments.

4.1. **Experimental design and parameter setting.** In order to verify the performance of our LSTPSVR, we compare it with SVR, TSVR, LSTSVR and LSTPISVR mentioned in the introduction part on UCI benchmark datasets and artificial test function, respectively. We choose the following three criteria to evaluate the performance of all algorithms.

Define $\hat{y}_i$ is the predicted value of output data, y_i is the actual output data, $\bar{y}$ is the mean of $\{y_i, i = 1, \ldots, n\}$, where n is the number of samples. Let $\mathrm{SSE} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$, $\mathrm{SSR} = \sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2$, $\mathrm{SST} = \sum_{i=1}^{n}(y_i - \bar{y})^2$, where SSE is the sum of squares due to error, SSR is the sum of squares of the regression, and SST is the total sum of squares, respectively.

Then, the root mean squared error (RMSE), the ratio between SSE and SST (marked as ET), and the ratio between SSR and SST (marked as RT) can be calculated as follows:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\mathrm{SSE}} \tag{36}$$

$$\mathrm{ET} = \frac{\mathrm{SSE}}{\mathrm{SST}} \tag{37}$$

$$\mathrm{RT} = \frac{\mathrm{SSR}}{\mathrm{SST}} \tag{38}$$

Note that the smaller is the RMSE, the smaller is the regression error; the smaller is the ET, the better is the consistency between actual and predicted value; the closer the RT is to 1, the better is the fitting performance of the algorithm.

Furthermore, we also record the training time cost by SVR, TSVR, LSTSVR, LSTPISVR and LSTPSVR, respectively. All experiments are implemented in the MATLAB R2015a on a PC with 3.3 GHz Intel I3 Processor and 4 GB RAM, and all results are averaged in 20 independent runs.

In order to obtain good prediction performance, setting suitable parameters for each regression algorithm is very important. In all experiments, due to the fact that the setting of ε_1 and ε_2 cannot greatly influence the prediction performance [25-30], and to make the comparison fair with SVR, we set $\varepsilon_1 = 0.1$ and $\varepsilon_2 = 0.1$, C_1 and C_2 are selected from the same set $\{2^i | i = -9, \ldots, 0, \ldots, 9\}$. In LSTPISVR [28], we set $C_1 = C_2$ and $v_1 = v_2$; v_1 and v_2 are selected from the same set as C_1. In our LSTPSVR, we set $C_1 = C_2$ and $\lambda_1 = \lambda_2$; λ_1 and λ_2 are selected from the same set as C_1. In linear test, we use linear kernel, while in nonlinear test, we use RBF kernel, and the kernel width parameter of RBF is set as $\sigma = 1.7320$.

4.2. **Experimental results and analysis.**

4.2.1. *The linear test.* The UCI benchmark datasets used in our experiments are presented in Table 1, whose sizes vary from 103 to 1503. They can all be downloaded from UCI machine learning repository.

TABLE 1. The UCI benchmark datasets used in our experiments

Dataset	**Number of training samples**	**Attributions**
Slump	103	8
Yacht	308	7
Stock	315	12
Boston	506	14
Enb_2012	768	7
Concrete	1030	9
Airfoil	1503	6

We employ standard 10-fold cross-validation technique. Table 2 presents the average RMSE, ET, RT and training time in 20 independent runs on seven different UCI benchmark datasets. The best average RMSE, ET, RT and training time are marked with bold font.

We can see clearly from Table 2 that most RMSE and ET of LSTPSVR are smaller than other algorithms, and the RT of LSTPSVR is closest to 1. This implies that our LSTPSVR algorithm has better prediction performance. Moreover, compared with powerful LSTPISVR, most evaluation criteria of LSTPSVR have improved except for the training time. This can be explained by the fact that the introduced new term promotes the prediction performance. By seeking two suitable projection axes, the data prior structural information is considered automatically. Therefore, it is beneficial for LSTPSVR to obtain better up- and down- bound functions. However, adding a new term to the objective function will make the linear matrix of LSTPSVR more complicated than LSTSVR and LSTPISVR. Hence, the training time cost by LSTPSVR is slightly longer than LSTSVR and LSTPISVR, but significantly shorter than TSVR. In summary, introducing variance can improve the prediction performance of our LSTPSVR algorithm.

TABLE 2. The results of average RMSE, ET, RT and training time on UCI benchmark datasets using different algorithms in linear test

Datasets	Algorithms	RMSE	ET	RT	Training time (s)
Slump	SVR	7.4062	0.9401	0.4848	0.2243
	TSVR	7.5346	0.9517	0.9413	0.0401
	LSTSVR	7.6539	1.1250	0.6392	$\mathbf{0.1371 \times 10^{-3}}$
	LSTPISVR	7.5307	1.1200	0.6885	0.2725×10^{-3}
	LSTPSVR	**7.3926**	**0.8003**	**1.0155**	0.4486×10^{-3}
Yacht	SVR	10.0247	0.4602	0.2844	2.5432
	TSVR	9.2564	0.3736	0.9591	0.1872
	LSTSVR	9.0291	0.3815	0.7889	0.2524×10^{-3}
	LSTPISVR	8.9222	**0.3695**	0.6865	$\mathbf{0.2141 \times 10^{-3}}$
	LSTPSVR	**8.7939**	0.3963	**1.0144**	0.9825×10^{-3}
Stock	SVR	0.1162	0.6788	0.4829	0.4921
	TSVR	0.1152	0.7555	0.9134	0.1722
	LSTSVR	0.1137	0.7378	0.4288	$\mathbf{0.1704 \times 10^{-3}}$
	LSTPISVR	0.1127	0.7510	1.0166	0.1909×10^{-3}
	LSTPSVR	**0.1055**	**0.6537**	**1.0093**	1.2118×10^{-3}
Boston	SVR	5.0899	0.3182	0.9539	12.4702
	TSVR	4.9027	0.2997	0.9804	0.4545
	LSTSVR	4.8075	0.2884	0.7783	0.3288×10^{-3}
	LSTPISVR	4.8300	0.2845	0.7847	$\mathbf{0.3237 \times 10^{-3}}$
	LSTPSVR	**4.7838**	**0.2750**	**1.0007**	2.8521×10^{-3}
Enb_2012	SVR	2.9642	0.0953	0.9030	43.2438
	TSVR	2.9424	0.0918	0.9962	1.3626
	LSTSVR	2.9588	**0.0838**	0.9289	$\mathbf{0.2974 \times 10^{-3}}$
	LSTPISVR	**2.8878**	0.0880	0.9462	0.3390×10^{-3}
	LSTPSVR	2.9343	0.0895	**1.0014**	2.9362×10^{-3}
Concrete	SVR	10.5028	0.4547	0.9510	113.8346
	TSVR	10.5521	0.3956	0.8936	2.9822
	LSTSVR	10.4277	0.3869	0.6093	0.3646×10^{-3}
	LSTPISVR	10.4424	0.3901	0.6230	$\mathbf{0.3341 \times 10^{-3}}$
	LSTPSVR	**10.3239**	**0.3799**	**0.9968**	3.0644×10^{-3}
Airfoil	SVR	6.9361	0.4714	0.7852	350.3474
	TSVR	4.8742	0.4942	0.7915	6.2883
	LSTSVR	4.7965	0.4814	0.5255	$\mathbf{0.3797 \times 10^{-3}}$
	LSTPISVR	**4.7850**	**0.4579**	0.7165	0.5110×10^{-3}
	LSTPSVR	4.8643	0.5635	**0.8363**	3.8001×10^{-3}

4.2.2. *The nonlinear test.* We conduct experiments on the most commonly used artificial nonlinear test function, i.e., sinc function, to test the performance of different algorithms.

The sinc function is defined as follows:

$$y_i = \mathrm{sinc}(x_i) = \frac{\sin x_i}{x_i} + n_i, \quad x_i \sim U[-3\pi, +3\pi] \tag{39}$$

where, x_i is the input data, $U[a, b]$ represents the uniformly-distributed variable in the closed interval $[a, b]$, n_i is noise, and y_i is the output data.

Additionally, in order to better test the performance of different algorithms, we use the following two categories of noise:

Category A: $n_i = \left(-\frac{|x_i|}{8\pi} + 0.5\right) \times \zeta_i$, $\zeta_i \sim U[-0.5, 0.5]$

Category B: $n_i = \left(-\frac{|x_i|}{8\pi} + 0.5\right) \times \zeta_i$, $\zeta_i \sim N[0, 0.25^2]$

where $N[0, 0.25^2]$ represents normally-distributed variable with zero mean and variance 0.25^2, i.e., Gaussian distribution.

In the nonlinear test, 100 training samples and 200 test samples are both randomly generated with two categories of noise. The average RMSE, ET, RT and training time in 20 independent runs on sinc function using two different categories of noise are presented in Table 3. Again, the best average RMSE, ET, RT and training time are marked with bold font.

TABLE 3. The results of average RMSE, ET, RT and training time on sinc function using different algorithms in nonlinear test

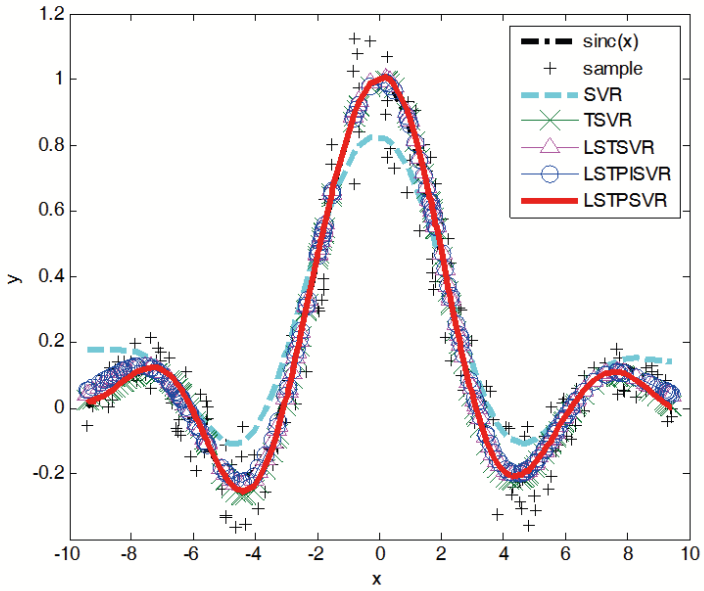
Categories of noise	Algorithms	RMSE	ET	RT	Training time (s)
A	SVR	0.1154	0.0871	0.9338	0.0764
	TSVR	0.0998	0.0708	0.9648	0.0494
	LSTSVR	0.0996	0.0711	0.9007	**0.0051**
	LSTPISVR	0.0988	**0.0697**	0.9098	0.0061
	LSTPSVR	**0.0986**	0.0707	**0.9902**	0.0077
B	SVR	0.1137	0.0764	0.9568	0.0813
	TSVR	0.0871	0.0546	0.9787	0.0507
	LSTSVR	0.0865	0.0546	0.9382	**0.0042**
	LSTPISVR	0.0858	**0.0544**	0.9378	0.0052
	LSTPSVR	**0.0853**	0.0549	**0.9881**	0.0068

It can be seen from Table 3 that LSTSVR costs the least training time compared with other algorithms. The reason is that the least squares method only needs to solve the solution of linear equations, hence the time complexity is much smaller than TSVR. This causes the training time cost by LSTSVR is much shorter than TSVR. However, in comparison to LSTSVR, a new term is introduced in our LSTPSVR algorithm which makes the solution of linear matrix more complicated. Hence, the computation of linear matrix becomes slower. Due to the same reason, LSTPSVR is a little slower than LSTPISVR, but greatly faster than TSVR.

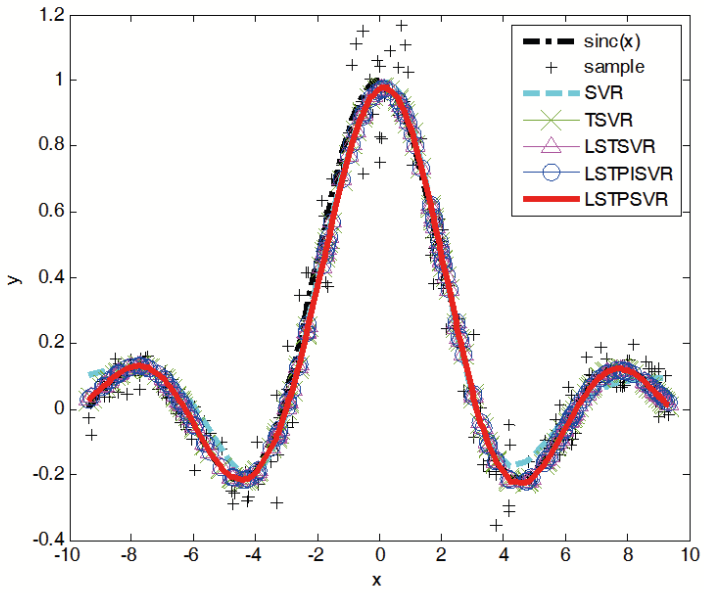
In addition to training time, our LSTPSVR has similar or better prediction performance in comparison with other algorithms. RMSE and ET of LSTPSVR are close to TSVR, LSTSVR, and LSTPISVR. Specially, RT of LSTPSVR has a significant improvement. The closer the RT is to 1, the better is the fitting performance of the algorithm. The underlying cause is that the data prior structural information is embedded into LSTPSVR automatically, which ensures that our LSTPSVR can find more accurate regression function than other algorithms.

In summary, our LSTPSVR introduces the variance of projected data, and the purpose of minimizing the variance is to find a proper projection axis that all projected zone is smallest. This means that the projection axis corresponds to $\boldsymbol{\omega}$ of LSTPSVR from mathematical perspective. Therefore, minimizing the variance is helpful to promote the prediction performance of the proposed algorithm.

Figure 1 shows the fitting capacity of different algorithms using two different categories of noise. It is clear that the fitting capacity of LSPTSVR is the best compared with other algorithms. This means the anti-interference ability of LSPTSVR is higher than other algorithms.



(a) Uniformly-distributed noise

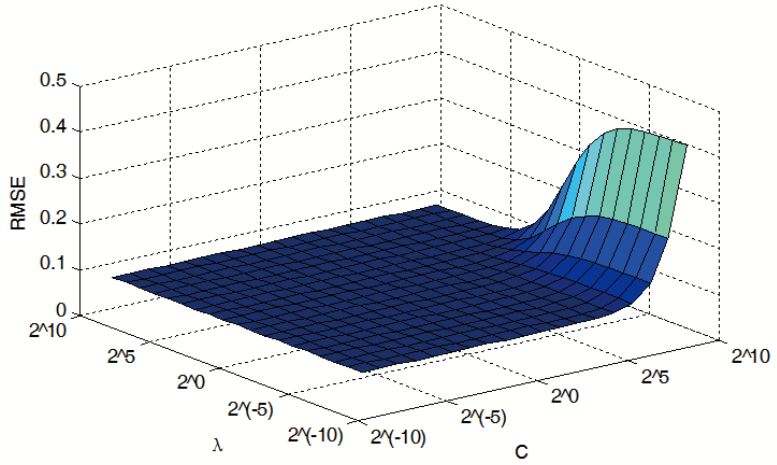


(b) Normally-distributed noise

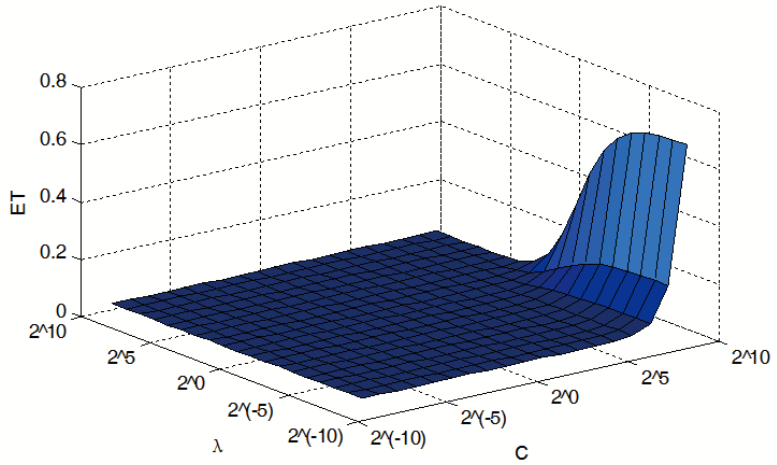
FIGURE 1. Fitting capacity of different algorithms

In our LSTPSVR, in order to counter-balance the introduced variance, we add a penalty parameter λ as its weight. Next, we will illustrate the impact of C and λ on evaluation criteria in Figure 2. The experimental results are based on noise category B, and the parameters C and λ are all selected from 2^{-9} to 2^{9}.

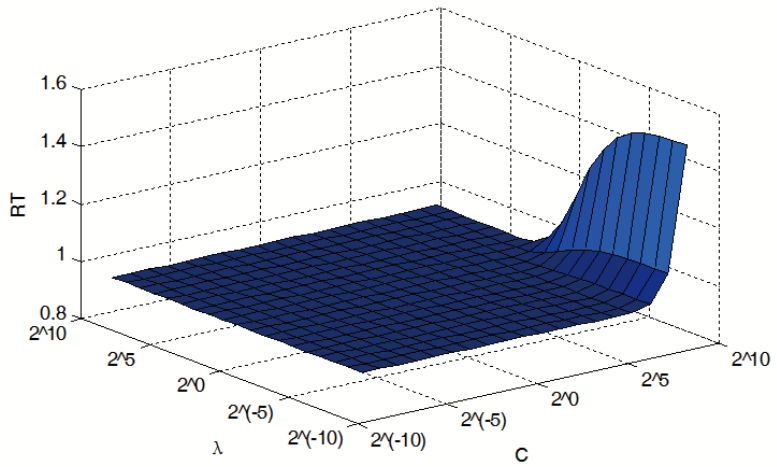
It can be seen clearly from Figure 2 that the proposed LSTPSVR algorithm is stable when given suitable λ and C. Furthermore, it is worth noting that the RMSE, ET and RT are unchanged in a large scale of C values when given suitable λ. Therefore, finding a suitable value of λ is a good approach to obtain good gernalization performance. On the other hand, finding a suitable C has the same effect.



(a) The change of RMSE



(b) The change of ET



(c) The change of RT

FIGURE 2. The change of three evaluation criteria under different C and λ during the whole optimization process

5. **Conclusions.** In this paper, we present a least squares twin projection support vector regression (LSTPSVR). The main contribution of this paper is that the data prior structural information is automatically embedded into the training process by introducing the variance of the projected data. Minimizing the variance of the projected data can ensure that we can obtain the most suitable projection axis. The results on linear and nonlinear test demonstrate that the prediction performance of LSTPSVR is better than other state-of-the art algorithms. Furthermore, LSTPSVR is stable when given suitable parameters.

However, it should be pointed out that the main shortage of LSTPSVR is that it is slightly more time-consuming than other methods since a new term is introduced into the object function. Therefore, finding a good way to shorten the training time of LSTPSVR is one of our future works. In addition, our LSTPSVR also loses sparsity. We hope these questions can be successfully answered in the near future.

Acknowledgment. This research was supported by the National Natural Science Foundation of China under Grant Numbers 21878124, 31771680 and 61773182.

REFERENCES

[1] V. Vapnik, *Statistical Learning Theory*, John Wiley and Sons, Inc., New York, 1998.
[2] J. Ruan, Y. Shi and J. Yang, Forest fires burned area prediction based on support vector machines with feature selection, *ICIC Express Letters*, vol.5, no.8(A), pp.2597-2603, 2011.
[3] J. Ruan, X. P. Wang and Y. Shi, Developing fast predictors for large-scale time series using fuzzy granular support vector machines, *Applied Soft Computing*, vol.13, no.9, pp.3981-4000, 2013.
[4] H. Kalke and M. Loewen, Support vector machine learning applied to digital images of river ice conditions, *Cold Regions Science and Technology*, vol.155, pp.225-236, 2018.
[5] L. Zhang and W. Zhou, Fisher-regularized support vector machine, *Information Sciences*, vol.343, pp.79-93, 2016.
[6] F. E. Gunawan, Improving the reliability of F-statistic method by using linear support vector machine for structural health monitoring, *ICIC Express Letters*, vol.12, no.12, pp.1183-1193, 2018.
[7] A. Safari, An e-E-insensitive support vector regression machine, *Computational Statistics*, vol.29, no.6, pp.1447-1468, 2014.
[8] L. Yuan and G. Zhong, Robust ε-support vector regression, *Mathematical Problems in Engineering*, vol.2014, pp.1-5, 2014.
[9] Q. Wu, A new class of ε-piecewise smooth support vector regressions, *Journal of Information Science and Engineering*, vol.31, no.5, pp.1813-1828, 2015.
[10] K. Wang, W. Zhu and P. Zhong, Robust support vector regression with generalized loss function and applications, *Neural Processing Letters*, vol.41, no.1, pp.89-106, 2015.
[11] Y. Zhang, S. Xu, K. Chen, Z. Liu and C. L. P. Chen, Fuzzy density weight-based support vector regression for image denoising, *Information Sciences*, vol.339, pp.175-188, 2016.
[12] Y. Xu, X. Li, X. Pan and Z. Yang, Asymmetric ν-twin support vector regression, *Neural Computing and Applications*, vol.30, no.12, pp.3799-3814, 2018.
[13] B. Schölkopf, A. J. Smola, R. C. Williamson and P. L. Bartlett, New support vector algorithms, *Neural Computation*, vol.12, no.5, pp.1207-1245, 2000.
[14] B. Gu and F. Pan, Effective ν-solution path for ν-support vector regression, *International Journal of Innovative Computing, Information and Control*, vol.11, no.6, pp.2103-2117, 2015.
[15] P. Hao, New support vector algorithms with parametric insensitive/margin model, *Neural Networks*, vol.23, no.1, pp.60-73, 2010.
[16] X. Peng and D. Xu, Projection support vector regression algorithms for data regression, *Knowledge-Based Systems*, vol.112, pp.54-66, 2016.
[17] Jayadeva, R. Khemchandani and S. Chandra, Twin support vector machines for pattern classification, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol.29, no.5, pp.905-910, 2007.
[18] X. Peng, TSVR: An efficient twin support vector machine for regression, *Neural Networks*, vol.23, no.3, pp.365-372, 2010.
[19] Y. Shao, C. Zhang, Z. Yang, L. Jing and N. Deng, An ε-twin support vector machine for regression, *Neural Computing and Applications*, vol.23, no.1, pp.175-185, 2013.

[20] S. Balasundaram and M. Tanveer, On Lagrangian twin support vector regression, *Neural Computing and Applications*, vol.22, pp.S257-S267, 2013.
[21] Y. Xu and L. Wang, K-nearest neighbor-based weighted twin support vector regression, *Applied Intelligence*, vol.41, no.1, pp.299-309, 2014.
[22] X. Peng, Efficient twin parametric insensitive support vector regression model, *Neurocomputing*, vol.79, pp.26-38, 2012.
[23] X. Peng, D. Xu and J. D. Shen, A twin projection support vector machine for data regression, *Neurocomputing*, vol.138, pp.131-141, 2014.
[24] Y. Xu, A novel twin support vector regression, *Journal of Information Science and Engineering*, vol.31, no.2, pp.627-640, 2015.
[25] Y. Xu and L. Wang, A weighted twin support vector regression, *Knowledge-Based Systems*, vol.33, pp.92-101, 2012.
[26] M. A. Kumar and M. Gopal, Least squares twin support vector machines for pattern classification, *Expert Systems with Applications*, vol.36, no.4, pp.7535-7543, 2009.
[27] H. Huang, S. Ding and Z. Shi, Primal least squares twin support vector regression, *Journal of Zhejiang University – Science C – Computers and Electronics*, vol.14, no.9, pp.722-732, 2013.
[28] S. Ding and H. Huang, Least squares twin parametric insensitive support vector regression, *Journal of Software*, vol.28, no.12, pp.3146-3155, 2017.
[29] H. Huang, X. Wei and Y. Zhou, A sparse method for least squares twin support vector regression, *Neurocomputing*, vol.211, pp.150-158, 2016.
[30] Z. Zhang, T. Lv, H. Wang, L. Liu and J. Tan, A novel least square twin support vector regression, *Neural Processing Letters*, vol.48, no.2, pp.1187-1200, 2018.
[31] B. Schölkopf, S. Mika, C. J. C. Burges, P. Knirsch, K. R. Müller, G. Rätsch and A. J. Smola, Input space versus feature space in kernel-based methods, *IEEE Trans. Neural Networks*, vol.10, no.5, pp.1000-1017, 1999.
[32] H. L. Xiong, M. N. S. Swamy and M. O. Ahmad, Optimizing the kernel in the empirical feature space, *IEEE Trans. Neural Networks*, vol.16, no.2, pp.460-474, 2005.
[33] O. L. Mangasarian and E. W. Wild, Multisurface proximal support vector machine classification via generalized eigenvalues, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol.28, no.1, pp.69-74, 2006.