Twin Least Squares Support Vector Regression of Heteroscedastic Gaussian Noise Model

SHIGUANG ZHANG\(^1,2\), CHAO LIU\(^1\), TING ZHOU\(^3\) AND LIN SUN\(^4\), (Member, IEEE)

\(^1\)College of Computer and Information Engineering, Henan Normal University, Xinxiang 453007, China
\(^2\)School of Computer Science and Technology, Tianjin University, Tianjin 300072, China
\(^3\)State-Owned Assets Management Office, Henan Normal University, Xinxiang 453007, China
\(^4\)College of Computer and Information Engineering, Henan Normal University, Xinxiang 453007, China

Corresponding authors: Shiguang Zhang (sg202006@126.com) and Ting Zhou (zhouting7606@163.com)

This work was supported in part by the Natural Science Foundation Project of Henan under Grant 182300410130 and Grant 182300410368, and in part by the National Natural Science Foundation of China (NSFC) under Grant 61772176 and Grant 61402153.

ABSTRACT The training algorithm of twin least squares support vector regression (TLSSVR) transforms unequal constraints into equal constraints in a pair of quadratic programming problems, it owns faster computational speed. The classical least squares support vector regression (LSSVR) assumpt that the noise is Gaussian with zero mean and the homoscedastic variance. However, it is found that the noise models in some practical applications satisfy Gaussian distribution with zero mean and heteroscedastic variance. In this paper, the LSSVR is combined with the twin hyperplanes, and then an optimal loss function for Gaussian noise with heteroscedasticity is introduced, which is called the twin least squares support vector regression with heteroscedastic Gaussian noise (TLSSVR-HGN). Like LSSVR, TLSSVR-HGN also lacks sparsity. To analyze the generalization ability of the proposed model, the sparse TLSSVR-HGN (STLSSVR-HGN) is proposed with a simple mechanism. The proposed model has been verified using the artificial data set, several benchmark data sets and actual wind speed data. The experimental results show that TLSSVR-HGN is a better technology than the other algorithms.

INDEX TERMS Least squares support vector regression, twin hyperplanes, equality constraints, heteroscedastic gaussian noise.

I. INTRODUCTION
Support vector machine (SVM) is a statistical learning theory for solving classification and regression problems [1]–[3], and it shows some advantages in various fields due to its good generalization ability. For example, it has been used in pattern recognition [4], feature selection [5], function approximation [6] and financial regression [7]. Compared with other machine learning algorithms, SVM has unique algorithm characteristics. However, from another perspective, SVM itself has the disadvantage of high computational complexity. To solve this problem, Many researchers have established some effective algorithms (e.g. SMO in [8], SVMTorch in [9]) and developed usable software tools (LIBSVM in [10]). In addition, many people have built some simple models. For example, LSSVM [11] is faster than classical SVM in terms of computational complexity by solving linear equations. \(\nu\)-support vector machines [12] verified parameter insensitive/edge models with arbitrary shapes. Mangasarian and Wild [13] proposed the multisurface proximal support vector machine classification via generalized eigenvalues (GEPSVM). From the GEPSVM, Jayadeva et al. [14] proposed a new machine learning method based on binary classification problem, which is called twin support vector machine (TWSVM). On this basis, Ding et al. [15] divided TWSVM into multiple categories and analyzed it based on basic theoretical and geometrical meanings. TWSVM produces two non-parallel hyperplanes, each hyperplane will be as close as possible to one of the two hyperplanes, and to some extent away from the other hyperplane. Compared with classical SVM, TWSVM has faster computing speed and better generalization ability. Thanks to these advantages, TWSVM has been applied in these fields, including identity recognition [16], feature selection [17], verification of handwritten signature [18], wind speed prediction [19], social lending [20] and classification of food [21] etc. To improve multi-classification, many people proposed some methods, including multi-class SVM.
using second-order cone constraints [22] and Multi-class support vector machine classifiers using intrinsic and penalty graphs [23] etc.

As for support vector regression (SVR), it is to solve the regression problem, and there are also many improved algorithms. In particular, Suykens et al. [24] proposed least squares support vector regression (LSSVR) by introducing the least squares method. In LSSVR, the inequality constraints of SVR are transformed into equality constraints. This strategy can reduce the complexity of SVR and thus improve the learning speed. The core of classical SVR is the minimization of the convex quadratic function, which obeys the pair of linear inequality constraints for all training samples, which leads to its slow learning speed. So Peng [25] proposed the twin support vector regression (TSVR) as an effective regressor, and defined a group of epsilon insensitivity parameters to solve two related SVM type problems. Zhao et al. [26] proposed the notion of twin hyperplanes with the fast speed of least squares support vector regression (LSSVR) yields a new regressor, termed as twin least squares support vector regression (TLSSVR). N.Parastalooi et al. [27] proposed modified twin support vector regression (MTSVR) for data regression. Peng [28] proposed a pair of quadratic programming problems (QPP) for directly optimizing TSVR in primal space (PTSVR) based on a series of linear equations. Y.Shao et al. [29] proposed two related SVM type problems are solved to determine a pair of ε-insensitive proximal functions. R.Khemchandani et al. [30] inspired by Bennett and Bi’s work on SVM [31], a new regression model called TWSVR is proposed. Moreover, many other researchers have also done a lot of work [32–38].

In addition, TSVR only solves a pair of smaller quadratic programming problems, which is different from the classical SVR. As a result, the number of constraints in each quadratic programming problem is smaller than that in the classical SVR, so the training samples speed will be faster. However, TLSSVR transforms inequality constraint problem into equality constraint problem, which is relatively low accuracy. Therefore, there is some limitations. In order to improve the accuracy of TLSSVR, we propose a new optimization algorithm based on twin support vector regression (TSVR) and twin least squares TSVR (TLSSVR) to make up for some shortcomings of TLSSVR. It mainly combines TSVR, LSSVR and introduces heteroscedastic, homoscedastic Gaussian noise error, which is called twin least squares support vector regression with heteroscedastic Gaussian noise (TLSSVR-HGN) and homoscedastic Gaussian noise (TLSSVR-GN). In addition, this article introduces the sparse version of TLSSVR-HGN (STLSSVR-HGN). In the real data set, we use the wind speed data set, the experimental results have verified that the accuracy of TLSSVR-HGN is higher than STLSSVR-HGN, TLSSVR-GN, TLSSVR, LSVR and SVR. At the same time, TLSSVR-HGN has better generalization ability than the other five algorithms. We have done similar experiments on other data sets, and the feasibility and effectiveness of the proposed methods are validated.

The rest of this article is structured as follows. The second section mainly explains some symbols used and the derivation of TSVR and TLSSVR formulas. The third section mainly introduces the TLSSVR-HGN modeling process, the TLSSVR-HGN algorithm and the STLSSVR-HGN algorithm. The fourth section mainly describes the experimental results and analysis on the artificial data set, several UCI data sets and wind speed data set. The last section summarizes this work.

II. RELATED WORK

In this section, we review the classical TSVR and TLSSVR based on Gaussian noise distribution.

Considering the training data set $X_i = \{(x_1, y_1), (x_2, y_2), \ldots, (x_l, y_l)\}$, where $x_i = (x_{i1}, x_{i2}, \ldots, x_{in})^T \in \mathbb{R}^n$, $y_i \in \mathbb{R}, i = 1, 2, \ldots, l$. A multivariate linear model is $f(x) = \omega^T \cdot x + b$, where $\omega \in \mathbb{R}^n, b \in \mathbb{R}$. For the sake of simplicity, let matrix $X = [x_1, x_2, \ldots, x_l], y = [y_1, y_2, \ldots, y_l]^T$, where the superscript $T$ represents the transpose symbol.

Assume the noise in $X_i$ is additive, namely

$$y_i = f(x_i) + \xi_i, \quad (i = 1, 2, \ldots, l),$$

where $\xi_i$ is random, under the assumption that the observations are drawn i.i.d.(independently and identically distributed) with $P(\xi)$ of mean $\mu$ and standard deviation $\sigma_i (l = 1, \ldots, l)$.

A. TWIN SUPPORT VECTOR REGRESSION

In order to improve the computational speed and generalization performance of normal support vector regression (SVR), in spirit of twin support vector machine (TWSVM) [14], Peng [25] proposed an efficient regressor, termed as twin support vector regression (TSVR), which is obtained by solving the following two quadratic programming problems, shown as

$$\min \min \{g_{\text{PSVR}} = \frac{1}{2} (y-e \cdot \epsilon_1 - (K(X, X^T)\omega_1 + e \cdot b_1))^T \cdot (y-e \cdot \epsilon_1 - (K(X, X^T)\omega_1 + e \cdot b_1)) + C_1 \cdot e^T \cdot \xi \}$$

s.t. $y - (K(X, X^T)\omega_1 + e \cdot b_1) \geq e \cdot \epsilon_1 - \epsilon, \xi \geq 0$ (2)

and

$$\min \{g_{\text{PSVR}} = \frac{1}{2} (y+e \cdot \epsilon_2 - (K(X, X^T)\omega_2 + e \cdot b_2))^T \cdot (y+e \cdot \epsilon_2 - (K(X, X^T)\omega_2 + e \cdot b_2)) + C_2 \cdot e^T \cdot \eta \}$$

s.t. $(K(X, X^T)\omega_2 + e \cdot b_2) - y \geq e \cdot \epsilon_2 - \eta, \eta \geq 0$ (3)

where $K(X, X^T)$ represents the kernel matrix whose element $k_{ij} = k(x_i, x_j^T), x_j^T$ is the row vector of $X, e$ is a vector of unities of proper size, $\epsilon_1 \geq 0$ and $\epsilon_2 \geq 0$ are threshold parameters, $\xi$ and $\eta$ are slack vectors. Here, $k(x_i, x_j) = \phi(x_i)\phi(x_j)$ is kernel function which gives the inner product of $\phi(x_i)$ and $\phi(x_j)$ in the high dimensional feature space, where $\phi(\bullet)$ is usually a nonlinear mapping, which
can map the input space \( x_i \) into a high-dimensional feature space \( \phi(x_i) \).

The dual problem of \( P_{TSVR} \) is expressed in the following
\[
\begin{align*}
\min \left\{ g_{D_{TSVR}} & = \frac{1}{2} \alpha^T H (H^T H)^{-1} H^T \alpha - g^T H (H^T H)^{-1} \\ & \quad \cdot H^T \alpha + g^T \alpha \right\} \\
\text{s.t.} \ 0 \leq \alpha \leq C_1 e
\end{align*}
\]

and
\[
\begin{align*}
\min \left\{ g_{D_{TSVR}} & = \frac{1}{2} \beta^T H (H^T H)^{-1} H^T \beta + h^T H (H^T H)^{-1} \\ & \quad \cdot H^T \beta - h^T \beta \right\} \\
\text{s.t.} \ 0 \leq \beta \leq C_2 e
\end{align*}
\]

where \( H = [K(X, X^T), e], g = y - ee_1, h = y + ee_2 \). When the optimal solutions \( \alpha \) of (2) and \( \beta \) of (3) are obtained, the optimal orientation vectors and biases follows:
\[
(\omega_1, b_1)^T = (H^T H)^{-1} H^T (g - \alpha),
\]
\[
(\omega_2, b_2)^T = (H^T H)^{-1} H^T (h + \beta).
\]

And two functions \( f_1(x) \) and \( f_2(x) \) are obtained as,
\[
f_1(x) = \omega_1^T K(X, x) + b_1 \quad \text{and} \quad f_2(x) = \omega_2^T K(X, x) + b_2.
\]

Each one determines the \( \epsilon \)-insensitive lower-bound or upper-bound regressors. Hence, the predictor of TSVR is constructed as
\[
f_{TSVR}(x) = \frac{f_1(x) + f_2(x)}{2} = \frac{1}{2} (\omega_1 + \omega_2)^T K(X, x) + \frac{1}{2} (b_1 + b_2).
\]

In order to obtain twin least squares support vector regression (TLSSVR) [26],
\[
\begin{align*}
\min \left\{ g_{P_{TLSSVR}} & = \frac{1}{2} \omega_1^T \cdot \omega_1 + \frac{C_1}{2} \sum_{i=1}^l v_i \cdot \xi_i^2 \right\} \\
P_{TLSSVR} : \ s.t. \ y_i = \omega_1^T \cdot \Phi(x_i) + b_1 + \xi_i + \epsilon_i (i = 1, \cdots, l) \quad (6)
\end{align*}
\]

and
\[
\begin{align*}
\min \left\{ g_{P_{TLSSVR}} & = \frac{1}{2} \omega_2^T \cdot \omega_2 + \frac{C_2}{2} \sum_{i=1}^l v_i^s \cdot \xi_i^{s2} \right\} \\
P_{TLSSVR} : \ s.t. \ y_i = \omega_2^T \cdot \Phi(x_i) + b_2 + \xi_i^s - \epsilon_2 (i = 1, \cdots, l) \quad (7)
\end{align*}
\]

where \( \omega_1, \omega_2 \) are the normal vectors of the hyperplane, \( b_1, b_2 \) are the offsets, \( \xi_i, \xi_i^s (i = 1, \cdots, l) \) represent the prediction residual, \( C_1, C_2 \) are the regularization parameters. \( v_i \geq 0, v_i^s \geq 0 \) are employed to weight the slack errors \( \xi_i, \xi_i^s \), respectively.

By the Lagrangian multiplier method, we have the solution of Eq. (6):
\[
\left[ \begin{array}{c} 0 \\ e \end{array} \right] = \begin{bmatrix} \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ y - \epsilon_1 e \end{bmatrix}
\]

where \( \hat{K}_{ij} = k(x_i, x_j) + \frac{\delta_{ij}}{\nu C_1}, \) with \( \delta_{ij} = 1(i = j), \) or \( \delta_{ij} = 0(i \neq j), i, j = 1, 2, \cdots, l. \) And weights \( v_i = \rho(\alpha_i < 0), \) or \( v_i = 1(\alpha_i \geq 0), \) \( \rho > 1. \) After obtaining the solution \( \alpha \) by (6), for any new testing samples \( x \in R^l \), we have the lower-bound prediction function of TLSSVR:
\[
f_1(x) = \sum_{l=1}^L \alpha_l k(x_i, x) + b_1.
\]

Analogously, for any new testing samples \( x \in R^l \), we have the upper-bound prediction function of TLSSVR by (7):
\[
f_2(x) = \sum_{l=1}^L \alpha_l^s k(x_i, x) + b_2.
\]

In LSSVR, the Lagrange multiplier vector \( \alpha \) is proportional to the error vector \( \xi \) in the training sets, indicating that LSSVR loses sparsity. In addition, compared with SVR, LSSVR has almost no robustness, which indicates that LSSVR also has some shortcomings compared with classical SVR, and some new methods are needed to make up for these shortcomings. And models TSVR and TLSSVR do not consider the heteroscedasticity of \( D_t \). In the following section, we construct a new twin Least squares support vector regression for heteroscedastic tasks.

iii. Twin Least Squares Support Vector Regression of Heteroscedastic Gaussian Noise Model and Algorithm

As a function approximation machine, the objective is to estimate an unknown function \( f(x) \) from the data set \( X_f \subseteq X_p \). Generally, the optimal loss function in a maximum likelihood sense [39], [40] is
\[
c(\xi_i) = -\log p(\xi_i).
\]

where \( p(\xi) \) is probability distribution of \( \xi_i \).
We assume that the noise in Equation (1) is Gaussian, with zero mean and heteroscedastic variance \( \sigma_i^2 \), for \( i = 1, \ldots, l \), that is \( p(\xi_i) = \frac{1}{\sqrt{2\pi\sigma}}e^{-\frac{\xi_i^2}{2\sigma^2}} \). By Equation (9), the optimal loss function corresponding to Gaussian noise with heteroscedasticity is

\[
c(\xi_i) = \frac{1}{2\sigma_i^2} \xi_i^2 \quad (i = 1, \ldots, l).
\] (10)

If the noise in Equation (1) is Gaussian, with zero mean and the homoscedastic variance \( \sigma^2 \), the loss function is degenerated to \( c(\xi_i) = \frac{1}{2} \xi_i^2 \) (\( i = 1, \ldots, l \)), which is the squared loss function.

### A. TWIN LEAST SQUARES SUPPORT VECTOR REGRESSION OF HETEROSCEDASTIC GAUSSIAN NOISE MODEL

Given the training samples \( X_j \), we construct a lower-bound nonlinear regression function \( f_1(x) = \omega_1^T \cdot \Phi(x) + b_1 \). Following the idea of forming two hyperplanes in TSVR, the model of twin least squares support vector regression of heteroscedastic Gaussian noise model (TLSSVR-HGN) can be formally defined as

\[
\begin{align*}
\min_{\omega_1, b_1, \xi_1} & \quad \frac{1}{2} \omega_1^T \cdot \omega_1 + \frac{C_1}{2l} \sum_{i=1}^{l} \frac{1}{\sigma_i} \cdot \xi_i^2 \\
P_{\text{TLSSVR--HGN}} : & \quad \text{s.t. } y_i = \omega_1^T \cdot \Phi(x_i) + b_1 + \xi_i + \epsilon_1 \\
& \quad (i = 1, \ldots, l)
\end{align*}
\] (11)

where \( \sigma_i^2 \) (\( i = 1, 2, \ldots, l \)) are variances; \( \epsilon_1 \geq 0 \) is threshold parameter, \( C_1 > 0 \) is penalty parameter; whereas \( \xi_i(i = 1, \ldots, l) \) are random variables.

Analogously, an upper-bound nonlinear regression function \( f_2(x) = \omega_2^T \cdot \Phi(x) + b_2 \). The uniform model of TLSSVR-HGN of an upper-bound can be formally defined as

\[
\begin{align*}
\min_{\omega_2, b_2, \xi_2} & \quad \frac{1}{2} \omega_2^T \cdot \omega_2 + \frac{C_2}{2l} \sum_{i=1}^{l} \frac{1}{\sigma_i^2} \cdot \xi_i^2 \\
P_{\text{TLSSVR--HGN}} : & \quad \text{s.t. } y_i = \omega_2^T \cdot \Phi(x_i) + b_2 + \xi_i^* - \epsilon_2 \\
& \quad (i = 1, \ldots, l)
\end{align*}
\] (12)

where \( \sigma_i^2 \) (\( i = 1, 2, \ldots, l \)) are variances; \( \epsilon_2 \geq 0 \) is threshold parameter, \( C_2 > 0 \) is penalty parameter; whereas \( \xi_i^*(i = 1, \ldots, l) \) are random variables.

We introduce the Lagrangian functional \( L(\omega_1, b_1, \alpha, \xi) \) as

\[
L(\omega_1, b_1, \alpha, \xi) = \frac{1}{2} \omega_1^T \cdot \omega_1 + \frac{C_1}{2l} \sum_{i=1}^{l} \frac{1}{\sigma_i^2} \cdot \xi_i^2 - \sum_{i=1}^{l} \eta_i \xi_i \\
+ \sum_{i=1}^{l} \alpha_i (y_i - \omega_1^T \cdot \Phi(x_i) - b_1 - \epsilon_1 - \xi_i).
\] (13)

To minimize \( L(\omega_1, b_1, \alpha, \xi) \), we derive partial derivative \( \partial L/\partial \omega_1 = 0 \), \( \partial L/\partial b_1 = 0 \), \( \partial L/\partial \alpha_i = 0 \), \( \partial L/\partial \xi_i = 0 \), respectively. Obtained by the KKT (Karush-Kuhn-Tucker) conditions, we have

\[
\begin{align*}
\nabla_{\omega_1} L(\omega_1) &= 0, \\
\nabla_{b_1} L(\omega_1) &= 0, \\
\nabla_{\xi} L(\omega_1) &= 0.
\end{align*}
\] (14)

Thus

\[
\omega_1 = \frac{1}{l} \sum_{i=1}^{l} (\alpha_i \cdot \Phi(x_i)), \quad \sum_{i=1}^{l} \alpha_i = 0, \quad \xi_i = \frac{1}{C_1} \cdot \alpha_i.
\] (15)

Substituting the extreme conditions into \( L(\omega_1, b_1, \alpha, \xi) \) and seeking maximum of \( \alpha \), we obtain the dual problem of the primal problem (11)

\[
\begin{align*}
\max_{\alpha} & \quad \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} (\alpha_i \cdot \alpha_j \cdot K(x_i, x_j)) \\
\text{s.t.} & \quad \sum_{i=1}^{l} \alpha_i = 0
\end{align*}
\]

**DTLSSVR-HGN**:

\[
\begin{align*}
\max_{\alpha} & \quad \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} (\alpha_i^* \cdot \alpha_j^* \cdot K(x_i, x_j)) \\
\text{s.t.} & \quad \sum_{i=1}^{l} \alpha_i^* = 0
\end{align*}
\]

where \( C_1 \) > 0 is penalty parameter. After \( \omega \) and \( \xi \), we can obtain the solution

\[
\begin{bmatrix}
0 \\
e \\
\hat{K}_1 \cdot \alpha
\end{bmatrix} = \begin{bmatrix}
0 \\
y - \epsilon_1 e
\end{bmatrix} \quad (17)
\]

where \( \hat{K}_{ij} = k(x_i, x_j) + \frac{\delta_{ij}}{\sigma_i^2} \), with \( \delta_{ij} = 1(i = j) \), or \( \delta_{ij} = 0(i \neq j) \) \( i, j = 1, 2, \ldots, l \).

Similarly, we have the dual problem of the primal problem (12)

\[
\begin{align*}
\max_{\alpha} & \quad \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} (\alpha_i^* \cdot \alpha_j^* \cdot K(x_i, x_j)) \\
\text{s.t.} & \quad \sum_{i=1}^{l} \alpha_i^* = 0
\end{align*}
\]

where \( C_2 \) > 0 is penalty parameter. After \( \omega^* \) and \( \xi^* \), we can obtain the solution

\[
\begin{bmatrix}
0 \\
e \\
\hat{K}_2 \cdot \alpha^*
\end{bmatrix} = \begin{bmatrix}
0 \\
y - \epsilon_2 e
\end{bmatrix}
\]

\[
\begin{align*}
\hat{K}_{ij} = k_2(x_i, x_j) + \frac{\delta_{ij}}{\sigma_i^{*2}C_2} \quad \text{or} \quad \delta_{ij} = 0(i \neq j) \quad i, j = 1, 2, \ldots, l.
\end{align*}
\]

The lower-bound nonlinear regression function \( f_1(x) = \omega_1^T \cdot \Phi(x) + b_1 \), and upper-bound function \( f_2(x) = \omega_2^T \cdot \Phi(x) + b_2 \) of TLSSVR-HGN can be obtained by solving...
where the above optimization problems, respectively. We obtain the lower-bound and upper-bound regressors of TLSSVR-HGN as
\[
f_1(x) = \sum_{i=1}^{l} (\alpha_i \cdot K(x_i, x)) + b_1,
\]
\[
f_2(x) = \sum_{i=1}^{l} (\alpha_i^* \cdot K(x_i, x)) + b_2.
\]

where
\[
b_1 = \frac{1}{l} \sum_{i=1}^{l} [y_i - \sum_{j=1}^{l} (\alpha_j \cdot K(x_i, x_j)) - \frac{l \cdot \sigma^2}{C_1} \cdot \alpha_i - \epsilon_1],
\]
\[
b_2 = \frac{1}{l} \sum_{i=1}^{l} [y_i - \sum_{j=1}^{l} (\alpha_j^* \cdot K(x_i, x_j)) - \frac{l \cdot \sigma^2}{C_2} \cdot \alpha_i^* + \epsilon_2].
\]

Once \(f_1(x)\) and \(f_2(x)\) are got, the end regressor of TLSSVR-HGN is constructed as
\[
f_{\text{TLSSVR-HGN}}(x) = \frac{f_1(x) + f_2(x)}{2} = \sum_{i=1}^{l} \alpha_i + \alpha_i^* \cdot K(x_i, x) + \frac{b_1 + b_2}{2}.
\]

Specially, the noise in Equation (1) is Gaussian, with zero mean and the homoscedastic variance \(\sigma^2\), following the idea of forming two hyperplanes in TSVR, the model of twin Least squares support vector regression of homoscedastic Gaussian noise model (TLSSVR-GN), also called twin kernel ridge regression of Gaussian noise model (TKRR-GN), can be formally defined as
\[
\min_{P_{\text{TKRR-GN}}} \{ g_{P_{\text{TLSSVR-GN}}} = \frac{1}{2} \omega^T \cdot \omega_1 + \frac{C_1}{2l} \sum_{i=1}^{l} \xi_i \}
\]
\[
P_{\text{TKRR-GN}} : \begin{align*}
y_i &= \omega_1^T \cdot \Phi(x_i) + b_1 + \epsilon_1 - \xi_i \\
&= (i = 1, \ldots, l)
\end{align*}
\]
and
\[
\min_{P_{\text{TKRR-GN}}} \{ g_{P_{\text{TLSSVR-GN}}} = \frac{1}{2} \omega^T \cdot \omega_2 + \frac{C_2}{2l} \sum_{i=1}^{l} \xi_i^2 \}
\]
\[
P_{\text{TKRR-GN}} : \begin{align*}
y_i &= \omega_2^T \cdot \Phi(x_i) + b_2 - \epsilon_2 + \xi_i^* \\
&= (i = 1, \ldots, l)
\end{align*}
\]
where \(\epsilon_1, \epsilon_2 \geq 0\) are threshold parameters, \(C_1, C_2 > 0\) are penalty parameters; whereas \(\xi_i, \xi_i^*(i = 1, \ldots, l)\) are random variables.

The lower-bound nonlinear regression function \(f_1(x) = \omega_1^T \cdot \Phi(x) + b_1\), and upper-bound function \(f_2(x) = \omega_2^T \cdot \Phi(x) + b_2\) of TLSSVR-GN can be obtained by solving the above optimization problems, respectively.

The dual problems of the primal problems (20) and (21) of TLSSVR – GN are
\[
\max \{ g_{D_{\text{TLSSVR-GN}}} = -\frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} (\alpha_i \cdot \alpha_j \cdot K(x_i, x_j)) + \frac{l}{2C_1} \cdot \frac{1}{2} \sum_{i=1}^{l} (\alpha_i^2) \}
\]
\[
D_{\text{TLSSVR-GN}} : \begin{align*}
\sum_{i=1}^{l} \alpha_i &= 0,
\end{align*}
\]
and
\[
\max \{ g_{D_{\text{TLSSVR-GN}}} = -\frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} (\alpha_i^* \cdot \alpha_j^* \cdot K(x_i, x_j)) + \frac{l}{2C_2} \cdot \frac{1}{2} \sum_{i=1}^{l} (\alpha_i^*^2) \}
\]
\[
D_{\text{TLSSVR-GN}} : \begin{align*}
\sum_{i=1}^{l} \alpha_i^* &= 0,
\end{align*}
\]
where \(C_1 > 0\) and \(C_2 > 0\) are penalty parameters.

**B. ALGORITHM OF AUGMENTED LAGRANGE MULTIPLIER METHOD**

Before dealing with the \(\text{eq}(17)\), we need to calculate \(v_i\). The parameters \(v_i\) is to make the regression hyperplane closer to \((x_i, y_i) \in X_f\). When the training errors occur in the training samples which lying down the \(\epsilon\)-insensitive lower-bound function, \(v_i\) will be weighting the training errors. In the beginning, we will make a rough estimate of the location of these samples using normal LSSVR [41], because we do not know which samples lie down the \(\epsilon\)-insensitive lower-bound function. From eq.(15), the slack errors \(\xi_i\) are proportional to the Lagrangian multiplier \(\alpha_i\). As shown below, the parameters \(v_i(i = 1, \ldots, l)\) are initialized.

\[
v_i = \begin{cases} 
\rho, & \text{if } \alpha_i < 0 \\
1, & \text{if } \alpha_i > 0
\end{cases}
\]

\(\rho = \frac{1}{l \alpha_i^2}\)

where \(\sigma_i\) is the heteroscedastic variance. Bring \(v_i(i = 1, \ldots, l)\) into eq.(17) to produce a new Lagrange multiplier \(\alpha\). If \(\alpha\) meets the above conditions eq.(24), \(v_i(i = 1, \ldots, l)\) are recomputed. When the index set \(S\) is no longer updated, this calculation stops iterating.

**Remark 1:** In step 2 of Algorithm 1, the computational complexity is \(O(l^2)\) when calculating the inverse matrix. In step 4, we calculate the inverse matrix. When the index
Algorithm 1 Lower-Bound Function of TLSSVR-HGN
1: **Initialization**: From the training data \(X_f\) get the kernel parameters, and determine \(\rho_1\) and \(\varepsilon_1\), and the regularization parameter \(C_1\);
2: From eq.(17) gets Lagrange multiplier sequence \(\alpha\), and get index set \(S_1\) whose elements correspond to \(\alpha_i<0\), where \(\alpha_i\) is the \(i\)th entry of vector \(\alpha\).
3: According to \(\alpha\), get \(v_i(i=1,\cdots,l)\)
   \[
   v_i = \begin{cases} \rho_1, & \text{if } \alpha_i < 0, \\ 1, & \text{if } \alpha_i \geq 0. \end{cases}
   \]
4: Plugging \(v_i(i=1,\cdots,l)\) into
   \[
   \begin{bmatrix} 0 \\ e^T \end{bmatrix} \begin{bmatrix} b_1^{\text{new}} \\ \alpha^{\text{new}} \end{bmatrix} = \begin{bmatrix} 0 \\ y - \varepsilon_1 e \end{bmatrix}
   \]
gets \(b_1^{\text{new}}, \alpha^{\text{new}}\) and index set \(S_1^{\text{new}}\).
5: if \(S_1 = S_1^{\text{new}}\) then
6: Let \(\alpha = \alpha^{\text{new}}, b_1 = b_1^{\text{new}}\) and go to step 10;
7: else
8: Let \(\alpha = \alpha^{\text{new}}, S_1 = S_1^{\text{new}}\) and go to step 3;
9: end if
10: Output the lower-bound function
   \[
   f_1(x) = \sum_{i=1}^{l} (\alpha_i \cdot K(x_i, x)) + b_1.
   \]

Remark 2: Similar to Remark 1, when the index set \(S\) does not change in Algorithm 2, we assume that the number of calculation cycles is \(l_1\), the computational complexity of this process is \(l_1 O(l^3)\). At last, the total computational complexity of Algorithm 1 is \((l_1 + 1) O(l^3)\).

Algorithm 2 Upper-Bound Function of TLSSVR-HGN
1: **Initialization**: From the training data \(X_f\) get the kernel parameters, and determine \(\rho_2\) and \(\varepsilon_2\), and the regularization parameter \(C_2\);
2: From eq.(19) gets Lagrange multiplier sequence \(\alpha^*\), and get index set \(S_2\) whose elements correspond to \(\alpha^*_i>0\), where \(\alpha^*_i\) is the \(i\)th entry of vector \(\alpha^*\).
3: According to \(\alpha^*\), get \(v_i(i=1,\cdots,l)\)
   \[
   v_i = \begin{cases} \rho_2, & \text{if } \alpha^*_i > 0, \\ 1, & \text{if } \alpha^*_i \leq 0. \end{cases}
   \]
4: Plugging \(v_i(i=1,\cdots,l)\) into
   \[
   \begin{bmatrix} 0 \\ e^T \end{bmatrix} \begin{bmatrix} b_2^{\text{new}} \\ \alpha^{\text{new}} \end{bmatrix} = \begin{bmatrix} 0 \\ y - \varepsilon_2 e \end{bmatrix}
   \]
gets \(b_2^{\text{new}}, \alpha^{\text{new}}\) and index set \(S_2^{\text{new}}\).
5: if \(S_2 = S_2^{\text{new}}\) then
6: Let \(\alpha^* = \alpha^{\text{new}}, b_2 = b_2^{\text{new}}\) and go to step 10;
7: else
8: Let \(\alpha^* = \alpha^{\text{new}}, S_2 = S_2^{\text{new}}\) and go to step 3;
9: end if
10: Output the upper-bound function
   \[
   f_2(x) = \sum_{i=1}^{l} (\alpha^*_i \cdot K(x_i, x)) + b_2.
   \]

Remark 3: By Remark 1 and Remark 2 we can get the total computational complexity is \((l_1 + l_2 + 2) O(l^3)\).

Algorithm 3 Sparse TLSSVR-HGN (STLSSVR-HGN)
1: Given the complete training set \(X_f\), get the TLSSVR-HGN model by Algorithms 1 and 2;
2: For \(X_f\) in step 1, remove a small amount of training samples(e.g. 5% of the set) with smallest values \(|\alpha_i + \alpha^*_i|\) in the spectrum;
3: Re-implement Algorithms 1 and 2 to get the TLSSVR-HGN based on the reduced training samples;
4: if the performance of TLSSVR-HGN degrades then
5: Go to step 9;
6: else
7: Go to step 2;
8: end if
9: Output the function of STLSSVR-HGN.

Remark 4: We assume that the number of computing TLSSVR-HGN model is \(n\). Therefore, the computational complexity of Algorithm 3 is \(n(l_1 + l_2 + 2) O(l^3)\), where the range of \(n\) will not be larger than our experiment.
IV. EXPERIMENTS AND DISCUSSION

In this section, to check the regression performance of TLSSVR-HGN, we compare it with STLSSVR-HGN, TLSSVR-GN, TLSSVR, LSSVR and SVR on several data sets, including the artificial data set and UCI data set. To further verify its validity, TLSSVR-HGN is used to predict the real-world application of short-term wind in a final experiment.

In all experiments, the regression algorithms are implemented in the Python 3.7 environment on Windows 7 running on a PC with 4 GB of RAM. The initial parameters of the proposed ALM approach are \( C \in [0.5, 250] \). We use 10-fold cross validation strategy to search the optimal positive parameters \( C \) [43]. To reduce the computational complexity of parameter selection, we set \( C_1 = C_2 = \epsilon_1 = \epsilon_2 \). For STLSSVR-HGN, we set the number of reduced samples to be five percent of the total training samples in Algorithm 3. Many actual applications suggest that Gaussian kernel function tend to perform well under general smoothness assumptions. In this work, polynomial function and Gaussian kernel function are used as the kernel function of different models.

\[
K(x_i, x_j) = ((x_i, x_j) + 1)^d
\]

where \( d \) is a positive integer, \( \gamma \) is a tuned parameter and determined from the interval \([0.1, 10]\) by cross-validation. Similarly, we also set \( \gamma_1 = \gamma_2 \) to reduce the computational burden of parameter selection.

In order to report the performance of the aforementioned algorithms, the evaluation criteria are specified before presenting experimental results. We introduce the following criterions, mean absolute error (MAE), root mean square error (RMSE), sum of squared errors (SSE), total sum of squares (SST), and sum of squares of the regression (SSR) to evaluate the performance of different models. Obtaining a smaller SSE/SST is usually accompanied by an increase in SSR/SST [25].

\[
MAE = \frac{1}{l} \sum_{i=1}^{l} |y_i - y^*_i| \quad (25)
\]

Formula(25): Mean absolute error.

\[
RMSE = \sqrt{\frac{1}{l} \sum_{i=1}^{l} (y_i - y^*_i)^2} \quad (26)
\]

Formula(26): Root mean square error.

\[
SSE = \sum_{i=1}^{l} (y_i - y^*_i)^2 \quad (27)
\]

Formula(27): Sum of squared errors.

\[
SST = \sum_{i=1}^{l} (y_i - \bar{y})^2 \quad (28)
\]

Formula(28): Total sum of squares.

\[
SSR = \sum_{i=1}^{l} (y^*_i - \bar{y})^2 \quad (29)
\]

Formula(29): Sum of squares of the regression.

\[
SSE/SST = \sum_{i=1}^{l} (y_i - y^*_i)^2 / \sum_{i=1}^{l} (y_i - \bar{y})^2 \quad (30)
\]

Formula(30): The ratio of the sum of squares of the testing samples to the sum of squares of the deviations.

\[
SSR/SST = \sum_{i=1}^{l} (y^*_i - \bar{y})^2 / \sum_{i=1}^{l} (y_i - \bar{y})^2 \quad (31)
\]

Formula(31): The ratio between the explanatory sum of squares deviation of the testing samples and the actual sum of squares.

Where \( l \) is the size of the samples, \( y_i \) is the real data, \( y^*_i \) is the prediction results, \( \bar{y} = \frac{1}{l} \sum_{i=1}^{l} y_i \) is the mean of testing samples. In addition, \#SV represents the number of support vectors, CPU time denotes the time of testing samples [26].

A. ARTIFICIAL DATA SET

First of all, we compare the six models on the \( sinc(X) \) function, which is used to test the performance of the regression. It is shown as

\[
sinc(X) = \frac{\sin(\pi X)}{\pi X}, \quad X \in [-4, 4]. \quad (32)
\]

To effectively check the performance of the proposed method, the training samples are perturbed by Gaussian noise in the experiment. The \( sinc(X) \) function with noise are shown as

\[
y_i = \frac{\sin(\pi X)}{\pi X} + \xi_i, \quad X \in [a, b], \quad \xi_i \sim N(0, 0.15^2). \quad (33)
\]

where \( U [a, b] \) represents the uniformly random variable in \([a, b]\) and \( N(c, d^2) \) represents the Gaussian random variable with mean \( c \) and variance \( d^2 \).

We sample 200 training samples and 200 testing samples, the testing samples are uniformly generated from the \( sinc(X) \) function. In addition, the testing samples do not contain any noise. Fig. 2 shows the prediction results of different models on the \( sinc(X) \) function. Fig. 3 shows the prediction errors of four models on the \( sinc(X) \) function. Table 1 shows the result comparisons of six models on the \( sinc(X) \) function.

Table 1 shows TLSSVR-HGN derives the smallest MAE, RMSE, SSE and SSE/SST, and the largest SSR/SST among these algorithms. In addition, its generalization performance is best, i.e., it owns smallest and largest evaluation criteria, respectively, viz. RMSE and SSR/SST. Fig. 3 shows the prediction error of TLSSVR-HGN is closer to zero than other three algorithms, Fig. 2 shows the fitting capacity of TLSSVR-HGN is perfect. However, like other algorithms such as TLSSVR-GN and TLSSVR, the solution of
FIGURE 2. Predictions of four models on sinc($x$) function.

TABLE 1. Result comparisons of six models on sinc($x$) function.

| Model       | MAR  | RMSE | SSE  | SSE/SST | SSR/SST | #SV  | CPU time(s) |
|-------------|------|------|------|---------|---------|------|-------------|
| TLSSVR-HGN  | 0.0027 | 0.0048 | 0.0046 | 0.0002 | 0.9766 | 200  | 3.18        |
| STLSSVR-HGN | 0.0033 | 0.0059 | 0.0069 | 0.0003 | 0.9712 | 120  | 3.16        |
| TLSSVR-GN   | 0.0084 | 0.0098 | 0.0191 | 0.0008 | 0.9594 | 200  | 3.39        |
| TLSSVR      | 0.0124 | 0.0144 | 0.0414 | 0.0017 | 0.9394 | 200  | 3.22        |
| LSSVR       | 0.0180 | 0.0237 | 0.0561 | 0.1755 | 0.4719 | 200  | 1.74        |
| SVR         | 0.0161 | 0.0211 | 0.0893 | 0.0036 | 0.8861 | 200  | 4.33        |

FIGURE 3. Prediction errors of four models on sinc($x$) function.

TLSSVR-HGN is not sparse, so we sparse TLSSVR-HGN, which is named STLSSVR-HGN. From Table 1, we also compare the testing CPU time for these methods. Except for the LSSVR, the results indicate STLSSVR-HGN takes obvious advantage in the testing time. The reason is that the number of support vectors is reduced. In short, our TLSSVR-HGN outperforms TLSSVR-GN, TLSSVR, LSSVR and SVR on approximating this artificial function.

B. UCI DATA SET

For this part, to further evaluate the regression performance of the proposed method, we use two UCI data sets [44], including the Boston house price and stock. For the two data sets, the parameters of this model are obtained using 10-fold cross validation. Each group of UCI data set is divided into 100 training samples and 100 testing samples.

On the stock market, we can use the market index to predict the overall trend of the stock. Figs. 4-7 show the prediction results of TLSSVR-HGN, STLSSVR-HGN, TLSSVR and
TABLE 2. Result comparisons of six models on stock.

| Model       | MAE  | RMSE | SSE  | SSE/SST | SSR/SST | #SV | CPU time(s) |
|-------------|------|------|------|---------|---------|-----|-------------|
| TLSSVR-HGN  | 0.0121 | 0.0148 | 0.0219 | 0.0686 | 0.8032 | 100 | 0.95        |
| STTLSSVR-HGN| 0.0122 | 0.0150 | 0.0225 | 0.0704 | 0.7963 | 65  | 0.91        |
| TLSSVR-GN   | 0.0154 | 0.0187 | 0.0348 | 0.1089 | 0.7931 | 100 | 0.93        |
| TLSSVR      | 0.0159 | 0.0193 | 0.0372 | 0.1165 | 0.7648 | 100 | 0.92        |
| LSSVR       | 0.0180 | 0.0237 | 0.0561 | 0.1755 | 0.4719 | 100 | 0.35        |
| SVR         | 0.0152 | 0.0194 | 0.0375 | 0.1174 | 0.6253 | 100 | 2.08        |

TABLE 3. Result comparisons of six models on house price.

| Model       | MAE  | RMSE | SSE  | SSE/SST | SSR/SST | #SV | CPU time(s) |
|-------------|------|------|------|---------|---------|-----|-------------|
| TLSSVR-HGN  | 0.0154 | 0.0203 | 0.0414 | 0.0095 | 0.8577 | 100 | 0.90        |
| STTLSSVR-HGN| 0.0165 | 0.0218 | 0.0475 | 0.0109 | 0.8460 | 58  | 0.87        |
| TLSSVR-GN   | 0.0323 | 0.0410 | 0.1681 | 0.0386 | 0.8326 | 100 | 0.90        |
| TLSSVR      | 0.0358 | 0.0456 | 0.2084 | 0.0479 | 0.8012 | 100 | 0.91        |
| LSSVR       | 0.0771 | 0.1280 | 1.6386 | 0.3765 | 0.5386 | 100 | 0.43        |
| SVR         | 0.0363 | 0.0504 | 0.2540 | 0.0584 | 0.6249 | 100 | 2.14        |

TLSSVR-GN on the stock, respectively. Fig. 8 shows the prediction errors of four models on the stock. Table 2 lists the error statistics of TLSSVR-HGN, STTLSSVR-HGN, TLSSVR, TLSSVR-GN, LSSVR and SVR models on the stock. On the Boston house price, it has 13 features that determine the trend of the house price. Figs. 9-12 show the prediction results of TLSSVR-HGN, STTLSSVR-HGN, TLSSVR and TLSSVR-GN on the house price, respectively. Fig. 13 lists the statistics of house price errors for the four models. Table 3 shows the error statistics of six models on the Boston house price.

From Tables 2 and 3, TLSSVR-HGN obviously outperforms the other five methods. We can see that TLSSVR-HGN
Table 4. Result comparisons of six models on wind speed after 10 min.

| Model         | MAE     | RMSE    | SSE     | SSE/SST | SSR/SST | #SV  | CPU time(s) |
|---------------|---------|---------|---------|---------|---------|------|-------------|
| TLSSVR-HGN    | 0.1244  | 0.1526  | 3.2386  | 0.0093  | 0.8200  | 100  | 0.87        |
| STLSSVR-HGN   | 0.1336  | 0.1638  | 2.6842  | 0.0107  | 0.8074  | 73   | 0.84        |
| TLSSVR-GN     | 0.1592  | 0.1954  | 3.8183  | 0.0152  | 0.7699  | 100  | 0.88        |
| TLSSVR        | 0.2246  | 0.2743  | 7.5235  | 0.0300  | 0.6854  | 100  | 0.91        |
| LSSVR         | 0.3675  | 0.4525  | 20.4728 | 0.0816  | 0.5102  | 100  | 0.37        |
| SVR           | 0.2858  | 0.3519  | 12.3847 | 0.0494  | 0.6049  | 100  | 4.36        |

Table 5. Result comparisons of six models on wind speed after 30 min.

| Model         | MAE     | RMSE    | SSE     | SSE/SST | SSR/SST | #SV  | CPU time(s) |
|---------------|---------|---------|---------|---------|---------|------|-------------|
| TLSSVR-HGN    | 0.1498  | 0.1982  | 3.9269  | 0.0256  | 0.7206  | 100  | 0.94        |
| STLSSVR-HGN   | 0.1544  | 0.2041  | 4.1656  | 0.0272  | 0.7128  | 78   | 0.88        |
| TLSSVR-GN     | 0.1667  | 0.2106  | 4.4383  | 0.0290  | 0.6908  | 100  | 0.94        |
| TLSSVR        | 0.2303  | 0.2886  | 8.3278  | 0.0543  | 0.5905  | 100  | 0.99        |
| LSSVR         | 0.4287  | 0.5279  | 27.8659 | 0.1111  | 0.4445  | 100  | 0.47        |
| SVR           | 0.3573  | 0.4399  | 19.3512 | 0.0772  | 0.5216  | 100  | 5.38        |

FIGURE 11. Prediction of TLSSVR on house price.

FIGURE 12. Prediction of TLSSVR-GN on house price.

Obtains the largest SSR/SST on the two data sets and keeps the smallest SSE/SST among the six methods. In addition, the experiment results show STLSSVR-HGN still takes obvious advantage for STLSSVR-HGN, TLSSVR, TLSSVR-GN and SVR in the testing CPU time. However, compared to TLSSVR-HGN, STLSSVR-HGN has larger RMSE and smaller SSR/SST. Therefore, STLSSVR-HGN is not as good as TLSSVR-HGN in terms of generalization capabilities. Figs. 8 and 13 show that the error of TLSSVR-HGN is closest to zero among the four algorithms. In conclusion, the proposed method has the good regression performance, STLSSVR-HGN has smaller testing CPU time than TLSSVR-HGN, TLSSVR-GN, TLSSVR and SVR.

C. SHORT-TERM WIND SPEED FORECASTING

To show the effectiveness of the proposed model, we collected wind speed dataset from Heilongjiang Province, China. Data-set consists of more than 1-year wind speeds, recording the average wind-speed values every 10 min. As a whole, these samples with 4 attributes, mean, variance, minimum, maximum, are given. We analyze one-year time series of wind speeds. The result shows that the error of wind speed in one or two hours with the persistence forecast ([19], [45]) satisfy the Gaussian distribution. However, their variances change with the average wind speed. This is a heteroscedastic kind of task.

Now we test the performance of different techniques on wind speed prediction. The experiment setup is described as follows. We transform the original series into a multivariate task by using a 12-dimensional vector $x_i = (x_{i-11}, x_{i-10}, \ldots, x_{i-1}, x_i)$ as input variables to predict $x_{i+\text{step}}$, where step $= 1, 3$. That is to say, we use the above model to predict the wind speed after 10, 30 minutes of every point $x_i$, respectively.

On the wind speed, we divide the data set into 100 training samples and 100 testing samples. To obtain the model parameters, we use 10-fold cross validation. In the data set, the next two sample points are separated by 10 minutes, so we first predict a sample point every 10 minutes and then a sample point every 30 minutes. Figs. 14-17 respectively
shows the wind speed prediction result of TLSSVR-HGN, STLSSVR-HGN, TLSSVR-GN and TLSSVR after 10 minutes. Fig. 18 shows the prediction error results of the four models on wind speed after 10 minutes. Similarly, Figs. 19-22 respectively shows the wind speed prediction result of TLSSVR-HGN, STLSSVR-HGN, TLSSVR-GN and TLSSVR after 30 minutes. Fig. 23 shows the prediction error results of four models on wind speed after 30 minutes. Table 4 shows the prediction error statistics of the six models on wind speed after 10 minutes. Table 5 shows the prediction error statistics of the six models on wind speed after 30 minutes.

Tables 4 and 5 show that compared with other algorithms, TLSSVR-HGN has lower MAE, RMSE, SSE, SSE/SST and higher SSR/SST. This indicates that TLSSVR-HGN is an
effective method for predicting wind speed. Fig. 18 and 23 show that compared with other algorithms, the prediction result of TLSSVR-HGN still is closer to the original data. In other words, the fitting result of TLSSVR-HGN is better than other three models. From Tables 4 and 5, we can also see that compared with TLSSVR-GN, TLSSVR, LSSVR and SVR, STLSSVR-HGN has smaller RMSE and larger SSR/SST; it indicates STLSSVR-HGN has better generalization performance than TLSSVR-GN, TLSSVR, LSSVR and SVR. In addition, STLSSVR-HGN still has the smallest testing CPU time among TLSSVR-HGN, TLSSVR-GN, TLSSVR and SVR. Hence, it is necessary to develop STLSSVR-HGN.

Observing from the error statistics in Tables 1-5, we can know TLSSVR-HGN model is better than STLSSVR-HGN, TLSSVR, TLSSVR-GN, LSSVR and SVR models in terms of some criterions (MAE, RMSE, SSE, SSE/SST and SSR/SST). In addition, we can also know LSSVR has the smallest testing CPU time among the six algorithms. In short, this experiments on the artificial data set, several benchmark data sets and the real world data of wind speed confirm the feasibility and efficacy of TLSSVR-HGN and STLSSVR-HGN furthermore.

V. CONCLUSIONS

In this paper, we propose a new regression technique, referred to TLSSVR-HGN. The classical TLSSVR, TLSSVR-GN, LSSVR and SVR techniques make the assumption that the error distribution is Gaussian. However, it was reported that the noise models in some real-world applications do satisfy heteroscedasticity. Thus, it is necessary to use a new technique to build prediction models. In this work, we derive a heteroscedastic noise loss function and develop a new technique by proposing a twin regressor of heteroscedastic noise model. According to KKT conditions, we introduce the Lagrangian functional and obtain the dual problem of TLSSVR-HGN. The ALM approach is applied to solve TLSSVR-HGN. To analyze the generalization ability of the proposed model, we propose sparse TLSSVR-HGN (STLSSVR-HGN). Finally, experimental results on the artificial data set, several UCI data sets and the real-world data of wind speed confirm the effectiveness of the proposed technique. We compared different models on stock, Boston house price and wind speed data set, it is concluded that TLSSVR-HGN has better generalization performance than other algorithms, and it is better than other models at predicting data.

This work just discussed the problem of regression models with Gaussian heteroscedasticity. In fact, there is a similar challenge in classification learning. In a similar manner, we can study the classification problem for the heteroscedastic noise model in the future.

REFERENCES

[1] V. N. Vapnik, The Nature of Statistical Learning Theory. New York, NY, USA: Springer, 1995.
[2] V. N. Vapnik, Statistical Learning Theory. New York, NY, USA: Wiley, 1998.
[3] C. J. C. Burges, “A tutorial on support vector machines for pattern recognition,” Data Mining Knowl. Discovery, vol. 2, no. 2, pp. 121–167, 1998.
[4] E. Osuna, R. Freund, and F. Girosi, “Training support vector machines: An application to face detection,” in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., 1997, pp. 130–136.
[5] S. Lee, C. Park, and J.-Y. Koo, “Feature selection in the Laplacian support vector machine,” Comput. Statist. Data Anal., vol. 55, no. 1, pp. 567–577, Jan. 2011.
[6] C.-C. Chuang, S.-F. Su, J.-T. Jeng, and C.-C. Hsiao, “Robust support vector regression networks for function approximation with outliers,” IEEE Trans. Neural Netw., vol. 13, no. 6, pp. 1322–1330, Nov. 2002.
[7] T. B. Trafalis and H. Ince, “Support vector machine for regression and applications to financial forecasting,” in Proc. IEEE-INNS-ENNS Int. Joint Conf. Neural Netw. (IJCNN) Neural Comput., New Challenges Perspect. New Millennium, Jul. 2000, pp. 348–353.
[8] J. L. Platt, “Fast training of support vector machines using sequential minimal optimization,” in Advances in Kernel Methods–Support Vector Learning. B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds. Cambridge, MA, USA: MIT Press, 1999, pp. 185-2008.
[9] R. Collobert and S. Bengio, “SVMTorch: Support vector machines for large-scale regression problems,” J. Mach. Learn. Res., vol. 1, no. 2, pp. 143–160, 2001.
[10] C. C. Chang and C. J. Lin. (2001). LIBSVM: A Library for Support Vector Machines. [Online]. Available: http://www.csie.ntu.edu.tw
[11] J. A. K. Suykens, L. Lukas, P. Van Dooreen, B. De Moor, and J. Vandewalle, “Least squares support vector machine classifiers: A large scale algorithm,” in Proc. Eur. Conf. Circuit Theory Design, 1999, pp. 839–842.
[12] P.-Y. Hao, “New support vector algorithms with parametric insensitive/margin model,” Neural Netw., vol. 23, no. 1, pp. 60–73, Jan. 2010.
[13] O. L. Mangasarian and E. W. Wild, “Multisurface proximal support vector machine classification via generalized eigenvalues,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 28, no. 1, pp. 69–74, Jan. 2006.
[14] Jayadeva, R. Khemchandani, and S. Chandra, “Twin support vector machines for pattern classification,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 22, no. 5, pp. 905–910, May 2007, doi: 10.1109/TPAMI.2007.1068.
[15] S. Ding, X. Zhao, J. Zhang, X. Zhang, and Y. Xue, “A review on multi-class TWSVM,” Artif. Intell. Rev., vol. 52, no. 2, pp. 775–801, Aug. 2019.
[16] H. Cong, C. Yang, and X. Pu, “Efficient speaker recognition based on writer-Independent parameters,” Pattern Recognit., vol. 48, no. 1, pp. 103–113, Jan. 2015.
[17] H. Yin, J. Qiao, P. Fu, and X. Y. Xia, “Face feature selection with binary particle swarm optimization and support vector machine,” J. Inf. Hiding Multimedia Signal Process. vol. 5, no. 4, pp. 731–739, 2014.
[18] Y. Guerbai, Y. Chibani, and B. Hadjadji, “The effective use of the one-class SVM classifier for handwritten signature verification based on writer-independent parameters,” Pattern Recognit., vol. 48, no. 1, pp. 103–113, Jan. 2015.
[19] Q. Hu, S. Zhang, M. Yu, and Z. Xie, “Short-term wind speed or power forecasting with heteroscedastic support vector regression,” IEEE Trans. Sustain. Energy, vol. 7, no. 1, pp. 241–249, Jan. 2016.
[20] A. Kim and S.-B. Cho, “An ensemble semi-supervised learning method for predicting defaults in social lending,” Eng. Appl. Artif. Intell., vol. 81, pp. 193–199, May 2019.
[21] P. Pouladzadeh, S. Shirmohammadi, A. Bakirov, A. Bulut, and A. Yassine, “Cloud-based SVM for food categorization,” Multimedia Tools Appl., vol. 74, no. 14, pp. 5243–5260, Jul. 2015.

[22] J. López, S. Maldonado, and M. Carrasco, “A novel multi-class SVM model using second-order cone constraints,” Int. J. Speech Technol., vol. 44, no. 2, pp. 457–469, Mar. 2016.

[23] A. Isotidis and M. Gabbouj, “Multi-class support vector machine classifiers using intrinsic and penalty graphs,” Pattern Recognit., vol. 55, pp. 231–246, Jul. 2016.

[24] J. A. K. Suykens, J. Vandewalle, and B. De Moor, “Optimal control by least squares support vector machines,” Neural Netw., vol. 14, no. 1, pp. 23–35, Jan. 2001.

[25] X. Peng, “TSVR: An efficient twin support vector machine for regression,” Neural Netw., vol. 23, no. 3, pp. 365–372, Apr. 2010, doi: 10.1016/j.neunet.2009.07.002.

[26] Y.-P. Zhao, J. Zhao, and M. Zhao, “Twin least squares support vector regression,” Neurocomputing, vol. 118, pp. 225–236, Oct. 2013.

[27] N. Parastaloop, A. Amiri, and P. Alieheidari, “Modified twin support vector regression,” Neurocomputing, vol. 211, pp. 84–97, Oct. 2016.

[28] X. Peng, “Primal twin support vector regression and its sparse approximation,” Neurocomputing, vol. 73, nos. 16–18, pp. 2846–2858, Oct. 2010.

[29] Y.-H. Shao, C.-H. Zhang, Z.-M. Yang, L. Jing, and N.-Y. Deng, “An ϵ-twin support vector machine for regression,” Neural Comput. Appl., vol. 23, no. 1, pp. 175–185, 2013.

[30] R. Khemchandani, K. Goyal, and S. Chandra, “TWSVR: Regression via twin support vector machine,” Neural Netw., vol. 74, pp. 14–21, Feb. 2016.

[31] J. Bi and K. P. Bennett, “A geometric approach to support vector regression,” Neurocomputing, vol. 55, nos. 1–2, pp. 79–108, Sep. 2003.

[32] W. Chu, S. S. Keerthi, and C. J. Ong, “Bayesian support vector regression with its application to short-term wind speed forecasting,” in Proc. 21st Century IEEE Int. Symp. Circuits Syst., Emerg. Technol., 2000, pp. 757–760.

[33] C. L. Blake and C. J. Merz. (1998). UCI repository of machine learning databases. Dept. Inf. Comput. Sci., Univ. Irvine. [Online]. Available: http://kdd.ics.uci.edu/

[34] LIN SUN (Member, IEEE) received the B.Sc. and M.S. degrees in computer science and technology from Henan Normal University, in 2003 and 2006, respectively, and the Ph.D. degree in pattern recognition and intelligent systems from the Beijing University of Technology, in 2010. He is currently an Associate Professor with the College of Computer and Information Engineering, Henan Normal University, China. He became a Postdoctoral Researcher with the Medical and Biological Engineering Research Group, Henan Normal University, China, in 2016. He has received funding from ten grants from the National Science Foundation of China, the China Postdoctoral Science Foundation, the Science and Technology Innovation Talents Project of Henan Province, and the Key Project of Science and Technology of Henan Province. He has authored or coauthored for over 70 articles. He has written three books on the topic of intelligent information processing. He holds 15 patents of invention. His main research interests include granular computing, optimization algorithms, intelligent information processing, and data mining. He has served as a reviewer for several prestigious peer-reviewed international journals.

[35] TING ZHOU received the B.Sc. degree from Shaanxi Normal University, in 2011, and the M.S. degree from Liaocheng University, in 1998. She is currently working with State-Owned Assets Management Office Henan Normal University, China. She is also a Lecturer at Henan Normal University. Her research interests include focused on machine learning and knowledge discovery.

[36] CHAO LIU received the B.Sc. degree from Shaanxi Normal University, in 2013, and the M.S. degree from Liaocheng University, in 1998. She is currently pursuing the master’s degree with the College of Computer and Information Engineering, Henan Normal University, China. Her main research interests include machine learning and knowledge discovery for regression.

[37] SHIGUANG ZHANG received the B.Sc. degree in mathematics from Shaanxi Normal University, in 1998, the M.S. degree in mathematics from Guangxi University for Nationalities, in 2007, and the Ph.D. degree in applied mathematics from Hebei Normal University, in 2014. He is currently working with the College of Computer and Information Engineering, Henan Normal University, China. He has authored more than 20 peer-reviewed articles. He has served as a reviewer in several prestigious peer-reviewed international journals. His research interests include optimization algorithms, machine learning, and knowledge discovery for regression.